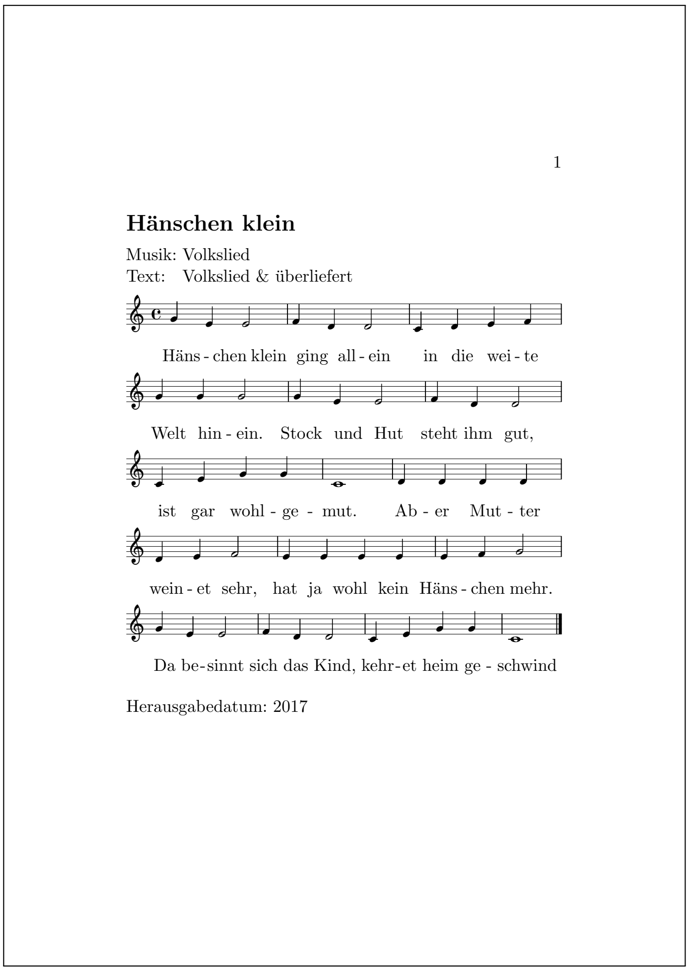
The Liederbuch-Package

[ðə liːdɐbuːx-pækɪdʒ]

Gabriel Ruprecht

created with a lot of help from LaTeX.SX and the German lilypond-forum

January 20, 2025



1

Hänschen klein

Musik: Volkslied
Text: Volkslied & überliefert

Häns - chen klein ging all - ein in die wei - te
Welt hin - ein. Stock und Hut steht ihm gut,
ist gar wohl - ge - mut. Ab - er Mut - ter
wein - et sehr, hat ja wohl kein Häns - chen mehr.
Da be - sinnt sich das Kind, kehr - et heim ge - schwind

Herausgabedatum: 2017

Version 1.1.0

Contents

1 Introduction

The Liederbuch-Package originated in the search for a standardized, simple and fast way, to produce small songbooks for my fraternity[1]. The idea was to provide a few commands to make a songbook. It should also include some designs and be easily usable by non-TeX-safe people.

The sample on the title page uses just the following commands:

```
\documentclass[a5paper, 11pt]{scrartcl}
\usepackage[ngerman]{babel}
\usepackage[utf8]{inputenc}
\usepackage{graphicx}
\usepackage{scrlayer-scrpage}

\usepackage{TestLiederbuch}


\LBHead{%
{\Large \textbf{\print{title}}}\hfill\mbox{}\vspace*{5pt}\\
\makebox[32pt]{Musik:\hfill} \print{composer} \\\makebox[32pt]{Text:\hfill} \print{lyricist}\vspace*{3pt}

}
\LBFoot{Herausgabedatum: \print{date}\hfill\mbox{}}

\setSpaceBeforeStropheValue{15pt plus 5pt}
\setSpaceAfterStropheValue{0pt plus 5pt}
\setSpaceBeforeHeadValue{15pt plus 10pt minus 5pt}
\setSpaceBeforeFootValue{15pt plus 10pt minus 5pt}

\begin{document}

\LBsong{TestLiederbuch}{1}{nt}

\end{document}
```

---

[1] It is a German fraternity and therefore quite different from american ones.

As you see, inside begin-end-document, there is only one line, which inserts the whole song of the songbook. The previous statements are just for the appearance of all songs. To access the songbook, here „TestLiederbuch“, you have to include it in the first place, via `\usepackage{ }` with the songbook name.

The songbook looks like the following. The `\ns`, stands for new syllable (German: neue Silbe). The star just adds a dash between the syllables:

```
\ProvidesPackage{TestLiederbuch}
\RequirePackage{liederbuch}

\begin{liederbuch}[
titleOfLargerWork = Test-Liederbuch &
date = 2017 &
seriesTitle = Demoprojekt
]{TestLiederbuch}

\begin{lied}[title=Hänschen klein & lyricist=Volkslied \& überliefert & composer = Volkslied]{nt}{1}
{%
\notenzeile{\includegraphics[width=\linewidth,page=1]{./TestLiederbuchSnippets/haenschenKlein.pdf}}%
{\nspace{5pt} Häns \ns* chen  \ns klein, \ns \ns ging \ns all \ns* ein \ns \ns in \ns die \ns wei \ns* te \nspace{4}}
\notenzeile{\includegraphics[width=\linewidth,page=2]{./TestLiederbuchSnippets/haenschenKlein.pdf}}%
{\nspace{12pt} Welt \ns hin \ns* ein. \ns \ns Stock \ns und \ns Hut \ns steht \ns ihm \ns gut, \nspace{5}}
\notenzeile{\includegraphics[width=\linewidth,page=3]{./TestLiederbuchSnippets/haenschenKlein.pdf}}%
{\nspace{6} ist \ns gar \ns wohl \ns* ge \ns* mut. \nspace{5} Ab \ns* er \ns Mut \ns* ter \nspace{3} }
\notenzeile{\includegraphics[width=\linewidth,page=4]{./TestLiederbuchSnippets/haenschenKlein.pdf}}%
{\nspace{4} wein \ns* et \ns sehr, \ns hat \ns ja \ns wohl \ns kein \ns Häns \ns* chen \ns mehr. \nspace{2} }
\notenzeile{\includegraphics[width=\linewidth,page=5]{./TestLiederbuchSnippets/haenschenKlein.pdf}}%
{\nspace{5} Da \ns be \ns* sinnt \ns \ns sich \ns das \ns Kind, \ns \ns kehr \ns* et \ns heim \ns ge \ns* schwind \nspace{2} }
}
\end{Lied}

\end{liederbuch}
```

As you see, the big frame is the Liederbuch-environment (songbook), inside which several Lieder (songs) can be. Each garnished with some meta data. If you look back to the cover page, you can see, that this meta data can be used in the header and footer of the songs. Each `\notenzeile` command takes two arguments: The score image and the lyrics.

2 General information

2.1 Terminology

The following list is for understanding the structure of the commands.

1. *GFM* are the tokens which precede every internal command of the package. They also precede some end-user commands. These are the first letters of all the first names of the original author (Gabriel Franz Maria). If some George Florian Michael in Oklahoma uses the same scheme, I'm really sorry, for the interference and incompability.
2. *Lied/Lieder* is German and translates to song/songs.
3. *Buch* translates to book
4. *Heft* translates to notebook, booklet
5. *Liederheft* translates to songbook. It describes a small songbook you use i.e. for weddings and is made out of a few printed pages, stapled together.
6. *Liederbuch* translates to songbook also. But it describes the songbook, you copied the songs for the i.e. wedding from.
7. *Note(n)* means note(s) as in music.
8. *Zeile* means line as in text line.
9. All the internal Commands follow the same rule of
   `\GFM@CurrentPackageAbbreviation@CommandName`

2.2 The Different Parts of the Liederbuch-Package

The following subsections are short descriptions of the packages. The full description is provided in each corresponding chapter.

2.2.1 Liederbuch

This (main) package provides commands and environments to create songbooks which can be used elsewhere. The features are:

- the font of the lyrics appear in the current LaTeX-font (can be overridden in the songbooks)
- customizeable spaces to change the appearance of the imported song. i.e. space before song head, space after song head etc. These spaces also allow the use of `plus` and `minus`, the way they are used in skips.
- use predefined head and foot styles for the songs or create your own styles.
- use meta data in the headers and footers. i.e. song title, composer, copyright, etc.

This package is not used in the document but in the sty-file, which serves as the songbook. If you have your songbook defined in the preamble, you need to load the package in the document of course.

2.2.2 Liederheft - unfinished, better do not use yet

This is a class, which provides commands and environments to easily create small booklets. This includes several predefined designs. This package is not limited to songbooks. It can be used for anything which has the character of a stapled home made booklet for one time use.

2.2.3 Print Liederbuch

This allows you to list all content of a songbook (liederbuch), you created. It is much more comfortable than looking at the source code.

3 Liederbuch-Package

This package provides the commands to create the songbooks, aka databases, where you choose the songs from.

3.1 Basics

On figure 3.1 on the following page you see the layout of each song. The head, notes, songtext and foot is grouped in boxes. These boxes are vertically aligned according to the spaces which can include `plus minus`. Example:

```
\setSpaceBeforeHead{30pt plus 1fil minus 2pt}
```

About the songtext and its alignment, we get into detail later.

3.2 What the packages uses

The package makes use of the following packages. This is just for your information and has almost no practical use. It is mostly internals:

- xparse
  Used for commands with more than one optional arguments and use of the * for multiple variants for commands.
- graphicx
  Not explicitely necessary, but necessary in most cases. Includes the note-snippets.
- etoolbox
  Used for creating command sequences
- environ
  Used for creating the Environments for the Liederbuch-package
- pgfmath
  Needed for using length in relation to the linewidth. If you do not use any unit for `\nspace{<some value>}`, it is read as percent of linewidth. This is very usefull, as the target linewidth might change a bit, and the alignment of the syllables wouldn't fit to the notes.

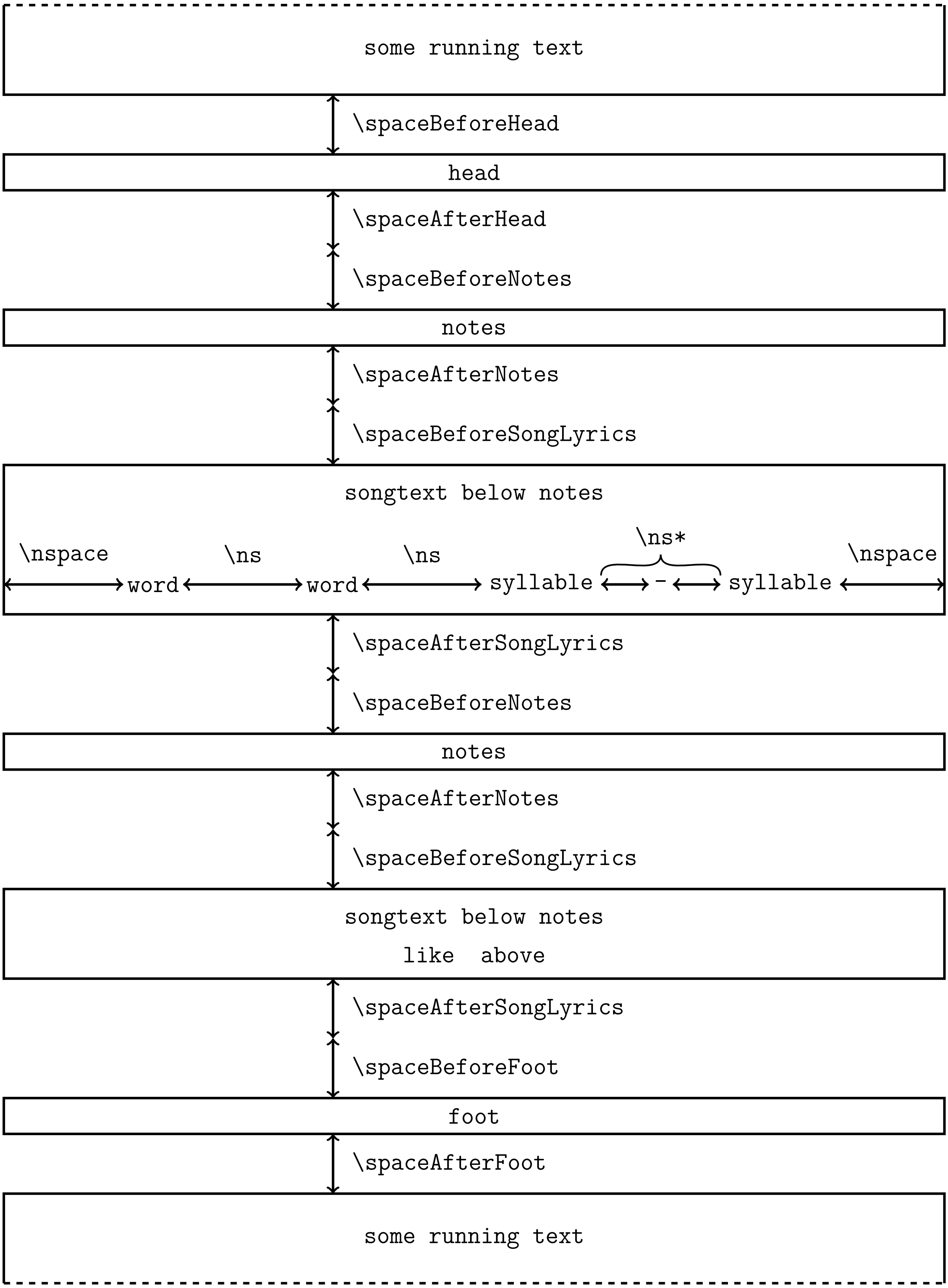


Figure 3.1: layout and spacing of a page with several musical lines

3.3 User Commands

3.3.1 Building a Songbook

Every songbook consists of at least one sty-file. You can also put the songbook into the preamble like in GFM-Setzhilfe[1] (typesetting helper), but the intended use is to have a separate file. This file contains the actual database (and may call other databases/songbooks itself). Furthermore, there may be score snippets for adding notes to the lyrics, pixel graphics from scans and other resources.

If you create and package a songbook, it should follow certain rules concerning the folder structure:

```
A new folder for all the content:
./this_is_an_example

And in this folder the sty-file and pdf-snippets (or png-
   graphics for example):
./this_is_an_example/this_is_an_example_songbook.sty
./this_is_an_example/01_song_snippet.ly
./this_is_an_example/01_song_snippet.pdf
./this_is_an_example/02_another_song_snippet.ly
./this_is_an_example/02_another_song_snippet.pdf
./this_is_an_example/03_summertime_(gershwin).ly
./this_is_an_example/03_summertime_(gershwin).pdf
./this_is_an_example/04_On_the_Road_Again_Version-A.ly
./this_is_an_example/04_On_the_Road_Again_Version-A.pdf
./this_is_an_example/04_On_the_Road_Again_Version-B.ly
./this_is_an_example/04_On_the_Road_Again_Version-B.pdf
```

If you have a songbook made by someone else, you are fine. If you haven't, you need to build it yourself. The preamble of the sty-file contains the following lines:

```
    \ProvidesPackage{name of the songbook}
    \RequirePackage{liederbuch}
```

You may add some of your own packages you want to use in the songbook. For example if you typeset the scores with musixtex, you can add this package. If you want to use tikz instead, you are free to go. Most likely, you will use the graphicx-package, as using pdf-snippets is the imho best way, to use this package.

To start a songbook, you must first write these lines:

```
    \begin{liederbuch}[<meta data>]{<name of the songbook>}
        % the songs (Lieder-environments) go here
        [...]
```

---

[1] This file can be downloaded from the git repo of the package: bitbucket–liederbuch-package

```
        [...]
        [...]
    \end{liederbuch}
```

This must contain all songs. This environment is provided by *liederbuch*. What the meta data does, we will come to at a later point. After you created a songbook, you can create songs inside it. Every song number can be included multiple times in different variants. If two songs have the same number and the same variant tag, the later overwrites the first:

```
    \begin{lied}[<meta data>]{<variant>}{<song number>}
        <the song content>
    \end{lied}
```

The song number identifies the songs. If you have to different kinds of song number 123 (i.e. piano score and choir score), you can make two different variants. i.e. `ps` and `cs`. The way, you name the variants, is up to you. At can be anything beside the `\empty`-command-sequence.

Neither song number, nor variant has to be an actual number. You can write ”One” instead of ”1”. It will work, but you should be at least consistent over the whole song boook. And writing eight-hundred-sixty-one instead of a numeral is highly depreciated.

Example

Assuming you have the following songbook created:

```
    \begin{liederbuch}{testSongbook}
    \begin{lied}[%
        meta data=this is just a test &%
        otherData = it really is]%
    {var1}{144000}
    No notes and text yet.
    \end{lied}
    \end{liederbuch}
```

If you call the song with

```
    \LBsong{testSongbook}{144000}{var1}
```

the ouput will be

```
    No notes and text yet.
```

The meta data elements are not used yet, but can be used later.

The output should be an actual song of course. To typeset these, there are several commands inside(!) the Lied-environment available.

Extended example

You start of course with the Liederbuch environment first:

```
\begin{liederbuch}[%
    songbook=Old Irish Folk Songs &%
    author = Travis O'Connor
]

\begin{lied}[
    title = What shall we do with a drunken saylor &
    composer = Traditional &
    lyricist = Probably a drunken saylor
]{nt}{23}
    %The notenzeile command goes here
\end{lied}

\begin{lied}[
    title = My lad got beaten up in the pub &
    composer = Johnny O'Neeye %
    lyricist = Jimmy Jailedup
]{nt}{24}
    %The notenzeile commands goes here
\end{lied}

\end{liederbuch}
```

For every combination of notes and text, you use one notenzeile command. If your song score has 5 lines of notes, you need the notenzeile command 5 times, if it is 3 lines, you need it 3 times. The notenzeile command is smart enough to show several strophes of text for one line, if needed:

```
\notenzeile[linewidth of the line | default:\linewidth]
{the part with the notes from the score}%
{the text of strophe 1}%
[the text of strophe 2]%
[the text of strophe 3]%
[the text of strophe 4]%
[the text of strophe 5]%
[the text of strophe 6]%
\notenzeile{\includegraphics[width=\linewidth,page=2]{
   somePath}}%
{\nspace{5} This \ns is \ns the \ns line \ns two \nspace
   {3}}
```

```
\notenzeile{\includegraphics[width=\linewidth,page=3]{somePath}}%
{\nspace{3em} and \ns three \ns of \ns the \ns do \ns* cu \ns* ment \ns of do cu ments\nspace{7pt}}
```

As you see, the strophe numbers 2 and above are optional arguments. Standard case should be one line of text for one line of notes, but you can add more. If you need more than 6 strophes below one line of notes, you do something, which you shouldn't do. If you want more, you can hack the source code.

Inside of the notenzeile command, there are several commands available:

```
\songLyrics{}
% This in an internal command, but you can use it.  It creates a makebox, with the \ns, \ns* and \nspace commands available inside.

\ns[space correction]
% This divides two words. You can optionally add an offset to the command, which shifts the next syllable (positive = right, negative = left). You can use standard length units. If you don't use any unit, percent-linewidth is used.

\ns*[space correction]
% It is exactly like \ns, but it adds a dash in between. It is meant to divide syllables of one word.

\nspace{length}
% This does exactly the same as hspace*, but this command also accepts no unit (just a number without an unit), which is interpreted as percent-linewidth.

\multiline{0.5\linewidth}{text for strophe 1}{text for str 2}[str. 3][4][5][6]
% This is usefull, if your song switches to refrain within one line or only some section of one song line has two text lines. It creates several stacked lines within one line.
```

So far, this creates all the notes and text together. Normally, songs are typeset, that only the first strophe is below the notes, the other strophes are printed as text blocks.

These text blocks are built by the strophe environment:

```
\begin{strophe}[strophe number]
```

```
    Some text in any format.
\end{strophe}
```

Normally, the strophe numbers are automatically counted up for every use of the strophe environment. So, the first time, you use this environment in each song, you should override this with the optional parameter strophe number. I.e. if the first usage is number 2, then you write `\begin{strophe}[2]`. The next strophe will then be automatically 3.

3.3.2 Using a Songbook

Using a songbook is pretty easy. You must include the songbooks with

```
\usepackage{nameOfSongbook}
```

and call the desired song with

```
\LBsong{songbook}{number}{variant}
```

Note: The name of the package you use needn't be the same name of the songbook. You can define the songbook `abc` inside a sty-file called `alphabet`. But it is highly discouraged.

There are several versions of each songnumber possible. Due to the variant, you can store the piano version of a song in the same songbook as the choir music version. If you use `\LBsong{paradiseSongs}{42}{4voice}` for example, you get the song number 42 out of the paradiseSongs-songbook in the version for four voices (which of course you must have defined beforehand).

If some meta data is incorrect and you can not or shouldn't change the source songbook, you can edit the meta data with the

```
\tweakMetaData{songbook}{song number}{song variant}{meta
   data element}
```

command. It just overwrites the meta data element, that was created for that specific song at load time. The principle is the same as in `\GFM@LB@unpackage`.

3.3.3 List of Songs (toc)

You can create a list of songs with

```
\listofsongs
```

It has by standard the same format as `\listoffigures`. The style can be selected by using

```
\listofsongsstyle{<style>}
```

The available styles are:

- ”simple” (default)
  `<song title> ...[...]... <page number>`
- ”simple with number”
  `<song number in the songbook> <song title> ...[...]... <page number>`
- ”simple with document number”
  `<song number in the document> <song title> ...[...]... <page number>`
- ”twolines”
- ”semitwolines”

The first three work. The other two don’t work yet.

3.3.4 Refs, References and Hyperref

Hyperref is supported and should work out of the box, if loaded. Each song can be referenced by

```
\pageref{LB_lied_<songbook>_<song number>_<variant>}
```

If you include a song more than once, each duplicate must have a suffix (a for first duplicate, b for second, c for third,...):

```
\pageref{LB_lied_<songbook>_<song number>_<variant>_<
   letter>}
```

Note: The first occurence has no letter suffix, the second occurence start with suffix ”_a”.

3.3.5 Babel

The Babel package is supported. The following terms are supported and can be used in the header and footer (see section Creating a Theme). The supported terms are:

```
\listofsongsname
```

```
\LBsongComposer
```

```
\LBsongLyricist
```

```
\LBsongEditor
```

`\LBsongYear`

`\LBsongCopyright`

`\LBsongEditorialOffice`

`\LBsongEditon`

`\LBsongPrint`

Each of these terms will be displayed in the selected language. If the alphabet uses upper-lower-case, the first letter is always a capital letter. Except for German of course, which has their own system and English, which uses ”This Kind of Unique Way”. If you find a translation mistake (by ChatGPT, though I checked it) or miss a language, please open a bug report (see

The supported languages are:

acadian
afrikaans
albanian
american
australian
austrian
bahasa
bahasai
bahasam
basque
brazil
brazilian
breton
british
bulgarian
canadian
canadien
catalan
croatian
czech
danish
dutch
english
esperanto
estonian
finnish
francais
french
frenchb
galician
german
germanb
greek
hebrew
hungarian
icelandic
indon
indonesian
interlingua
irish
italian
latin
lowersorbian
magyar
malay
meyalu
naustrian
newzealand
ngerman
norsk
nynorsk
polish
polutonikogreek
portuges
portuguese
romanian
russian
samin
scottish
serbian
slovak
slovene
spanish
swedish
turkish
ukrainian
uppersorbian
welsh
UKenglish
USenglish

3.3.6 Creating a Theme

The appearance of the songs consist mainly of three things:

- Header
- Footer
- Spacing

Changing the background of the songs isn't supported yet and might never be.

The header and footer can be edited via

```
\LBHead{Definition}
```

and

```
\LBFoot{Definition}
```

Here you can use the meta data-values given in the song. As you might remember, the songs are defined as i.e.

```
\begin{liederbuch}[
    titleOfLargerWork = Test-Liederbuch &
    date = 2017 &
    seriesTitle = Demoprojekt
]{TestLiederbuch}

\begin{lied}[
    title=Hänschen klein &
    lyricist=Volkslied \& Überliefert &
    composer = Volkslied
]{nt}{1}
    ... something ...
\end{lied}
\end{liederbuch}
```

An example header and footer would look like this:

```
\LBHead{\Large\print{title}\\
\normalsize Taken from \print{titleOfLargerWork}\hspace{30
  pt}\print{composer}}

\LBFoot{\print{date}\hfill\print{seriesTitle}}
```

As you see, the print-command prints any meta data, which had been defined before. If it is undefined, the print-command prints nothing. This example is of course not the peak of human art work, but it showcases the concept. It is possible to include almost anything (no limitations known yet). For the naming of the meta data refer to section 3.3.9 Meta Data on page 22.

For adjusting the spaces, you can use the following commands. They work with `\vskips` and can optionally take `plus` and `minus`. These are the commands to edit the spaces with some example values:

```
\setSpaceBeforeHeadValue{3pt plus 0pt minus 1.5pt}
\setSpaceAfterHeadValue{-1pt plus 4pt}
\setSpaceBeforeNotesValue{2pt minus 1pt}
\setSpaceAfterNotesValue{4pt}
\setSpaceBeforeStropheValue{3em plus 1pt minus 0.2in}
\setSpaceAfterStropheValue{0pt}
\setSpaceBeforeSongLyricsValue{0pt}
\setSpaceAfterSongLyricsValue{0pt}
\setSpaceBeforeFootValue{0pt}
\setSpaceAfterFootValue{0pt}
```

3.3.7 Using printliederbuch

If you want to have a catalogue of all your songs in a songbook, you can use `printliederbuch`. You just need a simple document like this:

```
\documentclass[a5paper, 11pt]{scrartcl}
\usepackage[ngerman]{babel}
\usepackage[utf8]{inputenc}
\usepackage{printliederbuch}
\usepackage{TestLiederbuch}
\usepackage{AnotherTestLiederbuch}

\begin{document}

\tableofcontents

\printLiederbuch{TestLB}
\printLiederbuch{AnotherTestLB}

\listofsongs

\end{document}
```

You need a `documentclass` of your liking, add some packages of your liking and then you need first, the printliederbuch package and second, the songbook, you want to have a catalogue of. The order is important. Your document needs only one command, the `\printLiederbuch` command. The argument of `\printLiederbuch` is not the `\usepackage` name, but the name, defined in `\begin{liederbuch}{LBname}`.

Currently, the table of contents only shows the song number and the variant, while the list of songs show the name only.

3.3.8 Tips and Tricks

First the outer spacing, then the syllable spacing

Start by doing the outer spacing first. If these are your lyrics

```
{Hänschen klein ging allein in die weite}
```

you start by adding the outer space and separating all syllables

```
{\nspace{5} \ns Häns \ns* chen \ns klein \ns ging \ns all
   \ns* ein \ns in \ns die \ns wei \ns* te \ns \nspace{2}}
```

Once the first and last syllable looks good, you can proceed with the adjustment of the space between the syllable (see next sections). If you don't use a `\ns` after/before `\nspace{x}`, the lyrics are easier to align, but they are less robust against different linelength. If your song shall work well in A4 and A5 either, consider using it.

Replacing $\backslash ns*$ with $\backslash ns - \backslash ns$

If `␣\ns*␣` separates two long syllables like "Schne-cken", `␣\ns*␣` will to its job. But if the syllables are short like "O-lé" or the notes are long, you better replace it with `␣\ns␣-␣\ns␣`. If this is a little bit too long, you can use `\ns␣-\ns` instead (no spaces).

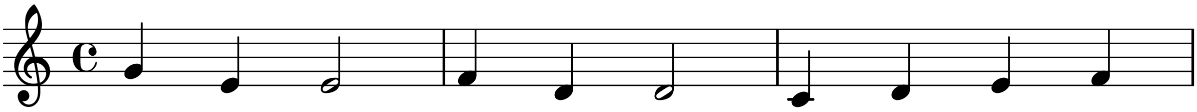
Double $\backslash ns$ and $\backslash nspace\{x\}$

Normally in songs, note lengths tend to be only two different lengths per line. i.e. if your shortest length is a quarter note, the second length is a half note. If the shortest is an eighth, the second one is a quarter. It is rather rare, that a third length occurs. Maybe a dotted note. Most notation software makes the space after a longer note a little bit longer.

Best practice has been to make one `\ns` after each shortest note duration and two `\ns` after the longer notes. For even longer notes it is better to use a `\nspace{x}` inline (with x=3 for example). i.e.:

```
\nspace{7} Häns \ns* chen\ns klein \nspace{2}\ns ist \ns weg,
   \ns \ns Kro \ns* ko \ns* dil \nspace{4}
```

This example is complete fictional and doesn't occur in any real song.



gets this song text:

```
\nspace{7} Häns \ns* chen\ns klein \ns ging \ns all \ns* ein \
   ns \nspace{4} in \ns \ns die \ns \ns wei \ns* te \nspace{4}
```

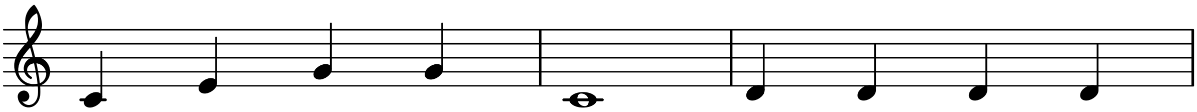
7% linewidth space on the left, 4% on the right. The syllable ”-chen” is very long. Therefore, a second `\ns` isn’t needed, though the corresponding note is a half note. ”in” and ”die” are very short syllables, a second `\ns` does the trick. After ”`\ns*` ein” two `\ns` isn’t enough, therefore a `\nspace{4}` does the trick.

Omitting a space here and there

You might have noticed in the previous example, that after ”-chen” there is no space. Normally it is `␣\ns␣` and normally it should be. But in this case, the syllable is already pretty long and we need to get rid of a bit of space. Therefore, the space before is omitted and it is `\ns␣` instead. You can only omit the space before `\ns` not after, as the space after is eaten by the compiler anyway.

Hard spaces (~) for short syllables

In die following example



the text is like this:

```
\nspace{6} ist \ns gar \ns wohl \ns* ge\ns -\ns mut. \nspace
   {6} Ab \ns* er~ \ns Mut \ns* ter \nspace{3.5}
```

The syllable ”`\ns*` er” is very short. The spacing will work better, if it is a little bit longer, so we add a hard space (~) to the syllable to make it longer.

3.3.9 Meta Data

The Sheet Music Consortium has created a list of tags for their sheet music to make it easy to find in the database. This seems to be a sensible scheme and is therefore partially reused for the meta data (see `https://digital.library.ucla.edu/sheetmusic/aboutProject.html#NameforSMCmetadata-NameMetadata`). The scheme here is mostly taken from there. Another scheme is `https://www.dlib.indiana.edu/smcmigrator/fieldnamehelp.php`. A website which doesn't exist any more. There are also some parts taken from there. All the tags in these schemes had been unified and consolidated in this list here. Therefore it is kind of a third deviating scheme taking from both of them and trying to have as least dicrepancies to them as possible. This scheme here also ommits a few parts, which seemed to duplicate others or seemed totally over the top. This section is already totally over the top and awfully perfectionist.

Subsections:

3.3.9.1 Titles

Each resource should have at least one title element associated with it. The following fields can be used to describe titles:

Tag		value examples	description
`title`	=	`Gaudeamus igitur`	A word or phrase that names the resource being described.
`subtitle`	=	`iuvenes dum sumus`	Secondary word or phrase that names the resource being described.
`alternativeTitle`	=	`De brevitate vitae`	The ”alternative title” data is used for other titles not covered elsewhere in the meta data record.
`firstLine`	=	`Gaudeamus igitur, iuvenes dum sumus`	The ”First Line” data is a direct transcription of the first line of lyrics appearing in the song.
`firstLineofChorus`	=	`no chorus`	The ”first line of chorus” data is a direct transcription of the first line of the chorus (refrain) appearing in the song.
`titleOfLargerWork`	=	`Allgemeines deutsches Kommersbuch`	The ”title of larger work” data is used when the item being cataloged is known to be one part of a larger work with a known title.
`seriesTitle`	=	`German student song books`	The ”series title” data is used to record a named series to which the item being cataloged belongs.
`uniformTitle`	=	`no idea, sorry`	A Uniform Title (also known as a Work Title) is a specific cataloger-created title as prescribed by the rules in AACR2. It is used to indicate clearly the musical work represented in a piece of sheet music.

3.3.9.2 Names

Tag		value examples	description
`composer`	=	`traditional`	”Composer” meta data is used to record the name of individuals or corporate bodies responsible for creating the musical content of the work being cataloged.
`arranger`	=	`I don't know`	”Arranger” meta data is used to record the name of individuals or corporate bodies responsible for the transforming the musical content of the work being cataloged from its original form, genre, instrumentation, etc., to another for publication. In an arrangement the musical substance remains essentially unchanged.
`lyricist`	=	`some guy of the 19th century`	”Lyricist” meta data is used to record the name of an individual or corporate body responsible for creating the lyrics or text of the work being cataloged.
`performer`	=	`The "hohe Kneipcorona"`	”Performer” meta data is used to record the name of an individual or corporate body indicated on the item being cataloged as a known performer of the work.
`dedicatee`	=	`Not for Elise`	”Dedicatee” meta data is used to record the name of an individual or corporate body to whom the work or publication is dedicated. Do not us record information about handwritten dedications printed on an item after publication; use ”Note” for this purpose instead.

`otherName`	=	`Unter Mitwirkung der Lindenwirtin entstanden`	”Other name” meta data is used to record the name of an individual or corporate body responsible for the creation of the item being cataloged that is deemed important but not appropriate for use in any other ”Name” or ”Publication” data. Do not record individuals or corporate bodies named in lyrics; use ”Name as Subject” for this purpose instead.
`engraver`	=	`Gabriel Ruprecht`	The person or organisation, who engraved the sheet music. Commonly with the help of a program like lilypond.
`lithographer`	=	`Heidenhain 600`	The person or company producing the printing plates of the sheet music, if deviating from ”publisherName”.
`coverEngraver`	=	`Andy Warhol's wife`	”Engraver” meta data is used to record the name of an individual or corporate body responsible for the cover engraving on the publication being cataloged, if deviating from ”engraver”
`coverLithographer`	=	`The gremlin in Andy Warhol's basement`	Use ”Lithographer” to record the name of an individual or corporate body responsible for the cover lithography on the publication being cataloged, if deviating from ”lithographer”
`coverArtist`	=	`Andy Warhol himself`	”Artist” meta data is used to record the name of an individual or corporate body responsible for the cover art on the publication being cataloged.

3.3.9.3 Subject

Tag		value examples	description
`topicalSubject`	=	`Drinking beer`	In a song with lyrics ”Topical Subject” is used to record what topics occur in the song’s lyrics.
`nameAsSubject`	=	`Professor Ubisunt Quiantenos`	”Name as Subject” is used to record a personal or corporate name that is the subject of a song’s lyrics.
`formGenreStyle`	=	`Student songs`	”Form/Genre/Style” may be used to record the class or category of music, including the musical style(s) of a work. Form is often described as the organizing element in music, or the historically or functionally specific kind of material.
`temporalSubject`	=	`Romantic period`	The ”Temporal Subject” meta data is used to record a named time period relevant to the item being cataloged.
`instrumentation`	=	`Vocal`	A listing of the performing forces called for by a particular piece of sheet music, including both voices and external instruments.
`placeNameSubject`	=	`Germany`	”Place Name Subject” is used to record named countries, states, provinces, counties, and cities associated with the music and lyrics of the item being cataloged.
`otherGeographicSubject`	=	`D-A-CH-Region`	The optional ”Other Geographic Subject” meta data is used to record named geographic places that are not countries, states, provinces, counties, or cities associated with the music and lyrics of the item being cataloged.
`localSubject`	=	`University`	”Local Subject” is used to record subject terms meaningful to the holding institution.

`coverSubject`	=	`Loreley in a sleeping dress`	”Cover Subject” is used to record the topical content of the image depicted on the cover of the item being cataloged.

3.3.9.4 Identification Numbers

Tag		value examples	description
`plateNumber`	=	`I have no clue.`	The Sheet Music Consortium adopts the AACR2 definition of a sheet music plate number, as distinct from the publisher number: ”A numbering designation assigned to an item by a music publisher, usually printed at the bottom of each page, and sometimes appearing also on the title page. It may include initials, abbreviations, or words identifying a publisher and is sometimes followed by a number corresponding to the number of pages or plates.”
`publisherNumber`	=	`Still no clue`	The Sheet Music Consortium adopts the AACR2 definition of a sheet music publisher number, as distinct from the plate number: ”A numbering designation assigned to an item by a music publisher, appearing normally only on the title page, the cover, and/or the first page of music. It may include initials, abbreviations, or words identifying the publisher.”
`catalogNumber`	=	`BWV31029`	An identifying number for a musical composition assigned by the composer, publisher or researcher. Bach-Werksverzeichnis for example.

`callNumber`	=	`No clue again`	An institution’s local identifier indicating the physical location of the item within the collection.
`otherIdentifier`	=	`Dunno, 1234 maybe?`	Any numbering/naming designation used to identify an item within a collection that is distinct from plate number, publisher number, catalog number, or call number.

3.3.9.5 Relational Identifier

Tag		value examples	description
`precedingItem`	=	`ADKB-34`	Identifier for a related item that is the immediate predecessor of the resource in a chronological relationship. Normally used for continuing resources such as serial publications.
`suceedingItem`	=	`ADKB-36`	Identifier for a related item that is the immediate successor of the resource in a chronological relationship. Normally used for continuing resources such as serial publications.
`originalItem`	=	`ADKB-35`	Identifier for the original item from which the resource was derived.
`hostItem`	=	`ADKB`	Identifier of the host item for the constituent unit in a vertical relationship. This information allows users to locate the physical piece that contains the component part described in the record.
`constitutentItem`	=	`no clue`	Identifier of a constituent part that has been described separately. This information allows users to located a related unit of the resource, particularly when it has been physically separated from the item of which it is considered a part.

`otherVersionOfItem`	=	`CVLB-68`	Identifier of a related version of the resource described in the record, such as translation in another language.
`otherFormatOfItem`	=	`ADKB-35--4-voices`	Identifier of a different format of the resource described in the record, such as a microform reproduction.

3.3.9.6 Notes/Description

Tag		value examples	description
`notes`	=	`There exist contradicting orders for the strophes 2 and 3. This version sticks to the one published in Riedingers Gesangsbuch; Line three contains a typo, that was in the original.`	Any notes, that don't fit anywhere else. The "Notes" meta data is used to record information that supplements information in the rest of the meta data record.
`description`	=	`A song about friendship, beer and wanderlust.`	A general description.

3.3.9.7 Date Information

Tag		value examples	description
`date`	=	`1631`	"Date" meta data holds the date an item was copyrighted or published.
`dateOfWork`	=	`1630-12-31`	The "Date of Work" records the date the piece of music was written.

3.3.9.8 Language

Tag		value examples	description
`language`	=	`deu`	Use ”Language” to record the language of the lyrics of the item being cataloged. Either i.e. German oder the ISO 639-3 code.

3.3.9.9 Type of Resource

Tag		value examples	description
`typeOfResource`	=	`notated music`	There seems to be a definition in the AACR2. But it is not open. So I can’t provide other values here.

3.3.9.10 Publication Meta Data - Origin Info

Tag		value examples	description
`publisherName`	=	`Guthenbergs Kommersbuchdruckerei`	Those responsible for the physical production and dissemination, of the resource.
`publisherPlace`	=	`Heidelberg`	The name of the place where a resource has been published.
`placePublished`	=	`Leipzig`	Where it had been published. i.e. Leipziger Buchmesse → Leipzig
`publisher`	=	`Verlag Arnold Schönberg - Weimar`	Name of the publisher.
`placePublisher`	=	`Weimar`	Where the publisher is located.
`dateIssued`	=	`1841-08-01`	When it had been published.
`copyrightDate`	=	`1841`	The year relevant for copyright and its expiration.
`dateCreated`	=	`1840-04-03`	When the work had been made. This must be some date prior to publishing.
`dateModified`	=	`1842-10-03, 1848-06-02`	If a work underwent some modifications like fixing typos.

`dateValid`	=	`1947`	Works with limited validity. i.e. phone books, pharamceutical drug lists. Can be for example also the expiration of the copyright.
`otherDate`	=	`1848, 1870-09-02`	Other relevant dates, which may be mentioned in a work or where the work has been used in a notable way
`edition`	=	`3`	In cases of reprints, it specifies, which edition this had been taken from.

3.3.9.11 Physical Description

Tag		value examples	description
`formMedium`	=	`book`	In what medium had this been published in. Book, leaflet, brochure, single page;
`physicalDescriptionNote`	=	`Original lost in fire`	Can be any useful information about the original copy.

3.3.9.12 Abstract

Tag		value examples	description
`abstract`	=	`This is a famous folksong, written down in 1843 by Hans Eicher. It gained popularity during the 1848 revolution and in the 1870/71 war.`	An abstract, if needed or wished.

3.3.9.13 Table of Contents

Tag		value examples	description

`tableOfContents`	=	`1. Intro`
		`2. Sailor songs`
		`3. Hiking songs`
		`4. Drinking songs`
		`5. Hooligan songs`
		`6. Wedding songs`
		`a) If the bride is beautiful`
		`b) If the bridesmaids are beautiful`
		`7. Again drinking songs`
			The table of contents

3.3.9.14 Instrumentation

Tag		value examples	description
`instrumentation`	=	`chamber orchestra`	A meaningful and well established value about the instrumentation of the piece.

3.3.9.15 Cartographics

This part is only for completeness. The original ressource, where this is from isn't available any more. Therefor a complete citation is not possible.

Tag		value examples	description
`scale`	=	`1:200000`	Scale is used to capture the ratio between actual size and a representation of size.
`projection`	=	`Merkaartor`	Projection describes the method of representing the surface of a sphere or other shape on a plane.

`coordinates`	=	`45.123,-30.543,48.321,-29.345`	The Coordinates element refers to the geographical coordinates of the bounding box covered by the resource.

3.3.9.16 Classification

Tag		value examples	description
`classificationSchema`	=	`LaTeX-Liederbuch-Schema`	Classification is used to document the class number for the resource, according to a standard scheme like RDA, MARC 21, LCC, LCSH or DDC.

3.3.9.17 Location Info

Tag		value examples	description
`url`	=	`https://de.wikisource.org/wiki/Allgemeines_Deutsches_Kommersbuch:122`	The URL field contains a URL that can be used to access the resource.
`bibliographicCitation`	=	`Gaudeamus igitur; Allgemeines Deutsches Kommersbuch, Seite 270; Friedrich Silcher, Friedrich Erk; Moritz Schauenburg; 1858; Lahr`	The Bibliographic Citation field contains the bibliographic citation for the resource.
`shelfLocation`	=	`3AC/F-K`	Shelf Location is used to record the shelf designation number for the resource.
`physicalLocation`	=	`Heimatmuseum, Kudamm 31, Berlin; Building 3/A; 3. Floor`	Physical Location refers to the institution or repository where the resource is held.

3.3.10 How it works

3.3.10.1 Building a Songbook

The first part contains only a some constants. Currently only the `\repeatleft` and `\repeatright`, which contain the repeat sign meant to be used in strophes:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Special music characters %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[...]
```

Handling of meta data:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Handling of the meta data %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\def\GFM@LB@unpackage...
```

This command stores the meta data to command sequences. The meta data items can be given as an argument in the Lied environment. `\GFM@LB@unpackage` loops over the meta data argument of the `Lied`- and `Liederbuch`-environment. For every pair of variable name and value, `\GFM@LB@unpackage` creates a LaTeXvariable with the value content. Each meta data ist stored to a command sequence in the format:

```
\GFM@LB@lied@<songbook>@<song variant>@<song number>@<meta data name>
```

For the songbook ”Folksongs”, songnumber ”5”, variant ”choral” and variable name ”composer”, the LaTeXvariable would look like this:

```
\GFM@LB@lied@Folksongs@choral@5@composer
```

It could expand to ”Frederic McCormick” (assuming this is the composer). This mechanism is only internally relevant. To use this variable fine and easily, there is the `\print{<variable name>}` command (see farther down).

Since the equals sign is not available in the meta data, it is made available again with the `\equals` command, which just expands to an equals sign.

If a song has wrong meta data, an entry is missing or any other reason, you can edit specific meta data entries. The better option is to edit the song book. But if this may not be possible for whatever reason, you can use this command:

```
\tweakMetadata{<songbook>}{<song number>}{<variant>}{<meta data}
```

One of the most important core parts of the package is the arrangement of syllables below the notes:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Environment and commands for        %%%
%%% notes + lyrics in native LaTeX font %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[...Constants...]

\def\songLyrics[<length> default = \linewdith]{
    [...]
    \def\nspace[*]{<some length or number>}
    \def\ns[*][<some length or number>]
    [...]
}%
```

`\songLyrics` creates a `\makebox[<length>]{notes}` with two commands inside: `\ns[*]` and `\nspace[*]`. `\nspace[*]` behaves exactly the same as `\hspace[*]` with the option for starred command, but unlike `\hspace` it also accepts no dimension unit. If no unit is given, it is translated to percent `\linewidth`. `\ns` behaves like `\hfill` and is able to shift the next sillable by an optional offset. The star inserts a dash between the sillables. i.e. `\ns*[-3]` separates two sillables with a dash in between und shifts the second sillable `0.03\linewidth` to the left:

```
{See \ns here \ns the \ns be \ns* gin \ns* ning}
```

becomes

See here the be-gin-ning

`\notenzeile` is a user command and combines the score image (notation) and the song lyrics. It takes two arguments. One graphic (the notes) and text below the notes. The text below simply calls `\GFM@LB@songLyrics`. As there are songs, that have more than one strophe below the notes, up to 6 are optional. It is 6 times the same. To hold everything together, the whole `\notenzeile` is inside a minipage:

```
\notenzeile{O{\linewidth} m m O{\empty} O{\empty} O{\empty} O{\empty} O{\empty}}{%

\multiline}{m m m O{\empty} O{\empty} O{\empty} O{\empty}}{%
```

`\multiline` can fit more than one line into one line. It is used, if only a part of a line has more than one text line. It is basically a few `\makebox`es stacked above each other inside a parbox.

Force page break is a very usefull function and pretty simple. It defines a list of numbers, if the current stave line / strophe equals one of the numbers a page break is added:

```
\forcePageBreakAfterStaveLine{<number>}[<n.>][<n.>][<n
   .>][<n.>][<n.>]

\forcePageBreakAfterStrophe{<number>}[<n.>][<n.>][<n.>][<n
   .>][<n.>]
```

Up to 6 page breaks within one song are possible. That should be enough as it is highly unlikely, that a song spans over more than 6 pages. If it does, please open an issue and show me. I'm highly interested.

Strophe is a simple environment that provides a numbering at the beginning based on a `\stepcounter`. It groups mostly the spacing and holds the strophes together:

```
\begin{strophe}[<set strophe number to x>]
This is the strophe text.
Sing sing sing.
Line breaks disappear.
I want to create an option for that.
\end{strophe}
```

The spacing constists of set-commands, stored values and the use commands (no prefix/suffix):

```
%%%%%%%%%%%%%%
%%% Spacing %%%
%%%%%%%%%%%%%%

%%% Head
\def\setSpaceBeforeHeadValue#1{\def\spaceBeforeHeadValue
   {#1}}
\def\setSpaceAfterHeadValue#1{\def\spaceAfterHeadValue
   {#1}}

%%% Notes
\def\setSpaceBeforeNotesValue#1{\def\spaceBeforeNotesValue
   {#1}}
\def\setSpaceAfterNotesValue#1{\def\spaceAfterNotesValue
   {#1}}

%%% Strophes
```

```
\def\setSpaceBeforeStropheValue#1{\def\
   spaceBeforeStropheValue{#1}}
\def\setSpaceAfterStropheValue#1{\def\
   spaceAfterStropheValue{#1}}

%%% Song lyrics
\def\setSpaceBeforeSongLyricsValue#1{\def\
   spaceBeforeSongLyricsValue{#1}}
\def\setSpaceBetweenSongLyricsValue#1{\def\
   spaceBetweenSongLyricsValue{#1}}
\def\setSpaceAfterSongLyricsValue#1{\def\
   spaceAfterSongLyricsValue{#1}}

%%% Foot
\def\setSpaceBeforeFootValue#1{\def\spaceBeforeFootValue
   {#1}}
\def\setSpaceAfterFootValue#1{\def\spaceAfterFootValue
   {#1}}

%%% Horizontal spacing
\def\setSpaceStropheIndentValue#1{\def\
   spaceStropheIndentValue{#1}}
\def\setSpaceHeadIndentValue#1{\def\spaceHeadIndentValue
   {#1}}
\def\setSpaceFootIndentValue#1{\def\spaceFootIndentValue
   {#1}}
```

Convert the space values into skips:

```
%%% Arranging the space values
\def\spaceBeforeHead{\vskip\spaceBeforeHeadValue\relax\
   noindent\mbox{}\hskip\spaceHeadIndentValue\mbox{}}

\def\spaceAfterHead{\relax\vskip\spaceAfterHeadValue\relax
   }

\def\spaceBeforeNotes{\relax\vskip\spaceBeforeNotesValue\
   relax}

[...rest works analogously...]
```

These commands save the definition of the header and footer for each song. They also implicitly arrange the spacing in the header and footer:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Styling of header and footer %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\newcommand{\LBHead}[1]{%
\long\def\GFM@LB@Head{\spaceBeforeHead #1\relax\
  spaceAfterHead\mbox{}}%
}%

\newcommand{\LBFoot}[1]{%
\long\def\GFM@LB@Foot{\spaceBeforeFoot #1\relax\
  spaceAfterFoot}%
}%
```

`Liederbuch` and `Lied` are the environments that create the songbooks and store each song in a command sequence. Every song is stored this way in Latex:

```
\liedBody;<songbook>;<song variant>;<number>
```

The `Lied` environment also stores each meta data value in an command sequence by calling the `\GFM@LB@unpackage` command:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Environments for creating  %%%
%%% the songbooks (liederbuch) %%%
%%% and the songs (lied)       %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Liederbuch environment
\NewEnviron{liederbuch}[2][]{%
\def\GFM@LB@LiederbuchNameTmp{#2}%
%%Lied environment %cslet
\NewEnviron{lied}[3][]{%
%##1=meta data ##2=variant ##3=nummer
\def\GFM@LB@LiederbuchVariantTmp{##2}%
\def\GFM@LB@LiederbuchNumberTmp{##3}%
\GFM@LB@unpackage{##1}%
\GFM@LB@unpackage{#1}%
\global\cslet{liedBody;\GFM@LB@LiederbuchNameTmp;##2;##3}\
  BODY%
}%
\BODY% Without this, the command sequences inside the
  environment won't be defined. That means, the whole
```

```
    content of the songbook (Liederbuch) is thrown away.
}
```

3.3.10.2 Using a Songbook

With the command

```
\usepackage{../path/to/songbook.sty}
```

you include a songbook. This automatically stores the content of the songs at loading time as described in the previous section. Now every song body is stored in a LaTeX macro. Each meta data element is also stored in a LaTeX macro. We need this later. When you call a song with the command:

```
\LBsong{<songbook name>}{<song number>}{<variant>}
```

i.e.

```
\LBsong{exampleSongs}{23}{nt}
```

It calls itself a series of macros:

```
%<setting some values>
\NewDocumentCommand{\print}{O{\empty} m}{%
    [...]
}%
\GFM@LB@Head%
%<setting some values>
%%%%%%%%%%%%%%%%%%%%%%%%%%%
\csuse{liedBody;#2;#1;#3}%
%=\liedBody;<songbook>;<song variant>;<number>
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%<setting some values>
\GFM@LB@Foot
```

As you see, it basically puts together head, body and foot. `\GFM@LB@Head` and `\GFM@LB@Foot` only contain the format of the head and foot. Inside these commands, the parameters songbook, song variant and number are available, therefore the correct meta data, like the correct title, author, etc. is automatically selected, when the `\print` command is used (see the example on page 18 and also 3.3.10.3 Themes on the following page)

3.3.10.3 Themes

For creating a theme, you can currently only define the style of the head and the foot of a song.

The principle is very simple. In

```
\LBHead
```

and

```
\LBFoot
```

a format for the head is defined and stored (with some spacing) into

```
\GFM@LB@Head
```

For the foot it is stored into

```
\GFM@LB@Foot
```

Inside them the `\print{<meta data name>}` is used. It is not defined there and expanding it would result in an undefined-error. But it is defined inside `\LBsong`, where `\GFM@LB@Head` and `\GFM@LB@Foot` are called. There it is expanded. Inside the `\LBsong` songbook, song variant and song number are available and put together to the command sequence, that expands to the correct meta data value (see section 3.3.10.2 Using a Songbook). The rest of the head and foot definitions are usual TEXcommands.

For the future, there shall be a marker, if a song had been broken inside the notes or inside the strophas. Depending on the marker, background images can be selected. But this is only an idea yet.

3.3.10.4 printliederbuch

The principle of this package is very simple. Before the songbook is loaded, the environments `Liederbuch` and `Lied` are redefined. The whole meta data is put into a table, the content of the songbooks is just put into a command sequence, which is called in the document.

4 Liederheft-Class

This is hardly finished yet. It shall be based on the koma-book class, provide a class itself, but I think, this is a bad idea anyway and I will ditch it. The plan exists, let's see, for how long.

4.1 Basics

No text yet. Just guess it.

4.2 What The Package Uses

- scrbook
  This is the class, which the Liederheft is based on.

4.3 User Commands

Just read the manual. Oh, this is the manual. How unfortunate.

4.4 How it works

Not at all yet. Maybe it never will.

5 Known bugs and problems

5.1 Undefined control sequence $\backslash OT$

undefined control sequence `\OT` maybe caused by ß in meta data. A simple workaround is to use `\def\OT#11{}` in the preamble.

5.2 Undefined control sequence $\backslash GFM@LB@songbook@123@nt@title$

This happens, if a song has no title specified in the metadata, but the listofsongs is used. Everything will still work and the error can be ignored. But the list of songs will not look like such a list is supposed to look like. You can use `tweakMetaData` to give this song a title. Or you can give it a title in the source. Alternatively you can deactivate the list of songs.

5.3 Spacing

If there is used plus and/or minus in spacing, this (might) result in extra space. The origin is unknown yet.

6 Bug reports

Nothing to say here. You can post any bug reports on bitbucket, if you find some:

`https://bitbucket.org/maestro-glanz/liederbuch-package/src/main/`

If you don't have a bitbucket account and don't want to create one, you can post an email to textinkerer.1904@gmail.com and hope, that I read it within 3 month. Note: Copying the mail address will fail. This is for spam precautions. You have to type it off your screen. This is a good memory exercise to keep your mind vital a flexible.

7 Revision History

2017-11-01: v0.1.0

2024-12-22: v1.0.0

2025-01-20: v1.1.0