# OpBible

## Automatic Study Bible Typesetting Using TEX

***Version 0.31***

*Petr Olšák, Šimon Konečný*

*Revised 25. 10. 2023*

http://opbible.org

*It is too small a thing for you to be my servant*
*to restore the tribes of Jacob*
*and bring back those of Israel I have kept.*
*I will also make you a light for the Gentiles,*
*that my salvation may reach to the ends of the earth.*

*Isaiah 49:6*

# Preface

The verse from Isaiah 49 on the title page is inspiring: It would be a small thing to write and typeset using TEX just one Bible commentary on one Bible version as one's life's work. The OpBible offers the opportunity to write a commentary on any Bible version (up to as many as there are Bible translations in one particular language) to anyone anywhere; once the notes apparatus is complete, the study Bible is typeset.

Šimon Konečný's long-standing vision of using TEX to automatically typeset study Bibles came to fruition through a collaboration with Petr Olšák. Starting from the realistic assumption that those who will use OpBible the most will also be those who have the least experience with TEX, we tried to make the usage as simple as possible.

However, there is no point in pretending that typesetting a study Bible is something trivial. To claim that it is a little tricky is a bit understatement; it is rather quite a challenge for any programmer.

Our intention to allow the production of a single commentary apparatus to go with multiple Bible translations simultaneously was further complicated by the fact that different versions of the Bible have different verse numbers in some places, differently broken paragraphs, different structures of poetic passages, different placement of headings added by the translators; that some translations have completely different book titles (and hence abbreviations), that file names cannot contain diacritics but book references can and should, and countless other such oddities.

All the problems we encountered have been overcome and everything works just fine to our satisfaction. The resulting collection of macros is quite complicated, therefore the definitions usage is irritatingly sensitive to any transgressions against TEX's syntax.

That's why OpBible includes tools to make it easier to find where typos, forgotten parentheses, etc. have been left; and in this documentation, wherever we have seen fit, we add paragraphs entitled **What to watch out for** and **Recommended**.

Petr Olšák
Šimon Konečný

# Contents

# 1 What makes the OpBible specific?

The main advantages of the OpBible over traditional typesetting are:

- **Price:**
  - The software itself is free as public domain under an Open Source license (see Preface).
  - More important, however, is the price which you'll save on a typesetter you'd otherwise have to pay for several years to hand-typeset the Study Bible.
- **Time:** Typesetting an entire Study Bible takes minutes, not years. When providing the full text of a Study Bible that breaks up the pages so that the notes end up on the same pages with the verses they're commenting on, it took about ten minutes on a system with a quad-core Ryzen 3 processor. If you're working on a particular book and only processing one, it's usually a matter of seconds.
- **Flexibility:** The result is not just one single version of the Bible in which you cannot make any changes. If you later decide to edit some notes, delete some or add some more, etc., you have this new revised version immediately typeset and ready for print.
- **Variability:** The result may not be just one Bible. The notes can be written in a way that allows as many Bibles as there are different translations in a given language to be typeset with the same note apparatus. For example, consider these six English translations of the Bible: (1) Bible in Basic English (BBE); (2) Jubilee2000; (3) NETfree; (4) Updated King James Version (UKJV); (5) Restored Names King James Version (RNKJV); and (6) Webster. Now let's take a brief look at Daniel, chapter 2, where the animals, countries, and other stuff of Daniel's vision are rendered as follows:
  - *Bear* is the same everywhere but
  - *Leopard* of most tranlations is *Tiger* in Jubilee2000;
  - usual *Ram* becomes *Male sheep,* as well as *clay* changes into *potter's earth* in BBE;
  - ordinary *brass* turns into *bronze* in NETfree;
  - *Greece* of BBE, Jubilee2000, and NETfree is *Grecia* in UKJV, RNKJV, and Webster; compare all six translations.

  You can write your commentary in the way that the result would be all six translations at the same time, with no need to modify the notes file, while the notes (or charts, maps, etc.), will display the phrases as they appear in your current Bible. See the sample book of Daniel, chapter two, the chart of Daniel's visions; compare all six translations.

  In English, there are over 450 different translations of the Bible. Most of them are not being used anymore, but still, you can choose from more than sixty various English versions. Actually, you can write your notes in a way that sixty (or four hundred and fifty) various translations can be used with them at the same time, without a need to change anything in the notes.
- **Interactive output format**: The result of the processing is a PDF file that is richly interwoven with active links. These make an invaluable aid for proof-reading before submission to the print (see below).
- **Precision:**
  - The average study Bible contains around 20,000 notes and within them around 60 to 80,000 references to Bible verses and other notes. The notes are written by humans; it is inevitable that they will contain errors from oversight or typos. It is practically beyond the human power to find and correct all of them. It is, however, within the power of a machine. OpBible prints warning messages if, for example, a note refers to a non-existent note or verse.
  - You can check for the accurateness of references to verses and notes that do exist (thus they do not trigger an error message), but for some reason they are not the right ones. The control is made possible by the fact that all references are active links that display the immediate context when the cursor hovers over the linked location (if you are viewing the PDF with the correct version of Evince, for example). And when clicked on, it jumps straight there.
  - The phrase that the note comments on is highlighted with a different font (e.g.,bold). Moreover, OpBible will also search for it in the Bible text inside the verse in question, and the pages will be broken in the way that the note ends up on the same page as the phrase it is commenting on. In case that there are two notes commenting on two different words in the same verse, and that the page break will come between those words, the notes will follow the phrases on their pages. See 5.2
  - The Bible texts are prepared (for example, by downloading from the internet) in separate files and there is no need for them to be modified in any way (assuming your source left

no misprints in them). All notes on them and other typesetting instructions are written in separate files. Then TEX puts everything together.

**What to watch out for:** Although the intent of this program is to spread God's Word and the Good news contained therein, the OpBible itself *forgives nothing!* (After all, what'd you expect, it's a software. Go to God for forgiveness.) As you'll learn below, the OpBible loads one entire book of the Bible into memory at a time and only then begins to digest your notes and piece them together into pages with their respective verses. If you happen to make a mistake in TEX syntax (e.g., forget a closing bracket, etc.), TEX will see the error somewhere in the middle of this matching procedure, and this in turn will result in a cluttered error message, in which it is highly likely that you will get lost and not find your way around.

The OpBible anticipates the occurrence of situations like this and offers useful tools to help you out of similar predicaments, but you'd better be careful and consistent. You won't go wrong with the first written note, to go through all the translations you plan to use to make sure everything works as it should. After that you may only work with one of them, but it is advisable to run TEX every time after you finish each new note, so by its protests you can easily find the one in which you have the mistake.

**Recommended:** Use the Linux operating system. OpBible, as a macro collection for TEX, will run on any system with the same results, but the Linux installation has several not-to-be-missed advantages. Among the most significant of these is the Evince PDF viewer, whose newer (as of 2023) version can display the immediate context of a hyperlinked reference by simply hovering the cursor over it without having to click on it. This is an invaluable aid for checking the accuracy of links to Bible passages or notes. Other systems do not (yet) offer this facility. While the hyperlinks in these systems do jump to the required location when clicked, they do not hit back to where the jump signal was sent from, which is tedious. In this documentation, we assume a Linux installation.

And of course, we recommend – or rather, consider it a prerequisite – at least basic literacy in TEX. If you don't have any experience with TEX, try starting here[1]. The time invested in this preliminary education will not be wasted; the more you understand TEX, the better (and more joyfully) you will write your notes.

# 2 What we need to run OpBible

It is necessary to have some kind of text editor that does not keep any hidden formatting information in the text (designed for editing programs, for example). It is up to the user what kind of editor suits them. The ideal editor is one that recognizes the programming language by its source file's extension and colors words according to its syntax. E.g., Vi, Emacs, ... choose, whichever would suit you.

You also need to have a TEX distribution with the LuaTEX program and OpTEX default macros, and finally you need some kind of output (i.e., PDF files) viewer (we recommend the last version of Evince).

It isn't important on which operating system you will run this, but we recommend Linux.

It is also possible to work in online mode without installing anything, see section 2.2.

## 2.1 TEX at the local machine

We recommend the latest TEXlive distribution. It can be installed on any operating system directly from the web. It is also included in common Linux distributions.

TEXlive contains the program LuaTEX, which will process your input files and produce output PDFs. It also includes the default macro set OpTEX that the OpBible macros need, and extends it with the options described in this manual. The default set of macros (often called format) defines how documents are marked up and controls formatting. The TEX distributions include other default macro sets. The best known is probably LATEX, but it is not needed for OpBible.

## 2.2 Variant: remote access on Overleaf.com

Overleaf is a web-based environment for shared preparation of TEX documents, perhaps by multiple users. You do not need to have TEXlive installed; you can run it online via Overleaf. The default

[1] This text is, to a degree, obsolete, as it refers to DVI output of TEX. Nowadays, modern TEX ouputs directly to the PDF format.

project containing files for processing by OpBible is also available. As an Overleaf user, you can copy (clone) it to your project and continue working there. However, working with the TEX distribution directly on your computer is significantly faster and more convenient.

**Recommended:** Something to consider is the time limit of Overleaf TEX processing. Free Overleaf registration gives you just twenty seconds, all paid subscriptions 4 minutes. While twenty seconds could be quite OK if you only work on one particular book, it can turn out that even 4 minutes might be not enough for the entire Study Bible when it is completed. This is just an opinion, not tested and verified fact, though. What we know for sure, is that one minute (the free time limit before October 6, 2023) is not sufficient even for a plain Bible with no notes at all. So if you insist on working on Overleaf (for example, when you often work from different locations), before compiling the entire Bible, it may be wise to download the project in your computer and finish it from there.

# 3 Running TEX

If you have a TEX distribution installed on your machine (for example, TEXlive 2022 or newer) and if you have a command line available on your system, then you can run TEX using:

```
optex main.tex
```

where `main.tex` is the name of the main file (it can have a different name). The `optex` command runs LuaTEX with the OpTEX macros. The result of the processing is a file `main.pdf` and the processing message is in the `log` file `main.log`.

You can test if this works for you (without OpBible macros for now). Create a simple `main.tex` file in a text editor with this content:

```
\fontfam[lm]
Hello world!
\bye
```

and run `optex main.tex`. View the resulting PDF file `main.pdf` with a PDF viewer.

# 4 Structure of files processed by TEX

## 4.1 Main file with information about all other files

The main file is the file that is submitted to TEX first. For example, it is listed on the command line to start TEX. It contains information about what other files to be read by TEX. Finally, TEX will create a PDF file of the same name as the main file name but with the extension `.pdf`.

The main file for running OpBible (for example, `main.tex` or `bible-en.tex`) might look something like this:

```
\load[opbible] %  the OpBible macros
\enlang % initializing English hyphenation patterns

% Translation variants:
\def\tmark {BBE}      %  Bible in Basic English
%\def\tmark {Jubilee2000} % Jubilee 2000
%\def\tmark {NETfree} % New English Tranlation
%\def\tmark {UKJV}    % Updated King James Version
%\def\tmark {RNKJV}   % Restored Names King James Version
%\def\tmark {Webster} % Webster Bible

\input {vars.tex}  % Phrase declarations for different translation options
\input {books.tex} % Book titles and bookmarks \amark

\def\txsfile {sources/\tmark-\amark.txs} % Location of txs files
\def\fmtfile {formats/fmt-\tmark-\amark.tex} % Location of fmt files
\def\notesfile {notes/notes-\amark.tex} % Location of notes files
\def\introfile {others/intro-\amark.tex} % Location of book introduction files
\def\articlefile {others/articles-\amark.tex} % Location of article files
```

```
\def\printedbooks {%
 Gen Exod Lev Num Deut Josh Judg Ruth 1Sam 2Sam 1Kgs 2Kgs 1Chr 2Chr Ezra Neh Esth Job Ps
 Prov Eccl Song Isa Jer Lam Ezek Dan Hos Joel Amos Obad Jonah Mic Nah Hab Zeph Hag Zech Mal
 Matt Mark Luke John Acts Rom 1Cor 2Cor Gal Eph Phil Col 1Thess 2Thess 1Tim 2Tim Titus
 Phlm Heb Jas 1Pet 2Pet 1John 2John 3John Jude Rev
}

\processbooks % Generates document with books declared in \printedbooks
\bye
```

Now, let's see what each of these lines does and which ones will require modification on your part for the specific needs of your project.

Using `\load[opbible]`, the TEX loads macros of the OpBible package. This set of macros is the most important program that takes care of the typesetting.

The `\enlang` command sets the English word division patterns, so it assumes English text. The `en` is an ISO language abbreviation; you can use other languages: `\cslang` for Czech, `\delang` for German, `\eslang` for Spanish etc. All available language options are listed in the OpTEX documentation.

The command `\def\tmark {`⟨*mark*⟩`}` defines the macro `\tmark` as a mark of the translation used *Translation mark*. The marks of all available translations are listed in the file `vars.tex`. One of them should be selected as the mark of the currently processed translation. For example, `BBE` is the mark for the Bible in Basic English. In the example, six options for defining a translation mark in the case of English Bibles are given. Only one option (the one actually selected) does not have the % comment character in front of it.

If you are currently working on a book in the `BBE` translation, leave the `main.tex` file in the above form. When you want to switch to, say, the UKJV, you will use the percent sign to comment out (i.e., make invisible to TEX) the line with the BBE, but make visible (uncomment) the line with the UKJV. Then the `\tmark` definition section will look like this:

```
% Translation variants:
%\def\tmark {BBE}      %  Bible in Basic English
%\def\tmark {Jubilee2000} % Jubilee 2000
%\def\tmark {NETfree} % New English Tranlation
\def\tmark {UKJV}    % Updated King James Version
%\def\tmark {RNKJV}   % Restored Names King James Version
%\def\tmark {Webster} % Webster Bible
```

One of the translations must always be active, in other words, `\tmark` has to be defined. If you forget to put a percent sign before the line you want to comment out, the world won't fall apart; the very last definition that TEX loads will apply, which will redefine any previous ones.

The command `\input {vars.tex}` reads the configuration concerning the translation variants from the `vars.tex` file. See section 4.2 for details. Do not touch this line, even though you will probably be editing the `vars.tex` file called by this line.

The `\input {books.tex}` reads the information about the marks (abbreviations) of the books of the Bible and the book names are assigned here. This information is discussed in more detail in section 4.3.

The macro `\txsfile` (defined by `\def`) specifies the location of `.txs` files in the directory structure. For each book of the Bible, there has to be one `.txs` file containing the core text for that book. File names vary by book mark, and if there are multiple translations, the file name also includes the translation mark. In the `\txsfile` macro, you can use `\tmark` as the translation mark and `\amark` or `\bmark` as the book mark. For book marks, see section 4.3; for the format of `.txs` files, see the discussion in the section 4.4. In the example shown above, `.txs` files are located in the `sources/` directory and are named ⟨*translation-mark*⟩`-`⟨*book-mark*⟩`.txs`, so for example `BBE-Gen.txs`.

The English translations mentioned above are ready to go, you don't have to create them for yourself. If you need some other existing translation, you need to get it into a format usable for OpBible, in the same form as the `*.txs` files in the `sources/` directory. The `maketxs` script (see 4.4) will help you prepare individual `.txs` books from an existing source.

If you are going to create a brand new translation and plan to use it with OpBible, it probably wouldn't hurt to compose files one by one for each book directly in the desired format, or even better:

create the file `your-translation.out` for the entire Bible, and run `maketxs your-translation.out` to create all the `.txs` files needed; see also 4.4.

The `\fmtfile` macro defines the location of the files specifying the formatting of the core text. Each book of the Bible of each translation uses its own formatting file. This is something that cannot be common to all translations (unlike notes), because the paragraph breaks and added headings can (and do) differ for each translation. Our intent was so-called non-destructive editing, in other words, formatting the biblical text without interfering with the `.txs` files. The formatting files are discussed in section 4.5.

The macro `\notesfile` defines the location of the notes files. That's where you will write your commenting notes printed with the core text.

Each book of the Bible has its own notes file. The notes refer to a place in the core text, and it is the job of TEX to create pages with both the core text and the notes commenting on it. For more details on how to write notes files, see section 5.1. Notice that the note files are common to all translations, i.e., there are no separate files distinguished by `\tmark`. The note writing rules allow for the possibility to have various translations over one common notation, so long as they are in one language (e.g., English; see section 6). If you want to write notes for a completely different language, it is the best to start a new project (preferably in another main directory) with different `.txs` files, different formatting and note files.

The macro `\introfile` specifies the files where the introductions to each book are written. So, it is possible (and advisable) to create an individual introduction text for each Bible book.

The macro `\articlefile` specifies the names of files which contain the theological articles. The articles and similar stuff can be placed practically anywhere in the Bible, on any page interleaved with the core text but not within an introduction text.

The macro `\printedbooks` contains the marks of the books you want to process by TEX. The example above calls to process of the entire Bible, i.e., all 66 books of the Protestant canon are printed. If you're only doing test prints, for example, you can process only some of the books of the Bible and have an alternative definition in the main file, for example `\def\printedbooks{Dan}`. Just put it after the definition for the entire Bible, because later definitions of the same macro take precedence over any earlier ones.

The `\processbooks` command starts processing all the books specified in the macro `\printedbooks`. For each book, TEX will read the corresponding core text from `.txs` file, formats it using the data from the appropriate formatting file and appends the notes from the appropriate note file. The introduction text for books and the articles are added too.

The `\bye` command will terminate TEX. Anything you type after this farewell to TEX will be ignored.

You can also add your own macros and settings to the main file before `\processbooks`, which will affect the entire typesetting.

For example, you can put a `\ChapterPre{`⟨*code*⟩`}` or `\ChapterPost{`⟨*code*⟩`}` declaration. These codes are then executed before and after each chapter.

## 4.2 File declaring translation variants

If we are working with a single translation variant, there is no need to create this file and use it. Then just remove (comment out) the instruction to read it from the main file.

In the example in section 4.1, the file `vars.tex` is read, which should contain the declaration of translation variant marks using `\variants`:

`\variants` ⟨*number-of-variants*⟩ `{`⟨*mark*⟩`}` `{`⟨*mark*⟩`}` ... `{`⟨*mark*⟩`}`

where ⟨*number-of-variants*⟩ is the number of translation variants (into one and the same language, e.g., English), and then all the marks of the variant translations are listed. For example,

```
\variants 6 {BBE} {Jubilee2000} {NETfree} {UKJV} {RNKJV} {Webster}
```

declares the abbreviations for the 6 English translation variants: BBE for Bible in Basic English, Jubilee2000 for Jubilee 2000, NETfree for New English Tranlation, UKJV for Updated King James Version, RNKJV for Restored Names King James Version, and Webster for the Webster Bible.

The translation variants thus defined must match the definitions of `\tmark` in the main `main.tex` file, including upper or lower case.

Consider in advance the number of translations you want to use (changing their number later will be very difficult, though not impossible) and especially the order of the translations: the same

order in which they are declared in the definition of \variants will apply to the entire project. In all the notes commenting on a phrase that spells differently in different translations, you will be listing the different phrase versions in that precise order.

If you know that a phrase or word will appear more often than just in a single note, you can define it directly in the vars.tex file using the \vdef command. The number of phrases listed after \vdef must be exactly the same as the number of ⟨*number-of-variants*⟩, each of them enclosed in brackets, and they must correspond to the translation variants in the same order as the variants listed in the declaration of \variants. For example:

```
\vdef {Greece} {Greece} {Greece} {Grecia} {Grecia} {Grecia}
```

declares that the name of Greece is transcribed differently in different variants of translation: it is *Greece* in BBE, Jubilee2000, and NETfree, but changes to Grecia in UKJV, RNKJV, and Webster.

When we write notes concerning this country in the notes file, we will just write \x/Greece/ (the first translation variant in the definition of \variants) and this will be turned into the corresponding phrase used in the currently processed translation that we have declared in the main file using \def\tmark{...}.

So after changing \def\tmark{...} in the main file, all occurrences of of \x/Greece/ in the text of notes will automatically start behaving differently and adapt to the phraseology of that particular translation variant. Then such words can be inflected or added various endings, for example: The entry \x/Greece/'s will yield the form Grecia's in the note under the UKJV, RNKJV, and Webster translations but will remain Greece's with BBE, Jubilee2000, and NETfree. This is discussed in more detail in section 6.

The command \variants declaring the abbreviations of the translation variants is unique (the only one) for the variants file, whereas there can be more \vdef commands defining variant phrases in the file, because there are of course many phrases that are used in different translation variants, not just the country of Greece.

The whole passages of text can be handled differently depending on the translation variant set. The branching command \switch is used for this purpose. It is discussed in more detail in the section 6.3. For example, the names of individual translations (which are then used in the page header) can be declared differently for different translations using \def\bibname:

```
\switch {BBE}{\def\bibname{Bible in Basic English}}%
        {Jubilee2000}{\def\bibname{Jubilee 2000}}%
        {NETfree}{\def\bibname{New English Tranlation}}%
        {UKJV}{\def\bibname{Updated King James Version}}%
        {RNKJV}{\def\bibname{Restored Names King James Version}}%
        {Webster}{\def\bibname{Webster Bible}}
```

This particular declaration is a part of the already prepared vars.tex file.

## 4.3 Books names file

In the main file, there is an instruction to read the books names file. The example above from section 4.1 uses \input {books.tex}. That file must contain the commands \BookTitle in the format:

\BookTitle ⟨*a-mark*⟩ ⟨*b-mark*⟩ {⟨*non-abbreviated book title*⟩}

There must be at least one space between the marks and the book title. The beginning of a file books.tex might look like this:

```
\BookTitle Gn Gen {The First Book of Moses (Genesis)}
\BookTitle Ex Exod {The Second Book of Moses (Exodus)}
\BookTitle Lev Lev {The Third Book of Moses (Levicitus)}
\BookTitle Nu Num {The Fourth Book of Moses (Numeri)}
\BookTitle Dt Deut {The Fifth Book of Moses (Deuteronomy)}
\BookTitle Jos Josh {Joshua}
\BookTitle Jdg Judg {Judges}
...
```

In the first column after \BookTitle, there are ⟨*a-mark*⟩s, which are further used in the text of the notes and are used to create links to the Bible passages and other notes. The ⟨*a-mark*⟩s are visible in printed text when the references are used.

In the second column, there are ⟨*b-mark*⟩s, which can be the same as ⟨*a-mark*⟩s, but may also be different. It is possible, for example, that the names of the .txs files were created by exporting from some software and the bookmarks are different than we need to use in the text of our notes. Then it is possible that in the main file declare the location of the .txs files using \bmark instead of \amark, i.e.

```
\def\txsfile {sources/\tmark-\bmark.txs}
```

and have the files BBE-Gen.txs, BBE-Exod.txs, while in the text (including the references) we use the marks Gn, Ex, etc., not Gen, Exod. The ⟨*b-mark*⟩s are never printed in the text.

The macro \amark contains the ⟨*a-mark*⟩ of the currently processed book and the macro \bmark includes the ⟨*b-mark*⟩ of the currently processed book.

Note that the macro \printedbooks (in the file main.tex) with the marks of all the books we want to process (see section 4.1) contains ⟨*a-mark*⟩s, not ⟨*b-mark*⟩s.

In the third parameter after \BookTitle, the book names are in brackets.

File with 66 entries of \BookTitle is generated automatically after extracting the core texts from Sword download using a program mod2tex and a python script maketxs (see section 4.4). Feel free to use them, but ⟨*a-mark*⟩s and book titles will probably need to be manually modified according to the conventions of the translation, as demonstrated in the example above.

Additional information about individual books can be added to the book title file using the \BookException, \BookPre, \BookPost commands. They have the following syntax:

```
\BookException ⟨a-mark⟩ {⟨exception-text⟩}
\BookPre ⟨a-mark⟩ {⟨text-before-book⟩}
\BookPost ⟨a-mark⟩ {⟨text-after-book⟩}
```

The ⟨*exception-text*⟩ is inserted before reading all files of the book (.txs, format, notes, intro, article files) inside the cycle for reading all books with \processbooks. Then the ⟨*text-before-book*⟩ is inserted after the book files have been read but before the first verse (or introduction text) is processed. Finally, the ⟨*text-after-book*⟩ is inserted after the last verse of the book.

Example of using \BookException: Let's assume your language has diacritical marks that happen to be a part of the abbreviations of some Biblical books, and even though you cannot use diacritics in the file names, you still want to use them in your text to refer to those books as they appear in your Bible. What you can do is to modify the value of the macro \amark so that it does not contain diacritics as follows (following example applies for Czech language):

```
\BookException Ž {\def\amark{Z}}
\BookException Př {\def\amark{Pr}}
\BookException Pís {\def\amark{Pis}}
\BookException Ř {\def\amark{R}}
\BookException Žd {\def\amark{Zd}}
```

Then the notes-\amark.tex files, for example, are actually named notes-Z.tex, notes-Pr.tex etc. In the notes text you normally use the marks of usual book abbreviations, e.g., Ž, Př, Pís, etc.

There are five books in the Bible that have only one chapter (Obadiah, Philemon, 2 and 3 John and Jude). Because the references to them are not written with the chapter number (Phlm 1:4) but only with the verse number (Phlm 4), we have to teach TEX which ones they are so that it does not expect the chapter number but knows that it's the number 1 which in turn is not printed anywhere. When you refer to one of such books, the reference is being interpreted in a different way, see section 7.4. This is achieved by defining a macro \nochapbooks which must contain the ⟨*a-marks*⟩ of these books: \def\nochapbooks {Obad Phlm 2John 3John Jude}, obviously identical to those already listed in the \BookTitle definition (in the books.tex file), as well as those listed in \printedbooks definition in the main.tex file.

## 4.4 Core text format, called .txs files

The core text of the Bible is assumed to be stored in .txs files (text source). Each of the 66 books of the Bible is stored in its own .txs file. The names of the .txs files and their locations must match the \def\txsfile declaration in the main.tex file (see section 4.1).

Each line of the `.txs` file contains one Bible verse started with `#`⟨*chapter-num*⟩`:`⟨*verse-num*⟩. The verses must be listed in the correct order. For example, the beginning of the `BBE-Dan.txs` file looks like this (parts of the text are omitted in the sample):

```
#1:1 In the third year of the rule of Jehoiakim, king of Judah ... with his forces.
#1:2 And the Lord gave into his hands Jehoiakim, king of Judah ... store-house of his god.
...
```

The core texts of the Bible can be obtained, for example, from the Sword modules https://www.crosswire.org/sword/modules/ModDisp.jsp?modType=Bibles. Individual `.txs` files can then be generated using the following procedure (on Linux): Unzip the ZIP downloaded from the web page above (the so-called module). You need have the `libsword-dev` package installed on your computer and the program `mod2tex`, which is part of OpBible. Use `installmgr -l` to find out list of downloaded modules. If your current directory is the location where you have unzipped the ZIPs and where the `modules` directory was created, then the modules will be found. The modules contain text in binary format and we need to convert them into text format. To do this, just type the following into the command line:

```
mod2tex module > file
```

where `module` is the name of the module. In the resulting file you have the complete core text of the translation (module). For example, after

```
mod2tex KJVA > KJVA.out
```

the complete translation of the King James Bible (including Apocrypha) is in the file `KJVA.out`. This can now be split into `.txs` files with the command

```
maketxs KJVA.out
```

This command will create the `KJVA-books.tex` file in addition to the 66 `.txs` files, in which contains the titles and abbreviations of each book, so there is:

```
\BookTitle Gen Gen {Genesis}
\BookTitle Exod Exod {Exodus}
\BookTitle Lev Lev {Leviticus}
\BookTitle Num Num  {Numbers}
\BookTitle Deut Deut {Deuteronomy}
\BookTitle Josh Josh {Joshua}
\BookTitle Judg Judg {Judges}
...
```

These titles are in English, as the Sword source does not use non-English module names. If your language is a non-English then it is necessary to manually edit this file and insert the titles of your local language instead of English ones. The abbreviations of the books are listed twice as well. The first one ⟨*a-mark*⟩ should be changed according to convention of abbreviations in your language's Bibles and the second column of abbreviations can stay in English as it already is (as these second abbreviations, the ⟨*b-marks*⟩, will be part of the `.txs` file names).

Then the declaration

```
\def\txsfile {sources/Eng\tmark-\bmark.txs}
```

in the main file will cause the created `.txs` files to be searched for in the `sources/` directory and their names are assumed `EngKJVA-Gen.txs`, `EngKJVA-Exod.txs`, `EngKJVA-Lev.txs`, etc.

If the rare case should arise that you were to compile the Bible from several different sources, say the Old Testament you wanted in Dr. Jan Hejčl's translation and the New Testament in František Žilka's translation, you would have to play a bit with the file names so that the resulting definition of `\tmark` would be the same for the whole Bible. The two `*.out` files (e.g., `HEJCL.out` and `ZILKA.out`)[2] would have to be merged into one and then named e.g., `CzeHecjlZilka.out` and only then execute `maketxs CzeHejclZilka.out`. Now you can have a `\def\tmark{HejclZilka}` definition in the main `main.tex` file and the resulting study Bible will have Hejčl's Old Testament and Žilka's New Testament.

[2] Their modules aren't on Sword; the `.out` files would have to be created by downloading them from e.g., `https://obohu.cz` and converting them to the desired form with some clever script (or manually?).

From now on, you no longer need to modify `.txs` files in any way. In the `sources/` directory, you can have a "data" store of all the core Bible texts for all used translation variants at one place. In the case of the six translation variants of the Protestant canon, you have $6 \times 66 = 396$ files here.

If you have something in the `.txs` files that you want to format differently, it is possible to use `\cnvtext{`⟨*what*⟩`}{`⟨*how*⟩`}` in the main file. TEX will look up all occurrences of ⟨*what*⟩ in each verse of `.txs` file and replace them with ⟨*how*⟩. For example, if you have sections of text in square brackets in the `.txs` files, i.e. `[something like this]` and you want to print them in italics, write in the main file:

```
\cnvtext{[}{\bgroup\it} \cnvtext{]}{\/\egroup}
```

It may turn out that the `.txs` file does not use the correct typographical quotation marks (i.e., in English "..."), but instead there are the programmer's quote marks `"..."`. Without interfering with the `.txs` file, this can be fixed by adding an instruction to the main file: `\quotationmarks{"}{"}`. This will automatically replace the programmer's quote marks in the `.txs` file with correct English typographical quotation marks. Similarly, you can declare the replacement of the programmer's quote marks with Czech or any other other quotation marks, for example by declaring `\quotationmarks{„}{"}`. The programmer's quote mark is then implicitly replaced by the opening typographical quotation mark (the first parameter in the declaration), but if it is followed by a space, end-of-verse, end-of-paragraph, period, or comma, it is is replaced by a closing typographical quotation mark (the second parameter).

## 4.5 Entries specifying the formatting of the core text in `fmt-*.tex` files

The core text in `.txs` files does not contain any formatting or additional information, such as chapter titles or where to end a paragraph or where to switch from block format to center-justification and back, or how to display poetry.

Since we don't want to interfere with the core text,[3] you need to declare this additional information and link it to the corresponding verses using the special commands `\fmtadd`, `\fmtpre`, `\fmtins`, `\fmtfont`, `\fmtkeep` and `\fmtrepl`. These commands are typically in the `fmt-*-*.tex` files, for example, `fmt-BBE-Dan.tex`. It is advisable to maintain these formatting files dependent both on the book (Daniel in the example above) and on the translation (the Bible in Basic English in the example). You can always start with one file for each book and create the files for other translations as copies of this default one. At the end of a day, however, it may turn out to be be necessary to modify the formatting instructions for different variants of the core text according to the translation used.

The syntax for using these commands is as follows:

`\fmtpre{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*fmt-command*⟩`}`
`\fmtadd{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*fmt-command*⟩`}`
`\fmtins{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*phrase*⟩`}{`⟨*fmt-command*⟩`}`
`\fmtfont{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*phrase*⟩`}{`⟨*fmt-command*⟩`}`
`\fmtrepl{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*phrase*⟩`}{`⟨*replaced phrase*⟩`}`
`\fmtkeep{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*phrase*⟩`}`

where ⟨*fmt-command*⟩ is the "command" given to TEX for formatting. For example, `\endgraf` marks the end of a paragraph; `\begcenter` opens a passage with centered text which must be somewhere later closed with `\endcenter`. Or `\chaptit{<text>}` inserts ⟨*text*⟩ as the chapter title, whereas `\schaptit{<text>}` inserts the pericope title elsewhere, other than before the first verse of the chapter, and makes space above and below that title.

The `\fmtpre` inserts ⟨*fmt-command*⟩ at the beginning of the specified verse (before the verse number which is printed in a form of a superscript). The `\fmtadd` inserts ⟨*fmt-command*⟩ at the end of the specified verse. Finally, `\fmtins` inserts ⟨*fmt-command*⟩ inside the verse after the first occurrence of the specified ⟨*phrase*⟩ that must exist exactly as given in the verse. Otherwise, TEX prints a warning message and inserts no ⟨*fmt-command*⟩ at all.

[3] Think of them as the Holy Scriptures, therefore "untouchable" texts. The only reason that might entitle you to interfere with the core text is the possibility that you might find an error in the source from the Sword (or wherever you've got it from), when compared against the hard copy. Then it is really better to fix such error on the spot; if so, do it in `*.out` file and recompile it with `maketxs` afterwards. Never rely too much on internet sources; always check them out. They can and do contain errors. For example, Sword modules are full of missing spaces, occasionally you'll find a wrong character there, or even a missing part of a verse, etc. See the file `errata-BKR.txt` on the project's homepage to learn how many such errors can one Sword module contain.

An example of how the `\fmt*` commands can be used can be seen in the file `fmt-BBE-Dan.tex`.

The command `\fmtfont{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*phrase*⟩`}{`⟨*font*⟩`}` is used to highlight the selected phrase with the selected font. For example, the `\fmtfont{1:7}{the name of}{\em}` in the book of Daniel will print the words "the name of" in italics (to indicate the word not in original language), because `\em` is an intelligent italics switch (it automatically adds the italics correction `\/` after the word, which you don't have to worry about, but which you would have to if you switched to italics with an ordinary `\it`). Any other font switch can be used instead of `\em`.

The command `\fmtrepl{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*phrase*⟩`}{`⟨*replaced phrase*⟩`}` replaces first occurrence of ⟨*phrase*⟩ by ⟨*replaced-phrase*⟩.

Occasionally (although rather rarely) you may run across the situation where something in the core text requires very specific handling. For example, the RNKJV translation attempts to display Hebrew names, especially God's name YHWH, in Hebrew alphabet, even in the midst of surrounding text in Latin alphabet. Hebrew language writes from right to left and you will have to manage to typeset Hebrew words correctly. In case you download the core text containing such Hebrew words (like RNKJV does), you can use the macro `\fmtrepl` which searches the occurrences of such Hebrew words and does what we want to print. Consult the file `fmt-RNKJV-Dan.tex`.

The command `\fmtkeep{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*phrase*⟩`}` makes first occurrence of the ⟨*phrase*⟩ unsearchable. So you can apply a next `\fmt*` command to the next occurrence of the ⟨*phrase*⟩. You will probably not need this too often; if you do, see the section 13.2.

In addition to `\begcenter` and `\endcenter`, it is also possible to use controlled indentation with `\ind`⟨*number*⟩ (as indent). At the point of insertion, the line is terminated and the next line begins indented by the ⟨*number*⟩ of paragraph indents. However, inserting such `\ind` commands via `\fmtins` or `\fmtpre` can be quite laborious and cluttered, yet the Bible is rife with poetic passages that require a lot of of differently indented lines. The command `\fmtpoetry` can be used for this purpose. We will first demonstrate its use in the example of the NETfree translation of Daniel 6:26–27:

all the land: "Peace and prosperity! [26] I have issued an edict that throughout all the dominion of my kingdom
people are to revere and fear the God of Daniel.

"For he is the living God;
he endures forever.
His kingdom will not be destroyed;
his authority is forever.
[27] He rescues and delivers
and performs signs and wonders
in the heavens and on the earth.
He has rescued Daniel
from the power of the lions!"

[28] So this Daniel prospered during the reign of Darius and the reign of Cyrus the Persian.

The poetic part of this sample was typeset this way:

```
\fmtpoetry{6:26}{God;// forever./ destroyed;//}
\fmtpoetry{6:27}{/ delivers// wonders/// earth./ Daniel// }
  \fmtins{6:26}{Daniel.}{\medskip\hglue-2mm}
  \fmtadd{6:27}{\bigskip}
```

`\fmtpoetry{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*formatting-data*⟩`}` determines the format of a particular verse. The ⟨*formatting-data*⟩ contain the word from the end of the line followed by one or more slashes. Number of of slashes indicates how many paragraph indents the next line after that word will be shifted to the right. ⟨*formatting-data*⟩ must necessarily end with one or more slashes and may (but do not have to) begin with one or more slashes. If it does, then the beginning of the line is indented by the appropriate number of paragraph indents, but the verse number is set off slightly to the left to the space of a paragraph indentation (see the explanation below in the **Recommended** paragraph).

If you want to insert extra vertical spaces when using `\fmtpoetry`, you can do that, but only after the `\fmtpoetry` command, as you can see in the example above where `\medskip` (half line space) is inserted. The rule is that when you use more formatting commands like `\fmtins` or `\fmtpre` in poetry, these multiple formatting instructions will eventually be executed in reverse order, so for example, for the verse 6:26 in the example above, the first `\medskip` is executed at the beginning of the poetic part of the verse 26, after the name Daniel, followed by a period: `\fmtins{6:26}{Daniel.}{\medskip\hglue-2mm}`.

**Recommended:** If the poetry line starts with the left quotation marks, it is a typographical convention to let them sick out to the left of the text block so that only the letters are aligned, and the empty space bellow the quotation marks does not disturb that alignment. (The same reason applies for the verse numbers but you don't have to worry about these, as they are shifted automatically.) Therefore, you can add some negative horizontal space in order to stick the quotation marks out of the poetry block. In the example above, we used `\hglue-2mm`.

# 5 Notes and other objects linked to the core text

Notes in `notes-*.tex` files (located according to the `\notesfile` declaration in the main file) contain, among other things, notes on individual verses or parts of verses. The command `\Note` will be the most frequent command in the `notes-*.tex` files, designed to type in the particular notes. Since the commentary apparatus is the main reason why the the OpBible package was created, the entire section 5.1 is devoted to it.

## 5.1 Notes linked to core text phrases: The `\Note` command

The main purpose of the OpBible tool is to create a PDF from the core text not only with the Bible text itself, but with notes linked to their verses. In order to achieve that we use the `notes-*.tex` files (for example, `notes-Gen.tex` for the book of Genesis), which contain notes on individual phrases of the core text according to the following format. Each individual note is prefixed with the command `\Note` in this required form:

`\Note` ⟨*chapter-number*⟩`:`⟨*verse-number*⟩ `{`⟨*phrase*⟩`}`
⟨*note-text*⟩
⟨*blank line*⟩

For example:

```
\Note 1:2 {the temple of his god}
Marduk was the chief god of the Babylonian pantheon (cf. ⟨Jer 50:2⟩).
```

The example is from the `notes-Dan.tex` file, i.e., the notes file on the book of Daniel. More precisely, the note refers to chapter one, verse two, and the phrase "the temple of his god." This phrase, case-sensitive, must necessarily exist in the specified verse of the core text. Then TEX links it to the corresponding place of the core text, ensuring that the phrase and its commenting note occur on the same page. In other words, the pages do not break by verse number and its corresponding note number; they break by the commented phrase. When a page break occurs inside a verse between two phrases that are being commented on, the notes follow their phrases, not just the beginning of the verse. See an example of this in section 5.2.

If the ⟨*phrase*⟩ is not present exactly in the specified verse of the core text, TEX reports a warning to the log and to the terminal during processing and matches the note with the given verse as if the ⟨*phrase*⟩ were at the beginning of the verse. You will use the notes without specified ⟨*phrase*⟩ for summarizing comment on a larger portion of the text; see the first two notes in the sample book of Daniel.

However, if two consecutive notes should both be without a specified ⟨*phrase*⟩, they will be printed in the reverse order to the order in which they appear in the `notes-*.tex` file. In the sample book of Daniel you'll see the note on 1:1-21 to go *before* the note on 1:1-6:28. In the `notes-Dan.tex` you'll find `\Note 1:1-21` typed *after* `\Note 1:1-6:28`. If you want to change the order of the notes in such rare cases, simply switch their order in the `notes-*.tex` file.

The printed note (depending on the typographic design) then contains the repeated chapter and verse numbers, followed by the ⟨*phrase*⟩ that is being commented on (for example, in bold) followed by the actual text of the note.

Sometimes it is necessary to search for a slightly different phrase in the core text than what we want to have in the printed note (e.g., the word in the core text is in a different case, or it is a slightly differently worded phrase). Then it is possible to type an equal sign immediately after the `{`⟨*phrase*⟩`}`, which in turn is followed by the `{`⟨*phrase-to-be-printed*⟩`}`, i.e.,

`\Note` ⟨*chapter-number*⟩`:`⟨*verse-number*⟩ `{`⟨*phrase*⟩`}={`⟨*phrase-to-be-printed*⟩`}`
⟨*note-text*⟩
⟨*blank line*⟩

In this case, ⟨*phrase*⟩ is searched for in the core text, but in the actual note, ⟨*phrase-to-be-printed*⟩ is printed as the note entry. For example:

```
\Note 2:32-33 {head}={The head ... gold, its chest and arms ... silver, its belly
   and thighs ... bronze, its legs ... iron, its feet partly ... iron and partly ... clay}
Moving from the head to the feet of the image, there is a decrease not only in the  weight
of the materials but also in its value. The image was clearly too heavy with fragile feet.
It is an illustration of the fate of all human kingdoms and civilizations:  at the end,
each of them will collapse by its own weight.
```

In this example, the phrase "head" is being searched for but the text "The head ... gold, its chest and arms ... silver, its belly and thighs ... bronze, its legs ... iron, its feet partly ... iron and partly ... clay" will be printed in bold font as the first part of the note. In this particular example, the note (or at least its beginning) will occur on the page where the first occurrence of the word "head" in the verse 2:32 takes place.

It is possible to have more notes on the same Bible verse. Each of the notes, however, have to be separated by a blank line as always. In case that two or more consequent notes have identical ⟨*chapter-number*⟩:⟨*verse-number*⟩, these numbers are printed only in the first instance, not in following notes on the same verse.

The individual notes in the source notes file are separated by blank lines. This is necessary, otherwise TEX would not know where the text ends when reading them. It also increases the clarity of the source file.

If the note refers to the whole verse (without a specified phrase), write `{}`, i.e., an empty search phrase. For example:

```
\Note 1:1-21 {}={Keeping ritual purity}
The prophet introduces the context of his book by recounting a personal history
(of his and of his friends) in the captivity, education, loyalty to God and service
to King Nebuchadnezzar.
```

Moreover, the last two examples also demonstrate the possibility to give a range of verse numbers. The verse range will be printed in the beginning of the note as expected. If the ⟨*phrase*⟩ for the search is blank (as in this very last example), then the note will be placed on the same page as the beginning of the first verse in the specified range. If ⟨*phrase*⟩ is non-empty, it must occur in the first verse of the range of verses. The range symbol "–" is the only "minus" character normally available on keyboard. Its ASCII code is 45. It must not be any special character that somehow resembles a horizontal dash.

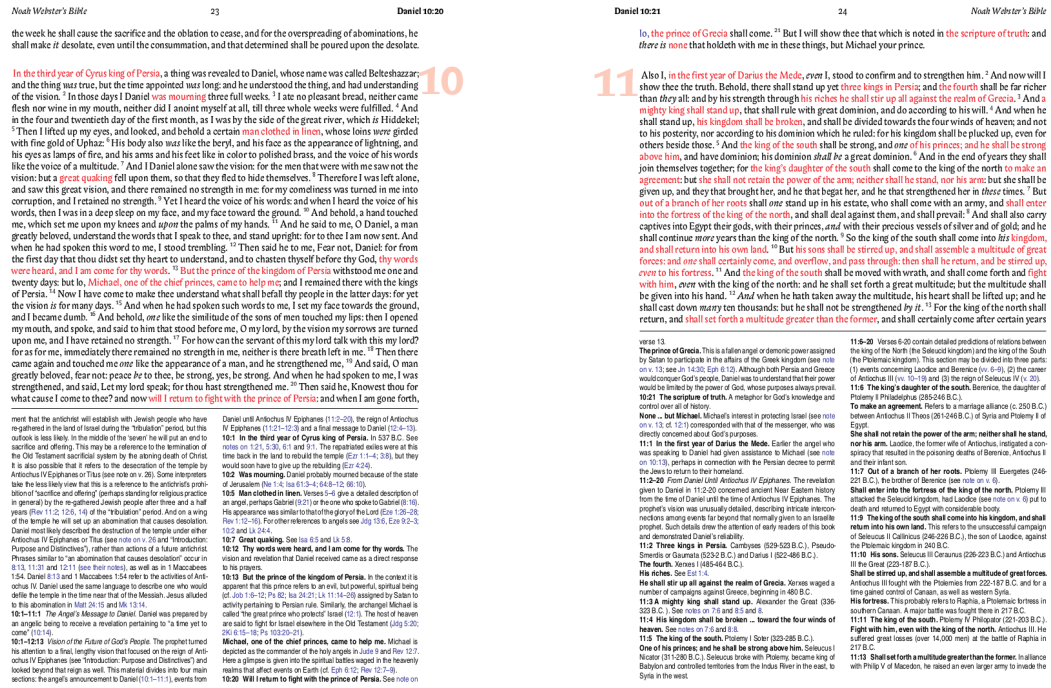
The order of printed notes on the same verse corresponds to the order of the phrases on which they comment in the core text. That means that the order in which they are written in the source file is irrelevant. Notes that link to an entire verse using the empty parameter `{}` are placed as the first, and if there are more than one of them, then their order is reverse to that in the source file, as stated above.

The phrases linked to the notes are printed in the `\notecolor` color. By default it is red, but you can say (for example) `\let\notecolor=\relax` in your main file, if you don't want to colorize these phrases.

## 5.2 Page break inside a verse

If different notes comment on different phrases within a single verse, and at the same time a page break after all the formatting happens to come between those phrases in the core text, then the notes follow their phrases to their respective pages. In other words, the page break is not determined by the verse number but by the phrase found in the core text and synchronized with the note.

In the sample Book of Daniel (of the Webster Bible translation), you can see this happen at the verse 10:20 (all the searched phrases are highlighted in red color). Look at page 23 where you read "will I return to fight with the prince of Persia" and then at page 24 where the same verse continues with the phrase "the prince of Grecia." The notes follow their respective phrases to correct pages. (You probably do understand that in a real hard copy Bible the page 23 would be the right one, as all odd pages are, and 24 would be its other side after you turn the page, don't you?)



## 5.3 Commands to insert other objects

In addition to the `\Note` commands, the `notes-*.tex` file can be used to write instructions for inserting additional objects related to a particular verse in the core text. Such an object is placed on the bottom of the page under the two columns of notes.[4] If the object is a quotation, it is placed on top of the page. We define the link to the location of the text using the ⟨*chapter-number*⟩:⟨*verse-number*⟩, similar to `\Note`. The object is placed on the same page where the corresponding verse begins. If the object does not fit on the page below the verse, it is placed at the bottom of the of the following page.

You can insert images using `\putImage` (section 5.4), articles using `\putArticle` (section 5.5), quotations using `\putCite` (section 5.6), images across two pages in an open book using `\putSpanImage` or `\putSpanText` (section 9.1).

The order in which you write the commands to insert these objects in the `notes-*.tex` file has no effect on the final appearance. You can have all objects of one type concentrated in one place in the source file, or you can have them placed between `\Notes`, typically by the verse number to which the objects are bound.

## 5.4 Inserting Images

To place images with a link to a specific verse, use the `\putImage` command:

```
\putImage ⟨chapter-number⟩:⟨verse-number⟩ {⟨caption⟩} [⟨label⟩] (⟨parameters⟩) {⟨file⟩}
```

For example:

```
\putImage 2:1 {Daniel's Visions} [daniels-visions] () {Nabuco.pdf}
```

- ⟨*chapter-number*⟩:⟨*verse-number*⟩ indicates the location where the image is placed.
- ⟨*caption*⟩ is the caption to the image. In the default typographic settings, it is printed in a strip that separates the upper part of the page from the the image itself.
- ⟨*label*⟩ specifies the image's reference label. You can then refer to it using `...see image \ref [daniels-visions] on page \pgref[daniels-vision]`, which, in the case of our example, prints ... see image Daniel's Visions on page 123. In doing so, the text in this link will be hyperlinked to the place where the image occurs. If you don't want to use ⟨*label*⟩, type `[]` (an empty label).

[4] According to the implicit page design; this can be redefined.

- ⟨*parameter*⟩ specifies optional additional instructions for formatting the image. They can be non-present, as the example suggests. By default, the image is stretched to the full width of the page. If you want it smaller, for example, type ⟨*parameters*⟩ in the space for `\picw=9cm` which will cause the image to be 9 cm wide and centered.
- ⟨*file*⟩ is the full name of the image file. It can have the extension `pdf` (for vector images) or `png` or `jpg` (for bitmap images). Image files must be stored in the `images/` directory. If they are located elsewhere, the parameter `\picdir` must be set for that location, for example `\picdir={BibleImages/}`. It is sensible to have it set in the main file, see section 4.1.

## 5.5 Inserting Articles

Articles with text (typically significantly longer than a note) can be placed on a page in a similar manner as images, i.e., in the default typographic setting at the bottom of the of the page on which the verse specified by ⟨*chapter-number*⟩:⟨*verse-number*⟩ happens to occur. If it does not fit there, it is inserted at the bottom of the next page. If it does not fit on a single page, its next part is inserted at the bottom of the next page (and so on until the whole article is complete).

Write the text of articles for each Biblical book in a file called by the macro `\articlefile`. For example, the file `articles-Gen.tex` contains all the articles for the book of Genesis. For how the content of such file has to look like, see below.

In the `notes-*.tex` file, you need to specify the article location request using `\putArticle`:

`\putArticle` ⟨*chapter-number*⟩:⟨*verse-number*⟩ {⟨*title*⟩} [⟨*article-number*⟩] (⟨*parameters*⟩)

For example:

```
\putArticle 6:1 {Who Was Darius the Mede?} [6] ()
```

- ⟨*chapter-number*⟩:⟨*verse-number*⟩ indicates the location to which the article will be placed.
- ⟨*title*⟩ is the title of the article. It is printed in a similar way as the caption of an image.
- ⟨*article-number*⟩ is a numeric designation of the article that must be unique within each single Biblical book. As articles are typically linked to chapters (we don't assume more than one article in a single chapter), then the ⟨*article-number*⟩ can (and should but does not necessarily have to) be the number of the corresponding chapter. The article is not placed following this number, however, the number is being used for references. For example, you can type `...see <"article" Da 6>a` and it will print ...see article Da 6. The format and properties of such links are described in section 7. In addition, the ⟨*article-number*⟩ is used to find the actual text of the article in the corresponding `atricles-*.tex` file. The
- ⟨*parameters*⟩ are optional parameters that specify the formatting of the article. If you are satisfied with the default settings (e.g., the width = `\hsize`) then leave the parentheses `()` empty.

The text of the articles to be inserted must be in the `articles-*.tex` file. This file must contain a line started by `\Article` as follows:

`\Article` [⟨*article-number*⟩] ⟨*text-of-the-article*⟩

Then it can continue with the next article given by another command `\Article` [⟨*article-number*⟩] followed by more text, etc. All the articles for a given book are thus grouped in a singe file.

If you specify the request `\putArticle` but the corresponding file `article-*.tex` or the corresponding line `\Article` [⟨*article-number*⟩] does not exist, TEX will end with an error.

## 5.6 Inserting quotations at the top of the page

The command `\putCite` ⟨*chapter-number*⟩:⟨*chapter-number*⟩ {⟨*text*⟩}, by default, inserts ⟨*text*⟩ as a quotation at the beginning of page containing ⟨*chapter-number*⟩:⟨*verse-number*⟩.

The ⟨*text*⟩ itself may contain a `\quotedby {<author>}` at the end. In such a case ⟨*author*⟩ is printed on a new line (if the quotation is placed on the right page) or, assuming there is enough space, it is shifted more to the right on the last line (if the quotations is placed on the left page).

For a controlled transition to a new line, you can use the `\nl` (new line) command in ⟨*text*⟩. For example:

```
\putCite 1:8 {You can't get second things by putting them first. \nl
              You get second things only by putting first things first.
\quotedby{C. S. Lewis}}
```

## 5.7 Inserting quotations in the margin of an article

Article text is formatted in two columns by default. It is possible to break the outer column and insert a quotation or something similar. Such text will stick out into the outer margin.

Inside the article (i.e., just after `\Article [`⟨*article-number*⟩`]`) you need to insert a declaration of the quoted text using: `\Cite` ⟨*letter*⟩ `{`⟨*text*⟩`}`. Here ⟨*letter*⟩ is typically `A`. But if you want to insert more than one quotations in a single article, you need to distinguish between them with additional letters, i.e., `B`, `C`, etc., and all quotations have to be written at the beginning of the article.

The `\Cite` command only declares the quote. Its actual insertion into the text is done by `\insertCite` ⟨*letter*⟩`\left` and at the same time `\insertCite` ⟨*letter*⟩`\right` elsewhere in the text of the article. If the article is on the left-hand side of a double page, the quotation is placed only by `\insertCite`⟨*letter*⟩`\left`, in other words, `\insertCite`⟨*letter*⟩`\right` is ignored. If the article is on the right-hand side of an open double page, the placement is is governed only by `\insertCite` ⟨*letter*⟩`\right`. The quotation will appear right below the line in which the `\insertCite` is given. The line itself is is not split because of this (in other words, the paragraph is not terminated because of the insertion of the quotation).

The reason why it is necessary to give two locations for `\insertCite` is following: We don't know ahead on which page (odd or even) the article with the inserted quotation will appear. Since the quotation should stick out in the outer margin, it should be placed in the first column on the left page, and in the second column on the right page, that is somewhere slightly different. The location of `\insertCite` ⟨*letter*⟩`\left` should therefore correspond to a line in the first column and `\insertCite` ⟨*letter*⟩`\right` to another line in the second column of the article.

It is wise to debug (i.e., test in advance) what the placement of the quote looks like for both options that may occur (left/right). If you want to to see how it works for the variant that doesn't match the correct position of a page, you can use the command `\swapCites` at the beginning of the article text which will switch the left/right position of the quotation. But this should not be left on for the final printing, thus activated `\swapCites` will cause a warning printed to the terminal and to the log file.

# 6 Different (but similar) versions of the core text

See section 4.2 for an introduction to this issue. This is detailed documentation for each option.

## 6.1 Declaring translation variants and using the `\x` command

If the variants are not declared with the `\vdef` command, then `\x/`⟨*phrase*⟩`/` command used in the text will output the same ⟨*phrase*⟩. However, it is possible to declare translation variants. The number of variants must be specified with the `\variants` command (see section 4.2). This is done just once for the whole document (the entire Bible). Then the `\vdef` commands can follow, always with exactly as many parameters and in the same order as the number of variants specified by the `\variants` command. For example:

```
\variants 6 {BBE} {Jubilee2000} {NETfree} {UKJV} {RNKJV} {Webster}

\vdef {took the kingdom} % BBE
      {possessed the kingdom} % Jubilee2000
      {take possession of the kingdom} % NETfree
      {possessed the kingdom} % UKJV
      {possessed the kingdom} % RNKJV
      {possessed the kingdom} % Webster

\vdef {He-goat} % BBE
      {Goat} % Jubilee2000
      {Male goat} % NETfree
      {He goat} % UKJV
      {Goat} % RNKJV
      {He-goat} % Webster
```

If `\def\tmark {`⟨*variant*⟩`}` is now declared in the main file, then the `\x/`⟨*phrase*⟩`/` will turn into the ⟨*phrase*⟩ of the specified ⟨*variant*⟩. In doing so, the ⟨*phrase*⟩ parameter of the `\x` command must

be identical to the first phrase specified in the \vdef command (in the examples above, it's BBE's took the kingdom and He-goat). If, in our case, the main file says \def\tmark{BBE}, then

\x/took the kingdom/ yields: took the kingdom
\x/He-goat/ gives: He-goat
\x/Whatever/ prints: Whatever

But if there is a \def\tmark{Jubilee2000} in the main file, then

\x/took the kingdom/ yields: possessed the kingdom
\x/He-goat/ gives: Goat
\x/Whatever/ will print Whatever and the terminal will warn that the phrase \x/Whatever/ has no declared translation in \vdef.

Selected parameters of \vdef may be empty (written as {}), indicating an undefined phrase for that translation. If such undefined phrase needs to be used with \x/.../, a warning message will be printed in the .log file.

In addition, the parameter may contain a single " character, indicating that the the same phrase is used as in the previous parameter. So our example above might also look like this:

```
\vdef {took the kingdom} % BBE
      {possessed the kingdom} % Jubilee2000
      {take possession of the kingdom} % NETfree
      {possessed the kingdom} % UKJV
      {"} % RNKJV
      {"} % Webster
```

## 6.2 Variant phrase declarations for matching notes with text

The \Note command may be immediately preceded by a declaration of the search phrase by variant translations using \ww (this is an abbreviation for watchword). The \ww command has as many parameters as there are translation variants declared by \variants command, and these parameters can be simple (in the format {⟨*wanted-phrase*⟩}) or compound (in the format {⟨*search-phrase*⟩}={⟨*what-to-print*⟩}). The immediately following \Note will then ignore its parameter for the search phrase and use the parameter from \ww corresponding to the translation variant being processed. For example:

```
\ww {if you do not make clear to me the dream and the sense of it} % BBE
    {if ye will not make known unto me the dream with its interpretation} % Jubilee2000
    {If you do not inform me of both the dream and its interpretation} % NETfree
    {if all of you will not make known unto me the dream, with the interpretation thereof} % UKJV
    {if ye will not make known unto me the dream, with the interpretation thereof} % RNKJV
    {if ye will not make known to me the dream, with the interpretation of it} % Webster
\Note 2:5 {If you do not tell me what my dream was and interpret it.}
    Nebuchadnezzar formulated a plan for testing his advisors. If they could not relate the dream back
    to him he would have no confidence in their interpretation (see <"v." 9>).
```

The Note 2:5 given here searches the text in verse 2:5 for "if you do not make clear to me the dream and the sense of it" when a variant translation of the BBE is being processed, and looks up the text "if ye will not make known unto me the dream with its interpretation" when the Jubilee2000 translation variant is being processed. The example assumes that six \variants have been declared using the \variants command for translation variants in the specified order.

Whenever \ww preceeds a \Note, the brackets for a {⟨*phrase*⟩} after the \Note can contain anything and it will not be displayed, as it will be replaced by the content of \ww. However, they always have to be present, even if empty.

You can also specify a different phrase for searching and for printing in a note, as shown in the following example:

```
\ww {head}={head ... gold, ... breast and arms ... silver, ... middle and sides ... brass, legs ...
        iron, ... feet in part of iron and in part of potter's earth} % BBE
    {head}={head ... gold, ... breast and arms ... silver, ... belly and thighs... brass, legs ...
        iron, ... feet  part of iron and part of baked clay} % Jubilee2000
    {head}={head ... gold, ... breast and arms ... silver, ... belly and thighs... brass, legs ...
        iron, ... feet  part of iron and part of clay} % NETfree
```

```
    {head}={head ... gold, ... breast and arms ... silver, ... belly and thighs... brass, legs ...
        iron, ... feet  part of iron and part of clay} % UKJV
    {head}={head ... gold, ... breast and arms ... silver, ... belly and thighs... brass, legs ...
        iron, ... feet  part of iron and part of clay} % RNKJV
    {head}={head ... gold, ... breast and arms ... silver, ... belly and thighs... brass, legs ...
        iron, ... feet  part of iron and part of clay} % Webster
\Note 2:32-33 {} Moving from the head to the feet of the image, there is a decrease not only in the
    weight of the materials but also in its value. The image was clearly too heavy with fragile feet.
    It is an illustration of the fate of all human kingdoms and civilizations:  at the end, each
    of them will collapse by its own weight.
```

The \ww also accepts the parameters in the form {"} like the \vdef does. The value of previous parameter is used instead of {"}.

The search and replace phrases are used exactly as they are written in parameters of the \ww command. This does not apply to \Note notes, which are not preceeded by \ww. Then when using

\Note ⟨*chapter:verse*⟩ {⟨*search-phrase*⟩} ⟨*text*⟩ ⟨*blank line*⟩

or

\Note ⟨*chapter:verse*⟩ {⟨*search-phrase*⟩}={⟨*what-to-print*⟩} ⟨*text*⟩ ⟨*blank line*⟩

the ⟨*search-phrase*⟩ is first transformed by the data from \vdef. Only if such data do not exist for the search phrase, ⟨*search-phrase*⟩ is being used as it is.

**What to watch out for:** Keep in mind that formatting a phrase (see 4.5) prevents it from being found, as the formatting sticks its own commands to it. So if your note is commenting on a phrase that already underwent some formatting (e.g., a line break within a poetry passage, or a font change, etc.) pick the closest word that is not influenced by this formatting as the search phrase, and use the complete phrase that you want to display as the replacement (⟨*what-to-print*⟩) one.

For example, you want to comment on the phrase *How great are his signs!* in Daniel 4:3. Writing directly \Note 4:3 {How great are his signs!} would yield a warning message and the phrase would not be found. That is because in some translations (e.g., NETfree, see the file fmt-NETfree-Dan.tex), the word *sign!* at the end of the sentence is part of formatting, while in some others (e.g., Jubilee2000 or Webster) the core text contains extra characters [] around some words that are being converted into font change command using \cnvtext (see 4.4) .

Therefore, you have to work around by using replacement (⟨*what-to-print*⟩) phrase:

```
\Note 4:3 {How great}={How great are his signs!}
```

**Recommended:** For clarity, it is worthwhile to have each translation on a new line, and to label it after the commenting percentage sign, so that we know what belongs where without groping. The last line of the notes-*.tex file for a particular book that TEX loads should contain a single \endinput command. Whatever follows below this instruction on subsequent lines will not be seen by TEX. (But don't confuse it with \end or \bye so that TEX doesn't stop running at this point but continues reading other files.)

Then, below \endinput, you can have a few lines prepared, e.g., in this form:

```
\ww {}={} % BBE
    {}={} % Jubilee2000
    {}={} % NETfree
    {}={} % UKJV
    {}={} % RNKJV
    {}={} % Webster
\Note 1:1 {}={}
```

and then just copy these lines in place of the new note, edit the chapter and verse number after \Note, or delete ={} where they are not needed. This way you don't lose track of which phrases belong where, whether you write them out manually or paste them from a Bible program or online resources.

## 6.3 Text processing branching by translation variants

You can use the \switch command to branch the text processing, depending on the set value of the \tmark parameter, i.e., depending on the translation variant currently being processed. The command has the following syntax:

```
\switch {⟨list of variants⟩} {⟨what to do⟩}%
        {⟨list of variants⟩} {⟨what to do⟩}% ... etc.
        {⟨list of variants⟩} {⟨what to do⟩}
```

The pairs {⟨*list of variants*⟩} {⟨*what to do*⟩} can be given as many times as you like. After each pair {⟨*list of variants*⟩}{⟨*what to do*⟩} (except for the very last pair) must be followed immediately and without spaces by another such pair. Therefore, when moving to the next line, write a percent sign after the closing parenthesis to cover the space from the end of the line. Spaces at the beginning of the next line do not matter. You can understand the percent sign after the pair as "next pair continues".

The ⟨*list of variants*⟩ is a single translation variant mark or a list of translation variant marks separated by a comma and no spaces. TEX then works as follows: If a variant defined by the \tmark parameter occurs in the ⟨*variant list*⟩, then the following ⟨*what to do*⟩ is executed. If there is no such variant, the following ⟨*what to do*⟩ is skipped. Example:

```
\switch {BBE}         {water-door of the Ulai}% BBE
        {NETfree}     {Ulai Canal}%             NETfree
        {Webster}     {river Ulai}%             Webster
        {Jubilee2000,UKJV,RNKJV} {river of Ulai} % UKJV, RNKJV, Jubilee2000
```

The example shows how to print the name of Ulai river, depending on the translation variant being processed.

Once TEX finds a match and does ⟨*what to do*⟩, then the possible following entries within the same \switch command are skipped. Furthermore, the rule is that if ⟨*list of variants*⟩ is empty, then ⟨*what to do*⟩ is always executed, if not skipped according to the previous rule. So an empty ⟨*list of variants*⟩ at the end of the \switch parameter pairs is evaluated as "all other cases". The example above can also be written as follows:

```
\switch {BBE}         {water-door of the Ulai}% BBE
        {NETfree}     {Ulai Canal}%             NETfree
        {Webster}     {river Ulai}%             Webster
        {}            {river of Ulai}           % UKJV, RNKJV, Jubilee2000
```

The \switch command can be used not only for single phrases within notes \Note, but also to entire sections of input text containing, for example, \Note, several \Note notes, several definitions, etc.

The \switch command cannot be used in the parameters of other macros. On the other hand, the command \x/⟨*phrase*⟩/ works inside \switch.

## 6.4 Renumbering verses according to translation variants

Some translation variants have different verse numbering. In such cases, the \renum command can be used as follows:

\renum ⟨*book-abbreviation*⟩ ⟨*default-chapter-number*⟩:⟨*default-verse-number*⟩ = ⟨*translation*⟩
⟨*chapter-number*⟩:⟨*verse-number-from*⟩-⟨*verse-number-to*⟩

where ⟨*translation*⟩ is the mark of a particular translation. If the \tmark is declared as ⟨*translation*⟩ using \def\tmark{⟨*translation*⟩} then the ⟨*default-chapter-number*⟩:⟨*default-verse-number*⟩ is replaced with ⟨*chapter-number*⟩:⟨*verse-number-from*⟩-⟨*verse-number-to*⟩. Such renumbering applies to the entire range of verses defined by ⟨*verse-number-from*⟩–⟨*verse-number-to*⟩.

For example, let's suppose that in the book of Daniel, in verses 10:20 and 10:21 we want to comment on several phrases, such as "angel of Persia" and "true writings". For some reason (not entirely clear) BBE has these phrases in opposing verses then the rest of the translations.

We number the notes according to the translation given as the first parameter of the definition of \variants in the vars.tex file, (in our example BBE) including references like "See note on 10:20."

Renumbered translations change the note number according to the actual verse number that the note comments on, including the reference, which is then printed as "See note on 10:21."

If the renumbering refers to a single verse only, then the identical ⟨*verse-number-from*⟩ and ⟨*verse-number-to*⟩ should be given, as in the example below:

```
\renum Dan 10:20 = Jubilee2000 10:21-21
\renum Dan 10:20 = NETfree 10:21-21
\renum Dan 10:20 = UKJV 10:21-21
\renum Dan 10:20 = RNKJV 10:21-21
\renum Dan 10:20 = Webster 10:21-21
```

The macro `\renum` can handle even such tidbits as shifting a number of a mere part of a verse. This example comes from the Czech SNC translation where the phrase „jen Bůh, který je v nebesích, odhaluje tajemství" ("there is a God in heaven who reveals secret things"), is for no clear reason a part of the verse 2:27 while all other Bibles have it in the verse 2:28. But the fun does not end here; if you want to comment also on the phrase „posledních dnů" ("the latter days") you'll now find it in the verse 2:28 where the rest of the Bibles have it as well. What are you going to do? How can you renumber just half of the verse?

Well, you can do it by applying `\renum` on the same verse twice: the first time *before* the `\Note` on the first phrase, and the second time *after* it. So you type:

```
\renum Da 2:28 = SNC 2:27-27
\ww {jest Bůh na nebi, kterýž zjevuje tajné věci}       %BKR
    {v nebesích je Bůh, zjevující tajemství}            %PSP
    {jest Bůh na nebesích, zjevující tajemství}         %CSP
    {je Bůh v nebesích, který odhaluje tajemství}       %CEP
    {Na nebi je však Bůh, který zjevuje tajemství}      %B21
    {jen Bůh, který je v nebesích, odhaluje tajemství}  %SNC
\Note 2:28 {} Podobně jako Josef v Egyptě (⟨Gn 40:8⟩; ⟨Gn 41:16⟩) si ani \x/Daniel/
nepřisvojuje poznání a výklad snu, nýbrž připisuje je Bohu.

\renum Da 2:28 = SNC 2:28-28
\ww {v potomních dnech}   %BKR
    {v posledku dní}      %PSP
    {v budoucích dnech}   %CSP
    {v posledních dnech}  %CEP
    {v posledních dnech}  %B21
    {posledních dnů}      %SNC
\Note 2:28 {} Dosl. \uv{v posledních dnech}, čemuž sz lidé rozuměli jako době obnovy národa
po návratu z exilu (viz ⟨Dt 4:30⟩). Tatáž fráze může označovat jakoukoliv budoucnost
(⟨Gn 49:1⟩). V NZ je použita celkem pětkrát, z čehož dvakrát odkazuje na věk, započatý
o Letnicích (⟨Sk 2:17⟩; ⟨Žd 1:2⟩), a třikrát na závěr dějin
před Kristovým druhým příchodem (⟨2Tm 3:1⟩; ⟨Jk 5:3⟩ a ⟨2Pt 3:3⟩).
```

Now everything works as it should: Where the numbering diverges, it renumbers; where it it matches, it stays the same. All Bibles except SNC have the first `\Note` numbered 2:28 but SNC has it as 2:27; then all of them have the second `\Note` as 2:28.

**Recommended:** You may want to use `\renum` also in other files, e.g., intro or fmt, not just notes. In such cases, if would be advisable to use `\renum` inside the `vars.tex` file, so that it applies everywhere without the need to retype the same instruction more times (for example, both in Introduction and Notes files).

# 7 Methods of creating hyperlinks

A reference is a part of the text by which the reader, even after printing it, can tell to what other place in the text (or in the internet) one can look at. Thus, it typically contains a numerical indication of page or the number of a chapter, section, etc. In addition, if the reader is working with a PDF viewer, then this part of the text can be *active*, i.e., when the mouse hovers over this text you can click on it and the PDF viewer will go to the specified place in the document (or to the Internet).

The Bible is invariably structured text. It contains (in the Protestant canon) 66 books with established markers for those books, each book has its chapters numbered from one and each chapter has verses numbered from one. So there is no need to have TEX generate these numbers automatically (as it does when typesetting, say, a technical text that is divided into chapters and sections), and thus there is no need to use labels in the source document (which TEX would assigns to the numbers generated during processing) and to refer by these labels in the source file, as described in section 1.4.3 of the OpTEX documentation. It is much more efficient to refer directly to a specific place in the Bible that already has for many centuries now, fixed chapter and verse numbers.[5]

The references to a specific place in the Bible are written between < and > in the source file. The text between these characters is printed as written (with the exceptions mentioned below). Nevertheless, TEX has to be able to interpret the reference correctly in order to make it active with a clicking capability to the correct place in the Bible. This is done by rather large amount of rules, therefore this section is dedicated to them.

## 7.1 Basic rule with complete data

The reference between < and > is of the form

```
<"text" data>
```

or just `<data>`. The complete ⟨*data*⟩ is of the form ⟨*book*⟩ ⟨*chapter*⟩:⟨*verse*⟩. Here ⟨*book*⟩ is the abbreviation of the book (it must be followed by space), ⟨*chapter*⟩ is the chapter number, and ⟨*verse*⟩ is the verse number. Example:

```
... see also verse <Jer 8:13>
... see also <"verse" Jer 8:13>
```

In the first case it prints ... see also verse Jer 8:13 and in the second case ... see also verse Jer 8:13. In both cases the area marked in blue will be active (clickable) and will lead to Jer 8:13.

## 7.2 Link specifier

The link's closing character > may be immediately (i.e., without a space) followed by a link specifier, which is one of the letters:

- `n` ... refers to a note,
- `a` ... refers to an article,
- `i` ... refers to an introduction,
- `g` ... refers to a gloss (which will hopefully come into play in one of the future versions of OpBible with the core text typeset in two columns).

The link specifier is not printed in PDF, it is just internal information where the active link should click. If the closing character > is not followed by any of these specifiers, it is a link to the verse (probably the most common case). Example of a note reference:

```
... see <"note on" Jer 7:4>n for more information.
```

prints ... see note on Jer 7:4 for more information. The click leads to the first note on Jer 7:4, not to the verse itself.

In the case of a reference to an article (specifier `a`), the full entry has the format ⟨*book*⟩ ⟨*chapter*⟩, i.e., the verse information is missing because the articles can be understood as introductions to chapters. In the case of a reference to the introduction to a book (specifier `i`), the full entry is in the format ⟨*book*⟩ (lacking both chapter and verse information) because these are book introductions. In other cases, the full entry has the format as described in section 7.1, with the exception described in section 7.4.

[5] The Archbishop of Canterbury, Stephen Langton, in the early 13th century, when he was teaching at the University of Paris (and has not yet become an archbishop) divided the Bible into chapters. Then, in the mid-16th century, the French books printer Robert Estienne divided the New Testament into verses and added the Old Testament, which Jewish scribes had centuries earlier divided into verses (but not into chapters). Since 1553, when Estienne published the first French Bible thus numbered, we have used this system to this day.

## 7.3 References to whole chapters

You can omit : and verse number in your reference text. Then the reference points to the first verse of the given chapter. Examples:

```
... see <"chapter" Dan 7>
... Joseph's story (<Gn 39-41>)
```

In the second example, the range of chapters is given and TEX creates an internal link to the first verse of chapter 39. Compare also with the section 7.6.

## 7.4 Exception for the format of the full entry for some books

Books Obad, Phlm, 2John, 3John, and Jude are not divided into chapters. In a reference to a verse, note to a verse or a gloss to one of these books, the chapter information is missing and the format of a complete reference looks like this: ⟨*book*⟩ ⟨*verse*⟩. In order to teach TEX to apply this exception, the list of abbreviations for these books needs to be defined in macro `\nochapbooks`. For example, the `books.tex` file says

```
\def\nochapbooks{Obad Phlm 2John 3John Jude}
```

Since there are different book abbreviations for different languages, this macro needs to be defined according to the language used. In any case, the abbreviations have to be exactly the same as the ones declared by the `\printedbooks` macro in the main file.

Example: `see <Phlm 3>` prints see Phlm 3 and it refers to verse 3 of the Philemon, not to chapter 3. On the other hand, `see <Gen 3>` prints see Gen 3 and refers to the beginning of the chapter 3 of Genesis. It is the reference to the whole chapter, as described in the section 7.3.

## 7.5 Incomplete data

Sometimes the reader can determine the location of a verse from the context, therefore ⟨*data*⟩ does not need to be complete. In the *incomplete data*, there may be a ⟨*book*⟩ missing, or ⟨*book*⟩ ⟨*chapter*⟩: or even everything. For example:

```
... we also see an analogy in <"verses" Jer 8:13>, <9:7> and <11:3>
... see verses <Jer 8:13>, <15>, <17>
... see all of the <"verses" Jr 8:13>--<22>,
... (cf. <Jr 8:13> and <"its note">n).
```

This yields: ... we also see an analogy in verses Jer 8:13, 9:7 and 11:3 ... see verses Jer 8:13, 15, 17 ... see all of the verses Jer 8:13–22. ... (cf. Jer 8:13 and its note).

The entries `9:7`, `11:3`, `15`, `17`, `22` and the last blank one in these examples are incomplete data. The reader knows that they refer to the book of Jeremiah and where no chapter is given, they refer to chapter 8 of Jeremiah. In the last example with a blank, the reader knows that it is a note on Jer 8:13. TEX knows that too, and it assigns correct links (to be clicked on) to the incomplete entries because incomplete entries take the unspecified information from the previous last entry. This rule applies locally to a single text object: note, article, introduction, etc. If the very first entry in a given text object is incomplete, the unspecified information is replaced with by the abbreviation of the book currently being processed, or by the number of the current chapter, or verse, respectively.

If an incomplete entry is prefixed with \, the unspecified information is taken from the book or chapter or verse currently being processed, regardless of which entry precedes. For example:

```
\Note 4.5 {} The idea is repeated in <Jer 4:5>. Also, compare to \<8:3>.
```

Here the reference 8:3 leads to verse 3 of chapter 8 of the actual current book. If the backslash sign \ were not there, then this reference would click to 8:3 of Jeremiah.

An incomplete entry is printed as is, as incomplete. The above rules only apply internally so that the active link will work properly after a mouse click.

## 7.6 Format for the verse range and for the section in the verse

In each entry, it is possible to have a range of verses in the format ⟨*from*⟩-⟨*to*⟩ instead of just ⟨*verse*⟩ or ⟨*chapter*⟩:⟨*verse*⟩. The TEX will create an internal link to the first verse of the range only and turns the hyphen sign (character -, ASCII 45) in the range into an en-dash.

Examples:

```
<Jer 8:3-7>,
<Jer 8:3-9:5>,
<3-7>,
<8:3-7>.
```

For example, the first link in this example prints as Jer 8:3–7 and offers a link only to Jer 8:3.

The same can be applied to the chapter range, as shown in the example in the section 7.3.

## 7.7 References to subverses

Sometimes we need to refer to a part of a verse, not the whole verse. This is done by appending a letter immediately after the verse number. For example,

```
... see <Dan 9:11b>
```

You can append such a letter to both complete and incomplete references. For the purpose of hyperlink, these letters are ignored but printed in PDF. Thus, the example given prints ... see Dan 9:11b, but the link leads to Dan 9:11.

## 7.8 Hidden data

When none of the rules are sufficient to create an internal link from a link listed above, you can enclose everything in the link to be printed in the programmer's quotation marks "..." and hide the subsequent entry to create the internal link. To do this, just append underscore _ followed by the data right after the closing programmers' quotation mark " Such entry is not printed. For example,

```
<"First Book of Samuel"_1Sam 1:1>
```

prints only the text First Book of Samuel, which internally refers (the click jumps) to the first verse of this book.

## 7.9 Renumbering the reference

If a link points to a verse that has a different number than the default translation (the first translation among the parameters of `\variants` declaration), then enter the reference according to the default numbering and TEX will recalculate it itself according to the data specified by `\renum`. It prints the recalculated data and uses it for the internal reference. Cf. 6.4.

## 7.10 Data reduction when printing

You may find it helpful to be able to write the complete data in parentheses for the reference and expect it to be automatically reduced to incomplete if they refer to the current book. If they do not refer to the current book, the entry will remain complete. You can do this by by adding a `\re` before the opening parenthesis of the entry (the eventual reduction rule applies only to this single entry) or by using the `\reduceref` command. If you you use it in a note (or inside a TEXgroup), the reduction rule is applied for all subsequent complete entries of that note (or of that TEX group). When used in a document declaration, the eventual reduction rule is applied to all complete data in the document.

Example:

```
\re<"verse" Dan 7:3>
```

is printed as verse Dan 7:3 if this reference is given outside the book Daniel. However, when this reference is given inside the book of Daniel, it is printed only as verse 7:3, which internally refers to Dan 7:3.

The reference reduction rule set by `\reduceref` can be turned off with the `\noreduceref` command. From there to the end of the note (TEX group) links behave as if `\reduceref` were not turned on.

## 7.11 The book's abbreviation (a-mark) can be printed differently

If the mark for a book is declared differently in different translations following `\vdef`, then use only the ⟨*book*⟩ entry in the references according to the first (default) variant. However, if you currently use an alternative translation via `\tmark`, the link will be printed according to the `\vdef` entry of

that alternative translation variant. Internally, nevertheless, references are linked according to the default variant.

You'll find this feature is useful when you run across two different translations which name the same book differently. In English it is rarely the case but let's suppose that (hypothetically) there exists an English translation that prefers "Paralipomenon" over "Chronicles". Let's call this hypothetical translation "Greek." Your default translation (the first translation in the `\variants` declaration) uses "Chronicles" as all English translations usually do.

Now you can declare `\vdef` for marks `1Chr` and `2Chr` (Chronicles) for most of English translations, especially the first one, and an alternative text for `1Par` and `2Par` (Paralipomenon) for the "Greek" translation. Then you can type the reference `<"see" 1Chr 2:3>`, you get see 1Chr 2:3 in the usual translation variants, but it will print see 1Par 2:3 if you use translation variant "Greek".

Non-English languages may use it more often, though. For example, in the Czech language, most translations use "Paralipomenon" where English Bibles use "Chronicles" but there is one translation (B21) that uses the title "Letopisů" instead of "Paralipomenon". Then `<"viz" 1Pa 2:3>` yields viz 1Pa 2:3 in most of Czech translations, except in B21 where it gives viz 1Let 2:3.

## 7.12 Bad links, i.e., links to a non-existent spot

If a reference is made to a non-existent verse or a non-existent note, then there are two possibilities. If it is a reference to a book that is intentionally not being printed (because, for example, you are working on a test books selection, see also `\printedbooks` in section 4.1), then the link is colored as if it were active, but it is not, and TEX gives no warning. However, if the link points to a non-existent verse or note within the printed books, then the link is active, the click goes to the last page of the PDF file, and a warning appears in the log, saying that the link is incorrect. Be aware, though, that whenTEX runs for the first time, all links lead to nowhere (as they are being created for the first time), thus there is a large number of warnings about incorrect links in the log. Only after you run TEX the second time, the links pointing to existing places are correctly interlinked.

## 7.13 Link Tracing

By default, detailed link tracing is enabled in the log file. You can turn this off by saying `\notracinglinks` and back on with `\tracinglinks`. In addition, you can use `\tracingouterlinks` to disable the suppression of link warnings on non-existent books, allowing the log to find any non-existent links resulting from a typo in the reference text.

## 7.14 Links to books and chapters

If you want a reference to a book that would click to the beginning of that book, write `\cref[`⟨*book*⟩`]`. For example, `\cref[Gen]` will print Gen with a link to the beginning of the book of Genesis. Similarly, `\cref[Gen 2]` will print Gen 2 with a click leading to Gen 2:1. If you want to print something else, then `[...]` must be closely followed by `{`⟨*text*⟩`}`, where ⟨*text*⟩ is the text to be printed and to become the active link. So, for example `\cref[Gen 2]{The Day of Rest}` will print The Day of Rest with a link to the beginning of the corresponding chapter (Gen 2:1).

## 7.15 Links to pages

You can place an invisible page link target in your text using `\pglabel[`⟨*label*⟩`]` and then you can link to that page using `\pgref[`⟨*label*⟩`]`; this will print the clickable number of that target page. Similar to `\cref`, you can use `\pgref[`⟨*label*⟩`]{`⟨*text*⟩`}` to print a ⟨*text*⟩ other than the page number; the text now clicks to the place where where the `\pglabel[`⟨*label*⟩`]` is located.

If a link is created by `<...>` then the abbreviation `\pg` may follow and it creates a linked page number which refers to the page where the destination of the previous link `<...>` is. For example `...see note <Dan 1:2>n on page \pg` prints ...see note Dan 1:2 on page 123.

# 8 Maps, images and their legends

The images (maps etc.) can be inserted to the bottom of the page by the `\insertBot` or `\putBot` command:

`\insertBot {`⟨*title*⟩`} [`⟨*label*⟩`] (`⟨*params*⟩`) {`⟨*data*⟩`}`
or
`\putBot` ⟨*chapter*⟩`:`⟨*verse*⟩ `{`⟨*title*⟩`} [`⟨*label*⟩`] (`⟨*params*⟩`) {`⟨*data*⟩`}`

The first one can be used in introductions, the image is inserted at the bottom of the current or the next page. The second one can be used in note files, the image is inserted at the bottom of the page where the given verse is (or at the bottom of the next page). The ⟨*title*⟩ is a caption of the image, the ⟨*label*⟩ can be used for references, the ⟨*params*⟩ can include more typesetting parameters and ⟨*data*⟩ includes commands for inserting the image itself. A comprehensive example will follow in section 8.1.

## 8.1 Translation variants

Just as you can change the wording of a search phrase in a note to match the current core text, you can also create map legends (and similar graphical objects showing expressions that vary in different translations) to always match the current version of the Bible.

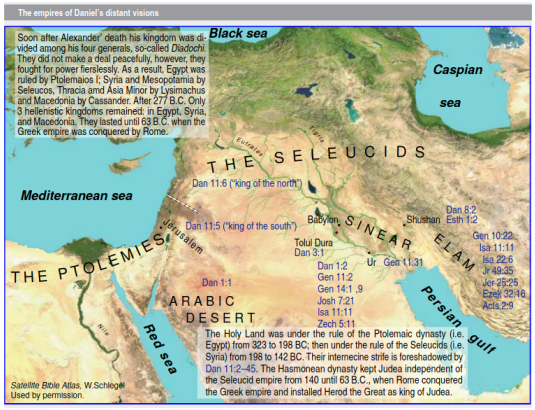
Suppose that we insert a separate blank map and text for the legend. Any text can be placed over an image using `\puttext` command:

```
\puttext ⟨x-dimen⟩ ⟨y-dimen⟩ {⟨text⟩}
```

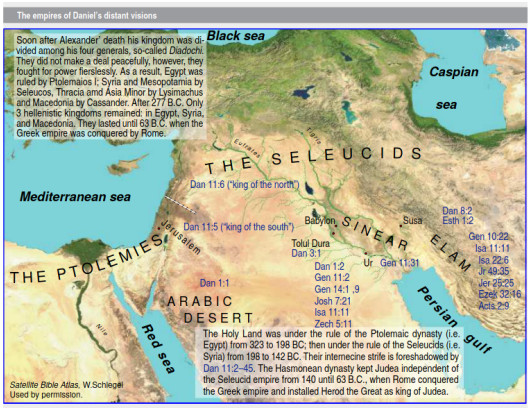
where ⟨*x-dimen*⟩ determines the horizontal offset on the x-axis, ⟨*y-dimen*⟩ the vertical offset on the y-axis, with the coordinates `0mm 0mm` being the bottom left corner of the image. The ⟨*text*⟩ is printed text. For example:

```
\puttext 5mm 62mm {Mediterranean sea}
```

You can insert a text over the map with a white semitransparent shadow as a background of the text. Use `\putstext` instead `\puttext` in this case. The parameters are the same with the same meaning. See section 8.4 for more information about it.




Map from the Introduction to Daniel: BBE




NETfree

For example, you can notice how in the sample above not only that the biblical references are active hyperlinks, but also that the name of the city of Susan (just above the Persian Gulf) changes from the BBE's Shushan into Susa of NETfree.

The image must be loaded before the description can start, and everything in it must be inside the definition of `\insertBot`. The above sample has been written as

```
\insertBot {Daniel's Remote Visions Empires}[map](){
  \inspic{fertile-crescent-crop.pdf}% blind map
  \Heros \cond \setfontsize{at 9pt}\rm %font
  \vskip-1mm
  \putstext 2mm 108mm {\vtop{\hsize6.5cm %box width
                      \baselineskip10pt %line spacing inside the box
                      \noindent %space saving, no need to indent
    \leftskip=3pt \rightskip=3pt %how much the semi-transparent shadow will overlap the text
     Soon after Alexander's death ... was conquered by Rome.}%end\vtop
  }%end \putstext
  \LMfonts\sans \setfontsize{at9pt}\rm
  \puttext 145mm 29mm {<"Acts 2:9"_Acts 2:9>}
  \puttext 145mm 32.5mm {<"Ez 32:16"_Ez 32:16>}
   .
```

```
  .
  .
  \puttext 2mm 5mm{{{{Heros \setfontsize{at 7pt}\it Satellite Bible Atlas,\rm W.Schlegel}}
  \puttext 2mm 2mm{\Heros \setfontsize{at 7pt}\rm Used with permission.}
}%end \insertBot
```

**What to watch out for:** There must not be a blank line inside `\insertBot`.

## 8.2 Macro `\town` for the town symbol on the map

The towns of Jerusalem, Babylon, Tolul Dura, Susan and Ur are visible on the map as tiny circles with a red center and black perimeter. The properties of this circle can be set with the macro `\townparams`, whose default values are as follows:

```
\def\townparams{
   \hhkern=.8pt % radius of the circle
   \lwidth=.5pt % contour line thickness
   \fcolor=\Red % circle color
   \lcolor=\Black % contour line color
}
```

The macro `\town` itself places this circle with coordinates, similar to the macro `\puttext`, but without additional text, e.g.,

```
  \town 101.5mm 53mm %the city Babylon
```

## 8.3 Inscriptions along the curve

Some of the inscriptions on the map require "bending" according to the terrain, especially the names of large areas, as in our case "The Ptolemies" and "The Seleucids," or the "Persian Gulf," or perhaps even the tiny names of rivers ("Euphrates," "Tigris," "Nile"). In order to do this we can use the `\c` command inside the `\puttext` parameter.

`\c[`⟨*first-angle*⟩`/`⟨*kern and rotate parameters*⟩`]{`⟨*text*⟩`}`

For example:

```
\puttext 62mm 70mm {\c[10/\kern7pt\pdfrotate{-1}]{THE SELEUCIDS}}
\puttext 2mm 37mm {\c[0/\kern4pt\pdfrotate{2.5}]{THE PTOLEMIES}}
```

The first number (before /) denotes the angle of the first letter of the text. The `\kern` command determines the spacing between letters; the number in the `\pdfrotate{`⟨*number*⟩`}` determines the strength of the curvature (the angle change between two successive letters). A negative value bends the text concavely (like a rainbow), a positive value convexly (like a bowl).

If we need a sign that waves in the shape of the letter S (is concave and convex at the same time), we would have to assemble it from two or more `\puttext` statements, glued together to look like one continuous text.

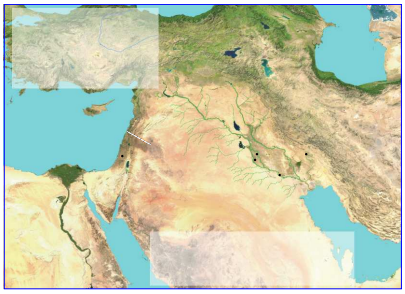
In the example above we have the name of the city of Jerusalem printed at an oblique angle but along a straight line so that it does not clash with "king of the South" and can be seen clearly. This can be achieved by using the `\c` command without `\pdfrotate` parameter:

```
\puttext 48mm 55mm {\c[-40/\kern1pt]{Jerusalem}}
```

The number `-40` was used to tilt the inscription.

## 8.4 Partially transparent background of continuous text

In the sample above, the continuous blocks of text have slightly lighter background through which the map from beneath is still visible. This can be achieved by typing `\putstext` (think of Put-Shadowed-Text) in the place of `\puttext`. This way you don't need any pre-prepared lighter spots on the blind map (see the image bellow), having to "hit" it with the text.



Before using \putstext for the first time, the level of transparency of the white shadow can be set. The default value is \def\shadowparameter{.1}. If you set \def\shadowparameter{1} you get a solid opaque white background; the smaller the number the more transparency.

However, this value is then stored in the page-resources of the output PDF and is used the same on all subsequent pages, so you cannot change it and have it different in different places of the same document. If perhaps there should be an unexpectedly excessive demand for the ability to change the transparency level on the fly, this may be an incentive for implementation in a possible future version of OpBible. For the time being, we did not find it necessary to complicate the macros by creating more and more page-resources, so the user should be satisfied with the option to set the transparency of the shadow under the text on maps uniformly for the entire Bible.

# 9 Timeline inclusion tools

## 9.1 Image or text over two pages

If we want to insert an image or text across two pages in an open double page, we can use

- \insertSpanImage: inserts a prepared PDF image, can be used in the book introduction,
- \insertSpanText: insert text (for example a timeline), can be used in the book introduction,
- \putSpanImage: insert a prepared PDF image, anchored relative to the number chapter and verse, can be used in the notes file,
- \putSpanText: like \putSpanImage, but inserts text instead of an image.

The \insertSpanImage and \insertSpanText commands place the image or text at the bottom of two pages according to the following rule. Suppose that the command itself is executed when TEX creates the current page of the number $c$. Then

- if $c$ is even and the image's (or text's) height allows it to fit on the current page, it will be placed on pages $c$ and $c+1$,
- if $c$ is even and the image or text does not fit on the current page, it will be placed on pages $c+2$ and $c+3$,
- if $c$ is odd, the image or text will be placed on pages $c+1$, $c+2$.

This ensures that the image or text will always be visible on the double page of an open book.

The \putSpanImage or \putSpanText commands work according to the same rule, only the number $c$ in this case corresponds to the page number on which the the beginning of the verse specified in the command parameter.

These commands have the following parameters:

\insertSpanImage {Title} [⟨*label*⟩] (⟨*parameters*⟩) {⟨*filename*⟩}

The ⟨*Title*⟩ is used in the header of the image and ⟨*label*⟩ can be set, to refer to the image (and thus create an active link) using \ref[⟨*label*⟩] at another place. If unused, you can leave the parameter empty: []. Furthermore, ⟨*parameters*⟩ can specify how the image is placed, typically this parameter is empty: (). Finally, ⟨*filename*⟩ is the name of the image file including the extension. Typically this is a PDF file, i.e., it has the extension .pdf.

\insertSpanText {Title} [⟨*column*⟩] (⟨*parameters*⟩) {⟨*text*⟩}

The parameters are the same as \insertSpanImage, only the last parameter is different. It contains the text to be printed across two pages. Typically there might be a set of \timeline, \timelinewidth, \arrowtext, \tlput, \tline, \tlines commands to create a timeline, see the section 9.2.

`\putSpanImage` ⟨*chapter*⟩`:`⟨*string*⟩ `{Title} [`⟨*column*⟩`] (`⟨*parameters*⟩`) {`⟨*filename*⟩`}`

In addition, there is a ⟨*chapter*⟩:⟨*verse*⟩ specifying the verse whose beginning is on page *c* and the image is positioned according to the rules mentioned above. The other parameters are the same as for `\insertSpanImage`.

`\putSpanText` ⟨*chapter*⟩`:`⟨*verse*⟩ `{Title} [`⟨*image*⟩`] (`⟨*parameters*⟩`) {`⟨*text*⟩`}`

It behaves the same as `\insertSpanText`, but with the addition of the ⟨*chapter*⟩:⟨*string*⟩ parameter has the same meaning as in the `\putSpanImage` command.

## 9.2 Commands to create a timeline

These commands can be used in the ⟨*text*⟩ parameter of the `\insertSpanText` or `\putSpanText` commands.

First, you need to specify the number of years (or other units) for the full width of the timeline. All other data will be entered in these units. In the following text we will refer to these units as years. Set the timeline parameters using `\timeline` and `\timelinewidth`:

```
\timeline ⟨number of years⟩
\timelinewidth ⟨width of the timeline⟩
```

For example, after

```
\timeline 500
\timelinewidth 25cm
```

the timeline will be 25cm wide and 500 years will be included in that width, so one year will represent a width of 0.5 millimeters. However, it is more usual to specify the width of the timeline as some fraction of the total width of the page (or double-sided image, if using timeline in `\insertSpanText` or `\putSpanText`). For example: `\timelinewidth=.95\hsize`.

The timeline is built line by line. Commands for text or line segments that are on the same line are written below each other in any order. `\vskip` ⟨*dimension*⟩ command moves to the next line, where the dimension can be specified in multiples of the line height using the unit `\l`, i.e., `\vskip 1.5\l` means a shift of one and a half lines down.

Text is inserted via `\tlput` ⟨*symbol*⟩ ⟨*position*⟩ ⟨*flag*⟩ `(`⟨*setting*⟩`) {`⟨*text*⟩`}`. The ⟨*symbol*⟩ parameter can be "a" if we want the text to be above the current rate position, or "b" if we want the text to be below the current rate position. When the ⟨*symbol*⟩ is "a", multiline text extends upwards, and when "b", it extends downwards.

The ⟨*position*⟩ parameter is the location on the timeline (in years) to which the text should be attached. From this point, the text will flow to the right if ⟨*flag*⟩ is specified by the `\rlap` instruction, and it will flow to the left if the ⟨*flag*⟩ is specified by the `\llap`. Finally, if ⟨*flag*⟩ is empty, the text will have centered lines and ⟨*position*⟩ then corresponds to this line center. The ⟨*setting*⟩ parameter can be be empty, i.e., `()`, or it may contain font settings, font color, etc., for following ⟨*text*⟩. A ⟨*text*⟩ contains the text to be printed. In the case of multi-line text, separate the lines with `\cr` (remember what carriage return meant in the era of typewriters?), for example:

```
\tlput b 25 (\it) {Abraham is\cr 100 years old}
```

The text has two lines, their common center is below point 25 of the timeline.

`\tline` ⟨*from*⟩`..`⟨*to*⟩ creates a horizontal line starting at ⟨*from*⟩ and ending at ⟨*to*⟩. The data is in years.

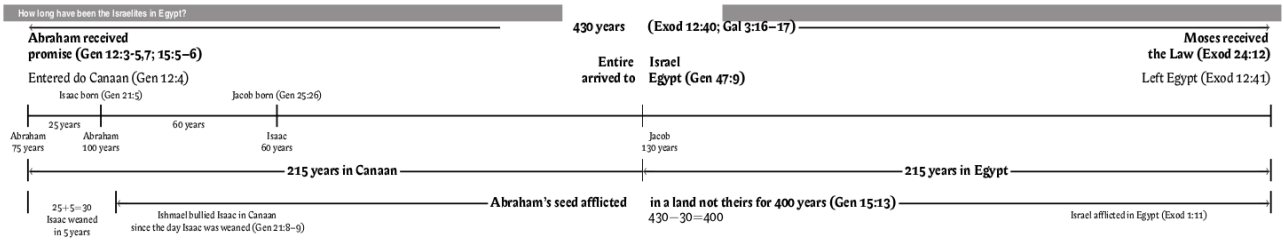
`\tlines` ⟨*w1*⟩`|`⟨*w2*⟩`|`⟨*w3*⟩`|` (and possibly more) inserts short vertical lines on the timeline. The number of ⟨*w*⟩`|` parameters can be arbitrary, each representing one vertical line, and the numbers between them indicate the distance between adjacent lines in years. For example

```
\tline 0..100
\tlines 20|20|20|20|20|
```

creates a 100 horizontal line and vertical lines on top of it to denote 20, 40, 60 and 80 years.

`\arrowtext` ⟨*from*⟩`..`⟨*to*⟩ `(`⟨*setting*⟩`) {`⟨*text*⟩`}` prints a horizontal line from ⟨*from*⟩ to ⟨*to*⟩ (data in years) and the middle of the line is broken to give space for the ⟨*text*⟩ which written at that point. Arrows pointing outwards from the line are at the edges of the line.

This timeline has been shrinked from a double-page size:




and was typeset as follows:

```
\def\small{\typosize[8/10]}
\putSpanText 12:40 {How long have been the Israelites in Egypt?} [] () {%we are in Exodus
\timeline 430         % total number of years (or other units) in the timeline
\timelinewidth 36cm   % the width of timeline (here: 430 years = 36 cm)
\arrowtext 0..430 (\bf) {\hskip20mm 430 years \hskip1cm (Exod 12:40; Gal 3:16--17)}
                      % prints arrow from..to, (format) {text} in the middle;
                      %\hskip controls the spaces in the middle where two pages meet
\vskip 2\l            % \vskip 2\baseineskip; \l is only shortcut for \baselineskip
\tlput a   0 \rlap(\bf)   {Abraham received\cr promise (Gen 12:3-5,7; 15:5--6)}  % print text
\tlput a 430 \llap(\bf)   {Moses received \cr the Law (Exod 24:12)}   % see \_doc above
                                                                      % for more details
\vskip 1.5\l          % \vskip 1.5\baselineskip
\tlput a   0 \rlap()      {Entered do Canaan (Gen 12:4)}              % print text
\tlput a 430 \llap()      {Left Egypt (Exod 12:41)}                   % print text
\tlput a 215      (\bf)   {\hskip-6mm Entire \hskip8mm Israel\cr
                                      \hskip 8mm arrived to
                                      \hskip7mm  Egypt (Gen 47:9)} % print text
                      % all \tlput's between \vskip's is printed in the single line
\vskip 1\l
\tlput a 25   (\small)   {Isaac born (Gen 21:5)}  % text
\tlput a 85   (\small)   {Jacob born (Gen 25:26)}  % text
\vskip1.1\l
\tline 0..430         % the horizontal line from..to
\tlines {0|25|60|130|215|0} % the vertical short lines, numbers mean distance between them
\vskip.2\l
\tlput b 12.5 (\small)    {25 years}    % text printed below
\tlput b 55   (\small)    {60 years}    % i.e the current position is at the top of the text
\vskip.7\l
\tlput b 0    (\small)      {Abraham\cr 75 years}
\tlput b 25   (\small)      {Abraham\cr 100 years}
\tlput b 85   (\small)      {Isaac\cr 60 years}
\tlput b 215  (\small)      {\hskip1cm Jacob\cr\hskip1cm 130 years}
\vskip2.5\l
\tlines {|215|215|}    % the vertical short lines
\arrowtext 0..215 (\bf) {215 years in Canaan}  % arrows and text
\arrowtext 215..430 (\bf) {215 years in Egypt}
\vskip2\l
\tlines {|30|400|}    % the vertical short lines
\arrowtext 30..430 (\bf) {Abraham's seed afflicted in a land not theirs for 400 years (Gen 15:13)}
\tlput b 15 (\small) {25$+$5$=$30\cr Isaac weaned\cr in 5 years}
\vskip.5\l
\tlput b 65 (\small) {Ishmael bullied Isaac in Canaan \cr since the day Isaac was weaned (Gen 21:8--9)}
\tlput b 385 (\small) {Israel afflicted in Egypt (Exod 1:11)}
\tlput b 215 () {\hskip2.5cm 430$-$30$=$400}
}%end of \putSpanText
```

# 10 Page formatting variants

By default, a single-column design is set for the main text of the Bible and for book introductions and annotations. Two-column type is set for notes on verses.

By default, chapter numbers are capitalized and in the outer margin. In the outer margin there are also enlarged quotation marks attached to quotations. The command `\normalchapnumbers` changes this setting: chapter numbers are then inserted in the left upper corner of the first paragraph and the enlarged quotation marks are removed. The outer margin is then shrunk because there is no more printed material in it.

By default, each `\Note` occupies a new paragraph in a double column print. The command `\mergednotes` makes sure that all notes on a single verse are always combined into a common paragraph. However, this comes at a cost of loosing the ability for the phrase in the note to follow its original in the Biblical verse on the same page. In other words, the entire paragraph (compound of several notes on the same verse) links only to the beginning of the verse, i.e., the beginning of the verse and the beginning of the note to that verse are on the same page, but the phrases themselves are not being searched for.

Other page formatting options are still in the planning stages and are not implemented in this version of OpBible.

# 11 Error search options

It may happen that you make a typo in the `notes-*`, `intro-*`, `articles-*` files. If you had included the file with a typo to the whole Bible processing, TEX would report an error at a completely different point in time than when it had read the file. Unfortunately, tracing back such an error is then very difficult. Moreover, typically the error occurs at a different stage of processing and thus it is often reported in a very incomprehensible and misleading message.

However, it is possible to process the newly written files directly first, but with no connection to the core text. In this mode, potential errors are reported more directly. To search for errors directly, use `\checksyntax` ⟨*file list*⟩ `{}` Here the file list are the names of the files to be checked (without the `.tex` extension). You can have, for example, at the end of the main file, line like this:

```
\checksyntax intro-Dan articles-Dan notes-Dan {}
```

This will perform a direct check of the listed files. The output is the text of these files without any formatting. Syntax errors in the files will show up in significantly more straightforward manner.

Note: using `\checksyntax` disables the `\processbooks` command, which in turn does nothing, so the core text of the Bible is not loaded at all.

# 12 Generating default files `notes`, `fmt`, `intro`

If the files `notes`, `fmt`, `intro` do not exist, OpBible does not complain and behaves as if they were empty. But you might want to have these files for all the books of the Bible already prepared in the appropriate directories and with a short introduction. Templates for such files can be written in a separate file (e.g.,`templates.tex`) and then run such file using the `optex templates` command. This will generate the default files according to the specified templates for all the books of the Bible. If a file with the specified name already exists, it is retained (i.e., it is not overwritten by the default file) and a warning about it will appear on the terminal and in the `log` file.

First you need the `books.tex` file with the names and abbreviations for all the books of the Bible, as mentioned in the 4.3 section. This can be generated, for example from Sword using the `mod2tex` script, see section 4.4.

In the `templates.tex` file, the packagge `opbible.opm` must first be loaded using `\input`, then the `books.tex` file, and then you can specify file templates with the `\filegen` command as follows:

```
\filegen {⟨file-name⟩}
⟨file-content⟩
\endfile
```

The ⟨*filename*⟩ must (and ⟨*file-content*⟩ can) contain the double character `@@`, which is automatically replaced by the book abbreviation ⟨*b-mark*⟩ specified in `books.tex`. As many default files as there are book titles declared in `books.tex` are then automatically generated. For example, `templates.tex` might look like this:

```
\input opbible.opm
\input books.tex

\filegen {intro-@@.tex}
%  Introduction to the book @@
\endfile

\filegen {notes-@@.tex}

%  Notes on the book @@

\endinput
\ww
    {}={} % BBE
    {}={} % Jubilee2000
    {}={} % NETfree
    {}={} % UKJV
    {}={} % RNKJV
    {}={} % Webster
\Note 1:1 {}={} %text terminated with a blank line
\endfile
```

The `optex templates` command will generate 66 files in this case `intro-1John.tex`, `intro-1Cor.tex`, etc. with the specified single line and 66 `notes` files (`notes-1John.tex`, etc.) with the content of the specified nine lines.

You can also use the triple character `@@@` in the ⟨*file-content*⟩, which is then converted into the full title of the book. So after specifying `% Introduction to @@@` in the `templates.tex` file you will get the file with the text `% Introduction to The First Book of Moses (Genesis)`, another with `% Introduction to The Second Book of Moses (Exodus)` etc.

In the example, the file names are chosen so that they are generated in the current directory. If you have prepared subdirectories in this directory corresponding to names, it is possible to generate files directly into them, for example, write `\filegen {intros/intro-@@.tex}`.

To generate a list of files with the names other than the ⟨*b-mark*⟩ list in the `books.tex` file, after reading in `books.tex`, define macro `\genbooks` with your chosen set of abbreviations. For example, after

```
\def\genbooks {Gn Ex Lv Nu Dt}
```

the `\filegen` command will only generate templates for the five books with the specified abbreviations. You can also define `\genbooks` before each `\filegen`, in which case you'll be able to have different book abbreviations for different file types (e.g., `intro-Gen.tex` and `notes-Gn.tex`).

**What to watch out for:** If you work online on Overleaf, you must not forget that Overleaf does not keep the files that have been generated during TEX run. This makes generating the templates a bit clumsy: after generating them (but before quitting Overleaf) you would have to download the file(s) and manually upload it (them) again. The files that you manually upload do not get deleted when you leave the project.

# 13 Tricks and Stunts

This sections offers some inspiration for demanding users, eager to perform even greater miracles than what average OpBible users might be satisfied with.

## 13.1 Chiasm

Serious students of the Bible probably know that some parts of it are written in form of chiasm; for example, a single verse:

**A** Many that are first
  **B** shall be last;
  **B'** and the last

**A'** shall be first.
(Matthew 19:30)

or a larger portion of text:

Life of Abraham in Genesis:
**A** Background and Early Experiences (11:10–12:9)
**B** Earlier Contacts with Others (12:10–14:24)
**C** Covenant with God (15:1–17:27)
**B'** Later Contacts with Others (18:1–21:34)
**A'** Progeny and Death (22:1–25:18)

or, for that matter, an entire book:

Leviticus:
**A** Ritual (1–7)
**B** Priesthood (8–10)
**C** Purity (11–15)
**D** Day of Atonement (16)
**C'** Purity (17–20)
**B'** Priesthood (21–22)
**A'** Ritual (23–27)

The way to type such structures is rather easy. Entire chiasm has to be enclosed inside `\begChiasm` and `\endChiasm` pair; then number of asterixes at the beginning of each new line determines the indentation. You only have to type:

```
\begChiasm
* Many that are first
** shall be last;
** and the last
* shall be first.
\endChiasm
\medskip
```

and OpBible will take care of the rest. If you don't want to have the boldfaced **A, B, C... B', A'** at the beginning of the lines, type `\style q` right after `\begChiasm`. If you then want them back, start the new chiasm with `\begChiasm \style Q`.

## 13.2 Formatting Tricks

This section follows the section 4.5. We are formatting the text of the book of Daniel, in the BBE translation, therefore we work inside the `fmt-BBE-Dan.tex` file. Let's suppose you want to let the reader know which words have been added by the translators that are not in the original language.

In BBE, Daniel 1:7, we read:

> And the captain of the unsexed servants gave them names; to Daniel he gave the name of Belteshazzar, to Hananiah the name of Shadrach, to Mishael the name of Meshach, and to Azariah the name of Abed-nego.

The phrase *the name of* was added by the BBE translators, it is not in the original Hebrew Bible and you want to indicate that by a font change. Typing `\fmtfont{1:7}{the name of}{\em}` would do the job just fine, but only for the very first occurrence of the phrase. In this particular case, however, we run into a problem because the exactly same phrase is there four times in the same verse, and all in the need of the font change. The phrase that have been already searched for and found cannot be searched for again.

There is a way around, as mentioned in 4.5.

The macro `\fmtkeep{`⟨*chapter-number*⟩`:`⟨*verse-number*⟩`}{`⟨*phrase*⟩`}` will help us here because it will make the first occurrence of the ⟨*phrase*⟩ unsearchable by hiding it inside the brackets `{}`. We use it repeatedly because we also need to exclude from search the second and the third occurrences of the phrase. The second time it will disable the next first visible phrase, which is now the second one, and so forth.

Typing

```
\fmtfont{1:7}{the name of}{\em}
\fmtkeep{1:7}{the name of}
\fmtfont{1:7}{the name of}{\em}
\fmtkeep{1:7}{the name of}
\fmtfont{1:7}{the name of}{\em}
\fmtkeep{1:7}{the name of}
\fmtfont{1:7}{the name of}{\em}
```

solves the problem neatly and to everybody's satisfaction, resulting in:

> And the captain of the unsexed servants gave them names; to Daniel he gave *the name of* Belteshazzar, to Hananiah *the name of* Shadrach, to Mishael *the name of* Meshach, and to Azariah *the name of* Abed-nego.

See the sample file of Daniel in BBE translation, verse 1:7.

**What to watch out for:** If you want to insert vertical space (e.g., `\medskip`) after the verse in which you have applied the font change on a word, then `\fmtins` containing `\medskip` must precede `\fmtfont` command. Otherwise TEX will crash and you end up with the mysterious message `You can't use \/ in vertical mode.` In other words, use

```
\fmtins{1:7}{Abed-nego.}{\medskip}
\fmtfont{1:7}{the name of}{\em}
```

and do not switch the order of these two lines.

# 14 Summary of basic commands and definitions

This section lists the files necessary to run OpBible, and their typical content.

**Recommended:** Compare the instructions below with the sample files that come as a part of the bundle.

## 14.1 Typically in the `main.tex` file

- `\load`: Macro package to be loaded:`\load [`⟨*macro package*⟩`]`. Example: `\load [opbible]`.
- `\tmark`: `\def\tmark{`⟨*t-abbreviation*⟩`}`: declaration of the t-abbreviation of a Bible translation, e.g., BBE, RNKJV, etc. Example: `\def\tmark{BBE}`.
- `\txsfile`: `\def\txsfile {`⟨*filename mask*⟩`}`: `.txs` location declaration. You can use `\amark` or `\bmark` in the filename mask. Example: `\def\txsfile {./txs/\tmark-\bmark.txs}`.
- `\notesfile`: `\def\notesfile {`⟨*filename mask*⟩`}`: notes files location declaration. Example: `\def\notesfile {./notes/notes-\amark.tex}`.
- `\introfile`: `\def\introfile {`⟨*filename mask*⟩`}`: book introductions location declaration. Example: `\def\introfile {./intros/intro-\amark.tex}`.
- `\articlefile`: `\def\articlefile {`⟨*filename mask*⟩`}`: article files location declaration. Example: `\def\articlefile {./articles/articles-\amark.tex}`.
- `\fmtfile`: `\def\fmtfile {`⟨*filename mask*⟩`}`: formatting data files location declaration. Example: `\def\fmtfile {./fmt/fmt-\tmark-\amark.tex}`.
- `\input`: File to be read in: `\input {`⟨*file-name*⟩`}`. Example: `\input vars.tex`.
- `\printedbooks`: `\def\printedbooks {`⟨*list of a-marks*⟩`}` declares a list of the books, separated by a space: the listed books are processed then by `\processbooks`. Example: `\def\printedbooks {Gen Exod Lev Num Deut}`
- `\processbooks`: instruction to process the Bible, or to be more specific, to process the books listed in `\printedbooks`.

You need to define the commands above in the main file, then read in the files `books.tex`, `vars.tex` and finally start processing with the command `\processbooks`.

## 14.2 Typically in the `books.tex` file

- \BookTitle ⟨*a-mark*⟩ ⟨*b-mark*⟩ {⟨*title*⟩}: declares a-mark, b-mark and title for each book. As each individual book is processed, the \def\amark{⟨*a-mark*⟩} and \def\bmark{b-mark} is executed. The a-mark is used in references within the Bible text, while b-mark may be used in file names. Example: `\BookTitle 2Sa 2Sam {II Samuel}`.
- \BookException ⟨*a-mark*⟩ {⟨*code*⟩}: before processing a book, the ⟨*code*⟩ is executed.
- \BookPre ⟨*a-mark*⟩ {⟨*code*⟩}: after printing the introduction, before the main text of the declared book, ⟨*code*⟩ is executed.
- \BookPost ⟨*a-mark*⟩ {⟨*code*⟩}: after printing the main text of the declared book, ⟨*code*⟩ is executed. Perhaps could be found useful for printing a reading plan but such example of this feature is not implemented in this version of OpBible. Nevertheless, feel free to create your own.
- \nochapbooks: \def\nochapbooks{⟨*list*⟩}: list of the a-marks of the books that are not divided into chapters (only have one chapter). Example:
`\def\nochabooks {Obad Phlm 2John 3John Jude}`.

## 14.3 Typically in the `vars.tex` file

This file declares variant phrases for different versions of the Bible translation.

- \variants ⟨*number*⟩ ⟨*list of t-abbreviations of translation variants*⟩: declares the number of translation variants and a list of t-abbreviations. The ⟨*number*⟩ must match the number of t-abbreviations as well as the number of parameters of the \vdef and \ww commands. Example: `\variants 6 {BBE} {Jubilee2000} {NETfree} {UKJV} {RNKJV} {Webster}`.
- \vdef ⟨*phrase variants*⟩; declares the variants of the phrase, the first of which is the reference one. The phrase variants are placed in brackets {⟨*like this*⟩}. Example:

```
\vdef {He-goat}   % BBE
      {Goat}      % Jubilee2000
      {Male goat} % NETfree
      {He goat}   % UKJV
      {Goat}      % RNKJV
      {He-goat}   % Webster
```

## 14.4 Typically in the `notes-*.tex` file

- \Note ⟨*chap-num*⟩:⟨*verse-num*⟩ {⟨*commented phrase*⟩} ⟨*note text terminated by a blank line*⟩: declares a note. Example: `\Note 5:12 {Belteshazzar} See <"note on" 1:7>n.`
- \ww ⟨*variants of phrase*⟩: must precede \Note. In each translation variant, the following note will be bound to the phrase of a particular phrase according to the currently defined t-mark. Example:

```
\ww {by the order of Belshazzar} % BBE
   {Belshazzar commanded}       % Jubilee2000
   {on Belshazzar's orders}     % NETfree
   {commanded Belshazzar}       % UKJV
   {commanded Belshazzar}       % RNKJV
   {commanded Belshazzar}       % Webster
 \Note 5:29 {}  Belshazzar honored Daniel like Nebuchadnezzar (<2:48>),
      but unlike Nebuchadnezzar he did not honor Daniel's God (<2:46-47>).
```

- \x/⟨*phrase*⟩/: prints the phrase according to the current t-mark and according to \vdef declaration. The ⟨*phrase*⟩ parameter is the reference phrase, i.e., it must be exactly the same as the first parameter of \vdef. If no t-mark is defined or if the first t-mark from the \variants list is defined, then ⟨*phrase*⟩ is printed as is. In any other case (t-mark other than the very first one of the \variants list is defined), ⟨*phrase*⟩ is printed as whatever is declared by \vdef for that particular t-mark. Example: `\x/He-goat/`.
- \putBot ⟨*verse number*⟩ {⟨*title*⟩} [⟨*label*⟩] (⟨*commands*⟩) {⟨*code*⟩}: sets up an image/diagram at the bottom of the page containing the ⟨*verse number*⟩. Example:
`\putBot 2:1 {Daniel's Visions of the Four Kingdoms} [danielsvisions] () {...}`.
- \putArticle: Inserts an article, declared in the `articles-*.tex` file. Example:
`\putArticle 5:20 {Who was Darius the Mede?} [6] ()`

- \putCite: Inserts a quotation on the top of a page above the Biblical text. Example: `\putCite 5:4 {Power corrupts; and absolute power corrupts absolutely.`
  `\quotedby {Lord Acton}}`

## 14.5 Typically in a `intro-*.tex` introduction file

- \title: After you define `\title` as something like this: `\def\title#1{\medskip\noindent{\bf#1}\par\nobreak}` then you can begin some paragraphs with it. Example: `\title{Overview:}`, `\title{Author:}`, etc.
- \insertBot {⟨*MapTitle*⟩}[⟨*label*⟩](⟨*params*⟩){⟨*data*⟩} inserts ⟨*data*⟩ at the bottom of the current page. Example

```
\insertBot {Map title} [label] () {
   \inspic{map-crop.pdf}
   \putstext 10mm 25mm {Text on the map}
}
```

- \Outline starts the last part of the introduction text. It enables nested lists declared by `\begitems`, `\enditems` and the text is printed only to the left column of the page. The command `\rightnote {<comment>}` can be used here; the ⟨*comment*⟩ is printed to the right column. Example: see the end of the file `intro-Dan.tex`.

## 14.6 Typically in an `articles-*.tex` file

- \Article [⟨*chapter-number*⟩] ⟨*text of the article*⟩: Declares ⟨*text of the article*⟩ for the given ⟨*chapter-number*⟩. The ⟨*text of the article*⟩ ends at the end of the file or by another `\Article` declaration. The ⟨*chapter-number*⟩ could be any label, unique for that particular article, but as we suppose that there will most likely not be more than one article per chapter, it sounds logical to assign the number of the chapter where we want the article to occur. This also ensures that you will easily remember the article's label when you call it to appear from the `notes-*.tex` file (see above).
- \Cite ⟨*capital-letter*⟩ {⟨*text of the quotation*⟩ `\quotedby{`⟨*author of the quotation*⟩`}}`. This is the way to declare a quotation within an article, as described in the section 5.7. Capital letter that identifies the article can be ⟨*A*⟩ if it is the first quotation in that article, then ⟨*B*⟩ if it is already the second one, etc. After such declaration, you will probably want to make it visible somewhere. You have to decide where in the article it would be best to insert it, and there you type
- \insertCite `A\left` or \insertCite `A\right`. See section 5.7 for more information.

## 14.7 Typically in a `fmt-*-*.tex` file

Data formatting file (named, for example, `fmt-BBE-Gen.tex`) can contain any of the following commands:

- \fmtpre {⟨*chapter-number*⟩:⟨*verse-number*⟩}{⟨*commands*⟩} inserts ⟨*commands*⟩ before specified verse.
- \fmtadd {⟨*chapter-number*⟩:⟨*verse-number*⟩}{⟨*commands*⟩} inserts ⟨*commands*⟩ after specified verse.
- \fmtins {⟨*chapter-number*⟩:⟨*verse-number*⟩}{⟨*phrase*⟩}{⟨*commands*⟩} inserts commands just after the ⟨*phrase*⟩ that has to exist in the specified verse.
- \chaptit {⟨*chapter-title*⟩} specifies the title to go before the first verse of a chapter, to be used inside `\fmtpre` as its parameter, for example, `\fmtpre{1:1}{\chaptit{CHAPTER TITLE}}`.
- \subtit {⟨*pericope title*⟩} is similar to `\chaptit` except that it comes inside chapter as a title of just part of a chapter. However, it could be used even before the first verse of a chapter, usually in situations where the chapter title is to be immediately followed by a pericope title. Example: `\fmtpre{1:1}{\chaptit{CHAPTER TITLE}\subtit{Pericope Title}}`.

## 14.8 When creating maps

A map or any sort of similar images will be always placed within some text, thus its commands will be called from either Introduction file or Notes file. OpBible allows you to place text on the top of such image. The text over a map has not only the advantages of the notes text (like hyperlinked references, variations according to the translation, or renumbering verses) but also few other as well, like placement by coordinates, text along curved path, partially transparent background under the

text, etc., see the section 8. It makes sense to use these features if the image you use is a blank map with no characters in it.

- `\insertBot {⟨Map's title⟩}[⟨label⟩](⟨optional parameters⟩){⟨data⟩}` Map's title will show up in a horizontal bar above the image; ⟨*label*⟩ serves for reference purposes as usually. Optional parameters can modify the dimensions of the image (e.g., `\picw=3in`) but if you are happy with the default setting (which is `\picw=\hsize`), type in empty parentheses `()`. The last parameter ⟨*data*⟩ includes `\inspic {image-file}` command followed by an optional map legend. The map legend is everything that will show over the image as text of any kind. In can include definitions like
- `\Heros \cond \setfontsize{at 9pt}\rm` to set the font;
- `\putstext 2mm 108mm {\vtop{\hsize6.5cm \baselineskip9pt \noindent \leftskip=3pt \rightskip=3pt Soon after Alexander' death...}}` to type a text with partially transparent background over a map.
- `\puttext 55mm 53mm {<"Dan 11:5 (“king of the south”)"_Dan 11:5>}` to place a text that is an active hyperlink at the same time;
- `\puttext 48mm 55mm {\c[-40/\kern1pt]{Jerusalem}}` to place a text rotated 40°;
- `\puttext 130mm 50mm {\c[-40/\kern4pt\pdfrotate{-1}]{ELAM}}` to place a concave or convex text.

### 14.9 When creating timelines

- `\putSpanText 12:40 {⟨title⟩} [⟨label⟩] (⟨parameters⟩) {⟨the timeline itself⟩}` – you can use an pre-prepared image for a timeline, in which case you would call it by `\putSpanImage` instead of `\putSpanText`. Remember, however, that this will rob you of the possibility to have references as active hyperlinks and variations of phrases following different Bible versions. If you insist on having these features, it is preferable to stick with `\putSpanText`.
  The timeline itself can be created by:
- `\timeline 430` – sets the total number of years (or other units) in the timeline.
- `\timelinewidth 36cm` – the width of timeline (here: 430 years = 36 cm).
- `\arrowtext 0..430 (\bf) {430 years (Exod 12:40; Gal 3:16--17)}` – prints arrow between from..to units with the text in the middle.
- `\vskip 2\l` is equal to `\vskip 2\baselineskip`; `\l` is only shortcut for `\baselineskip`
- `\tlput a 0 \rlap(\bf) {Abraham received\cr promise (Gen 12:3-5,7; 15:5--6)}` – prints text;
- `\tline 0..430` – prints horizontal line from..to units;
- `\tlines {0|25|60|130|215|0}` – prints vertical short lines, numbers mean distance between them.

## 15 Generating technical documentation of OpBible

The detailed technical documentation is prepared. The text of this documentation is part of the file `opbible.opm` itself. You can generate a PDF from this text by the command:

```
optex -jobname opbible-techdoc '\docgen opbible'
```

The `opbible-techdoc.pdf` is created. Use this command three times because the references must be correctly linked and the created Index and Table of contents need data from previous TEX run.

## 16 After you're done

After you finish writing your commentary, you have all the formatting done, you have checked all the references to make sure they are correct, there are no more warning messages in the log file, and you are quite happy about how your Study Bible looks like, you can now disable blue color of the links by typing (somewhere in the `main.tex` file) `\hyperlinks\relax\relax`. (They will still function as hyperlinks, though, only now in `\Black` color, thus indistinguishable from the surrounding text.)

The printing house to which you would submit your resulting PDF should be able to add their own crop marks. If for some reason you want to do it yourself, you can use the macro `cropmarks.tex` which is part of olsak-miscs macro collection.

# 17 Index