The Song♭ook Package

Version 4.5

Christopher Rath
<Christopher@Rath.ca>

2010/04/30

Abstract

This package provides an all purpose songbook style for LaTeX2e. The package allows for three types of output from a single input file: words and chords books for the musicians to play from, words only songbooks for the congregation to sing from, and overhead transparency masters for congregational use. The style will also print a table of contents, an index sorted by title and first line, and an index sorted by key. It attempts to handle songs in multiple keys, as well as songs in multiple languages.

Contents

Preface to version 4.5

What's new in version 4.5:

- Added a `compactsong` option to the `song` environment; which allows per-song use of `compactsong` formatting.

Preface to version 4.4

What's new in version 4.4:

- there are no functional changes
- at Donald Arseneau's request, added language to state that conditionals.sty is public domain

Preface to version 4.3

What's new in version 4.3:

- found a problem with the optional parameter to the xlatn environment added in Release 4.2; that optional parameter has been removed
- a new songTranslation environment added to provide the required capability for the Artist index option
- added a new song to the sample songbook to act as a test case for translation

Preface to version 4.2

What's new in version 4.2:

- added a new Artist index option
- added a new optional parameter to the xlatn environment; in support of the new Artist index option

Preface to version 4.1a

What's new in version 4.1a:

- Corrected a bug whereby the new exclude song mode was throwing an error when either the `\SBRef` or `\SBMargNote` commands were used

Preface to version 4.1

What's new in version 4.1:

- a new optional ⟨*Include?*⟩ parameter has been added to the `song` environment; that parameter allows you to have a song omitted from the printed songbook yet still have the song counter incremented and the song's table of contents entry written to a separate TOC file (see the description of the `song` environment, below, for more details)
- to go along with the new optional ⟨*Include?*⟩ parameter is a new `\usepackage{}` option, `printallsongs`, which overrides the individual song option declarations and prints all the songs in the songbook
- the song "My Sun and My Shield" was removed from the sample songbook; it turns out that this is a Ted Sandquist song and is not in the public domain
- `chordbk`'s `compactsong` option is still experimental

Preface to version 4.0

What's new in version 4.0:

- the Song♭ook style has now completed its transition to LaTeX2e (I *think*): there is now a single `.sty` file which accepts options in order to invoke the different songbook styles. The Song♭ook style now also accepts and produces reasonable output for all of LaTeX2e's standard papersize options.
- the song title block (where the title, copyright info., etc. are listed) has been changed, use of the `\centerline` macro has been replaced with a `center` environment. This change is *not compatible* with previous versions of `songbook.sty` and requires you to re-verify all page breaks (mostly in words-only mode). The reason for making the change is to allow long song titles to line-wrap (instead of hanging off the edge of the page), and this means that the definition of the following macros has been changed: `\STitle`, `\CpyRt`, `\WAndM`, and `\ScriptRef`. The centering of these lines is now also done within a center environment; in each the centering may now be disabled by adding an optional first parameter (any value except '`Y`')
- since the change to the title block invalidated pagination of the previous version I have taken the opportunity to fine tune the value of `\SpaceAfterSong`, a value that is used primarily in words-only mode: the inter-song gap has been decreased to `\vspace{0ex plus10ex minus3ex}` (from `\vspace{0ex plus15ex minus0ex}`)
- a new space command, `\SpaceAfterTitleBlk`, has been created to allow the space between a song's title block and its versicles to be tuned by the user; this was a previously hardcoded value
- a bug in the `SBBracket` environment has been corrected: long lines were not always exhibiting their hanging indentation
- the style now supports all of LaTeX2e's standard papersizes. While the output will not be ideal for all papersizes, it does produce sane and usable results for all papersizes. I would be most appreciative if European users would send me page layout corrections for the A4, A5, and B5 sizes
- added a new environment, `SBOpGroup` (i.e., "an open group"), serves to group the lines of a verse or chorus together, but not indent or label them. Use of this environment allows for better control of font changes, proper indentation of wrapped lines, and automatic spacing of open groups which follow one another. I strongly suggest that `SBOpGroup` be used to enclose any set of lines which don't otherwise end up in one of the songbook environments when typsetting with this package
- `chordbk` mode now supports one variation: `compactsong`. In `compactsong` mode the songs are laid out in two columns; note that the song title block spans the two columns. The songs are set in a smaller typeface to allow them to fit into the smaller space two column mode supplies. This mode should be considered experimental for the present time; see its description, below, for more details
- `conditionals.sty` has been updated with more current information and macros, as supplied by Donald Arseneau
- all of the verse-like environments now have their `\baselineskip` amount expressly calculated just prior to laying out their lines. This has been done in order to overcome the problem all previous versions of the songbook style had which was that linespacing differed based upon whether a particular line

contained chords. Now all lines are spaced the same, regardless of whether they contain a chord. I consider the previous behaviour—where linespacing varied—to be a bug. If you really must retain the old behaviour this can be done by inserting the following code into the preamble of your document:

```
\renewcommand{\sbSetsbBaselineSkipAmt}
  {\setlength{\sbBaselineSkipAmt} {\baselineskip}}
```

- the `\SBDefaultFont` command no longer needs to be specified at the top of each songbook
- fixed a bug that was inhibiting the Song♭ook style from detecting blank and empty `song` parameters
- the commands which had previously been listed as deprecated (i.e., to be removed in some future release) have all been removed
- the following commands have been moved from the "Obsolete Macros" section into "Deprecated Macros" section and will be removed in the next major release of the Song♭ook style: `\False`, `\True`, `\ChordBk`, `\Overhead`, `\SongEject`, `\WordBk`, and `\WordsOnly`

A few minor changes were made during release testing of version 4.0. The following changes occured between version 4.0pre2 and 4.0:

- the spacing around the `SBBracket` environment has been *tuned*: `\SpaceAfterSBBracket` has been increased, and a new `\SpaceBeforeSBBracket` amount has been added
- added missing space around the `SBBracket*` environment; using `\SpaceBeforeSBBracket` and `\SpaceAfterSBBracket`
- removed unused length, `\SBBracketHangAmt`
- added a new `\LeftMarginSBBracket` length and rewrote the part of the `SBBracket` environment that creates the tag and left indents the versicle. The `SBBracket` environment now left aligns its words with those of the `SBVerse` and `SBChorus` versicles.

Part I

High Level Documentation

1 Description

The Song♭ook document style provides a core set of functions for the production of songbooks. Three pre-defined songbook formats and one variation are provided (and they are invoked via options to the `\usepackage{songbook}` command) and they are typically used along with LaTeX's `book` class. One of the following options *must* be specified or the Song♭ook style will throw an error: `chordbk`, `wordbk`, or `overhead`.

An empty minimal songbook looks like the following:

```
\documentclass{book}
\usepackage[chordbk]{songbook}

\begin{document}
  \begin{song}{}{}{}{}{}{}
  \end{song}
\end{document}
```

We'll start by explaining the `\usepackage[]{songbook}` options:

`chordbk`

`chordbk` a songbook suitable for musicians which gives both lyrics and words (this is the default mode of the Song♭ook document style)—one variation to this style is offered, compact song mode (see below). This option is specified as `\usepackage[chordbk]{songbook}`

`wordbk`

`wordbk` a words-only songbook suitable for mass distribution to those singing but not playing an instrument. This option is specified as `\usepackage[wordbk]{songbook}`

`overhead`

`overhead` to produce overhead transparencies from songbook source files. This option is specified as `\usepackage[overhead]{songbook}`

Other additional options supported by the Song♭ook style include all LaTeX's standard papersize options, and:

`compactsong`

`compactsong` this option only takes effect along with `chordbk`. It causes the songs to be set in two columns, where the song title information spans the both columns. It is specified as `\usepackage[chordbk,compactsong]{songbook}`

The version of `compactsong` provided in this release should be considered experimental! The formatting produced in this mode is not always desirable. An outstanding question to be answered is whether or not new songs title blocks should span both columns, and whether each song should generate a page break; in other words, should this feature set be implemented as two pieces: compactsong and compactbook. The idea would be to provide a compactsong environment, which could be judiciously used on a per song basis, and a compactbook mode which would result in a compressed songbook, where the words and chords book would look very much like a words only songbook (but with chords).

`printallsongs`

`printallsongs` this option causes all songs in a songbook to be printed, regardless of what ⟨*Include?*⟩ option may have been specified on each individual `song` environment

2 Commands

This section is broken into several subsections. Hopefully this makes the individual commands easier to understand by placing them in a meaningful context. Since some forward references exist, it may be necessary to read through the entire *Commands* section a couple of times before it makes complete sense.

This reference section will present terse command and environment descriptions; more detailed descriptions, along with examples, may be found in the implementation detail section at the bottom of this document.

Note that each subsection's descriptions are presented in alphabetical order; while this doesn't make the sections quite as easy to read, it makes them much more useful for reference purposes.

2.1 Environments

The Song♭ook style defines several new environments to make the formatting of songbooks easier and more consistent (and most of them have parameters). Unless otherwise noted, all of the environments are `verse`-like: wrapped lines are indented more than the first line is indented.

`\begin{SBBracket}{`⟨*bracket tag*⟩`}`⟨*. . . stuff to enbracket. . .*⟩

`SBBracket`

`\end{SBBracket}` is the environment used to mark certain lines of the song with a tag and bracket. An example usage is to mark the line of the song played to end the piece, if it is somehow different than the chords played if one were to repeat the song. For example:

```
   Be\Ch{Am}{cause} of \Ch{Dm7}{what} the...
\end{SBChorus}

\begin{SBBracket}{Ending}
   Give \Ch{F}{thanks,}\Ch{C/F}{} \Ch{Bb/F}{}...
\end{SBBracket}
```

This is very similar to the `SBOccurs` environment, the difference being how the section of the song is marked.

`SBBracket*` There are two versions of this environment: `SBBracket` and `SBBracket*`. They operate identically, except that the `*`ed version doesn't print its tag and bracket in words-only modes.

At present, `\SBBracket` and `\SBBracket*` are fragile and are not compatible with `SBVerse`, `SBChorus`, or any other environment; with the exception of the `song` environment.

`SBChorus` `\begin{SBChorus}`⟨*. . . the chorus. . .*⟩`\end{SBChorus}` is the environment to wrap around a chorus that you wish to be indented and given a chorus tag ("Ch:"). A song with one verse and one chorus, where the chorus is sung after the verse would probably use the `SBChorus` environment. Whereas, if the chorus was sung first, an `SBVerse` environment would probably be used. The indent amount for lines that are too long is set by redefining the `\HangAmt` command.

`SBChorus*` The `SBChorus*` version of this command indents but does not place a `\SBChorusTag` before the chorus.

`SBExtraKeys` `\begin{SBExtraKeys}{`⟨*song content*⟩`}\end{SBExtraKeys}` is the environment used when you wish to list the song again in another key. Typically, this environment is used along with an `\STitle` command. For example:

```
\begin{SBExtraKeys}{
   \STitle{You Alone}{D}

   \begin{SBVerse}
      \Ch{D}{Ho}\Ch{F#m}{ly,} \Ch{G}{Ho}\Ch{D}{ly,}
      ...
   \end{SBVerse}
}\end{SBExtraKeys}
```

`SBOccurs` `\begin{SBOccurs}{`⟨*the occurrence*⟩`}`⟨*. . . stuff to group. . .*⟩`\end{SBOccurs}` is the environment used to mark a given line of the song with a tag and brackets. For example "1,3" would designate that this passage applies to the 1st and 3rd occurances. For example:

```
   Be\Ch{Am}{cause} of \Ch{Dm7}{what} the...
\end{SBChorus}

\begin{SBOccurs}{1,3}
   Give \Ch{F}{thanks,}\Ch{C/F}{} \Ch{Bb/F}{}...
\end{SBOccurs}
```

`SBOpGroup` `\begin{SBOpGroup}`⟨*. . . stuff to group. . .*⟩`\end{SBOpGroup}` is the environment in which unmarked versicles are placed; so called "open groups".

`SBSection` `\begin{SBSection}`⟨*. . . the section. . .* ⟩`\end{SBSection}` is very much like LaTeX's verse environment, except that here the sections are numbered. The indent amount for lines that are too long is set using the `\HangAmt` command. This environment would be used in place of the `\SBVerse` environment for songs which are broken into pieces/sections, in place of, or in addition to, verses.

`SBSection*` The `SBSection*` version of this command indents but doesn't place an `\SBSectionCnt` before the chorus. Similar to LaTeX's `\section*` command, the section counter is not incremented either.

`SBVerse` `\begin{SBVerse}`⟨*. . . the chorus. . .* ⟩`\end{SBVerse}` is the environment to wrap around a verse that you wish to be indented and given a verse number (`\SBVerseCnt`). A song with one chorus and one verse, where the verse is sung after the verse would probably use the `SBChorus` environment. Whereas, if the chorus was sung first, an `SBVerse` environment would probably be used. The indent amount for lines that are too long is set with the `\HangAmt` command.

`SBVerse*` The `SBVerse*` version of this environment indents but down not place an `\SBVerseCnt` before the chorus; similar to LaTeX's `\section*` command, the verse counter is not incremented either.

`song` `\begin{song}[`⟨*1*⟩`]{`⟨*2*⟩`}` . . . `{`⟨*7*⟩`}` ⟨*. . . the song. . .* ⟩`\end{song}` is the environment which each song resides within. The parameter list is quite long, and is defined as:

1. Optional format string (Include song? / Compact song mode?);
2. Song title;
3. Key song is written in;
4. Copyright information;
5. Name(s) of composer and lyricist;
6. Scripture reference for the song;
7. Copyright licensing information.

The `song` environment takes care of making index entries, incrementing `\SBSongCnt` and page generation (if necessary). Note, this environment makes use of `\everypar`. See the *Example* section, below, for a sample one-song songbook document.

The optional format string parameter allows per-song control of certain typesetting attributes. Each of the attributes is optional, and a single attribute, or multiple attributes, may be used. The available values are:

`Y` or `N` These two characters tell Song♭ook whether to include the song in the songbook. This "Include this song?" option is referred to within this documentation as "⟨*Include?*⟩". If you don't specify a value (and you typically will not), then it behaves as though you provided a value of "`Y`". When a value of "`N`" then the song is excluded from the current songbook; however, a table of contents record is written to a separate file (*jobname*`.tocS`).

`C` or `F` These two characters tell Song♭ook whether the song should be presented in `compactsong` mode or full size presention mode (`chordbk` formatting only).

`\CBExcl` `\OHExcl` `\WBExcl` `\WOExcl` Some predefined macros have been provided which allow conditional exclusion of a song (they are used in the optional parameter): `\CBExcl`, `\OHExcl`, `\WBExcl`, and `\WOExcl`; respectively, these correspond to exclude in `chordbk` mode, `overhead` mode, `wordbk` mode, and when in words-only (i.e., not in `chordbk`) mode.

As an organisation's songbook grows, and time passes, it is not uncommon for the songbook to become overly large. The ⟨*Include?*⟩ parameter allows for a songbook's songs to be easily removed and re-added, without requiring old songbooks to be destroyed or overhead transparencies renumbered.

When the "copyright information" or "composer & lyricist" parameters are left empty then the string defined by the `\SBUnknownTag` macro used (instead of leaving whitespace in the song header.

`songTranslation` `\begin{songTranslation}{`⟨*1*⟩`}` . . . `{`⟨*4*⟩`}` ⟨*. . . the translation. . .*⟩`\end{songTranslation}`is the new song translation environment. The parameter list is defined as:

1. Translation language;
2. Translated song title (in the foreign language);
3. Translation permission;
4. Who performed the translation.

The `songTranslation` environment always occurs within a `song` environment; it resets the verse counter, causes the title and other parameter information to be displayed, and makes the appropriate index and table of contents entries. It is important for the `songTranslation` environment to occur within a song environment, because the `songTranslation` environment inherits the song environment's `\everypar` definition.

`xlatn` `\begin{xlatn}{`⟨*1*⟩`}` . . . `{`⟨*3*⟩`}` ⟨*. . . the translation. . .*⟩`\end{xlatn}`is the old song translation environment—this environment is considered obsolete and will be removed in a future relase of the Song♭ook macros; it has been replaced by the `songTranslation` environment. The parameter list is defined as:

1. Translated song title (in the foreign language);
2. Translation permission;
3. Who performed the translation.

The `xlatn` environment always occurs within a `song` environment; it resets the verse counter, causes the title and other parameter information to be displayed, and makes the appropriate index and table of contents entries. It is important for the `xlatn` environment to occur within a song environment, because the `xlatn` environment inherits the song environment's `\everypar` definition.

2.2 Primary Song♭ook Macros

Along with the Song♭ook environments, these are the macros you will most often use when constructing a songbook (of any style).

`\CBPageBrk` `\CBPageBrk` forces a new page if `\ifChordBk` is true.

`\Ch` `\Ch{`⟨*chord*⟩`}{`⟨*syllable*⟩`}` the chord over lyrics command definition. This is the most commonly used command in the Song♭ook style. The words-only substyle turns off the chord generation and just prints the second parameter. The ⟨*chord*⟩ parameter is left-justified over the ⟨*syllable*⟩ parameter. Any '#' or 'b' characters in the ⟨*chord*⟩ parameter are replaced with '♯' and '♭' characters, respectively. Also, if a bass note is specified in a chord (by way of a '/' character followed by the note) then it will appear in a smaller font than the rest of the ⟨*chord*⟩.

It is often desireable to typeset a chord—or set of chords—inside square brackets, to indicate that they are optional. A lighter weight font is probably desired, so that the brackets do not detract from the chord name, so any '[' and ']' characters are typeset with the font specified by the `\ChBkFont` macro.

To set the chord raise amount to a value that matches version 1.x and 2.x releases of the Song♭ook style, insert the following command into the preamble of your document:

```
\renewcommand{\SBChordRaise}{\SBOldChordRaise}
```

`\Chr` `\Chr{`⟨*chord*⟩`}{`⟨*syllable*⟩`}` this command performs the same function as the `\Ch` command with one exception: the `\Chr` command inserts a rule, at the height specified by the `\SBRuleRaiseAmount` macro, when the chord is wider than the syllable. The default value creates an extended em-dash-like rule; a value of 0pt creates an underbar-like rule. See the *Usage Guidelines* section of this document, below, for a more detailed explanation.

`\ChX` `\ChX{`⟨*chord*⟩`}{`⟨*syllable*⟩`}` this command performs the same function as the `\Ch` command with one exception: the `\ChX` command causes spaces trailing the command to be ignored. See the *Usage Guidelines* section of this document, below, for a more detailed explanation.

`\CSColBrk` `\CSColBrk` generates a column break here if we're in `compactsong` mode.

`\makeArtistIndex` `\makeArtistIndex` starts creation of an index of songs by artist (composer). If you need to add your own information to this index use the `\artistIndex[][]` command, documented in the *Detailed Documentation* section, below.

`\makeKeyIndex` `\makeKeyIndex` starts creation of an index of songs by key. If you need to add your own information to this index use the `\keyIndex[][]` command, documented in the *Detailed Documentation* section, below.

`\makeTitleContents` `\makeTitleContents` starts creation of a table of contents. If you need to add your own information to this index use the `\titleContents[][]` command, documented in the *Detailed Documentation* section, below.

`\makeTitleContentsSkip` `\makeTitleContentsSkip` starts creation of a table of contents of songs exluded from the current songbook. This macro operates in the same manner as `\makeTitleContents`.

`\makeTitleIndex` `\makeTitleIndex` starts creation of a title and first line index. If you need to add your own information to this index use the `\titleIndex[][]` command, documented in the *Detailed Documentation* section, below.

`\NotWOPageBrk` `\NotWOPageBrk` forces a new page if `\ifWordsOnly` is false.

`\OHContPgFtr` `\OHContPgFtr` prints a page heading continuation footer on overheads; this macro must be manually inserted where needed. `\OHContPgHdr` is a no-op, except when `\ifOverhead` is true.

`\OHContPgHdr` `\OHContPgHdr` prints a page heading continuation header on overheads; this macro must be manually inserted where needed. `\OHContPgHdr` is a no-op, except when `\ifOverhead` is true.

`\OHPageBrk` `\OHPageBrk` forces a new page if `\ifOverhead` is true.

`\SBBridge` `\SBBridge{`⟨*the bridge*⟩`}` is used to encapsulate a bridge: it causes ⟨*the bridge*⟩ to be set with `\SBBridgeTag`, using in the `\SBBridgeTagFont` font. In words-only mode this command is a no-op.

`\SBEnd` `\SBEnd[`⟨*use in words-only*⟩`]{`⟨*the ending*⟩`}` is used to encapsulate a song ending: it causes ⟨*the ending*⟩ to be set with the `\SBEndTag`, using in the `\SBEndTagFont` font. The first parameter is optional and if used is put in square brackets; specifying any value except 'N' will cause the ending to be used in words-only mode. Some examples of its intended use are:

This will cause the ending to be printed in words-only mode. Note how the parameter is specified in square brackets!

```
\SBEnd[Y]{Give \Ch{F}{thanks,} \ldots}
```

In this case the ending is a no-op in words-only mode.

```
\SBEnd{\Ch{A}{} \Ch{B/A}{} \Ch{D}{}}
```

`SBIntro` `\SBIntro[`⟨*use in words-only*⟩`]{`⟨*the introduction*⟩`}` is used to encapsulate any introduction to a song: it causes ⟨*the introduction*⟩ to be set with an intro tag of "Intro:", using in the `\SBIntroTagFont` font. The first parameter is optional and if used is put in square brackets; specifying any value except 'N' will cause the ending to be used in words-only mode. Some examples of its intended use are:

This will cause the ending to be printed in words-only mode. Note how the parameter is specified in square brackets!

```
\SBIntro[Y]{\Ch{D}{} \Ch{C}{} Ooooh}
```

In this case the ending is a no-op in words-only mode.

```
\SBIntro{{\SBLyricNoteFont Guitar and drums}}
```

`\SBMargNote` `\SBMargNote{`⟨*marginal note*⟩`}` is used to place a note of some kind in the margin of a songbook. In words-only mode this macro is a no-op.

`\SBRef` `\SBRef{`⟨*book title*⟩`}{`⟨*page or song number*⟩`}` creates a reference in the margin to another music book, or tape. This provides a method for directing people to resources they may use to learn the song. The marginal reference only prints when `\WordsOnly` is `\False`.

`\SBem` `\SBem` prints an *em-dash* (i.e., "—") when `\WordsOnly` is `\False.` See `\SBen`.

`\SBen` `\SBen` prints an *en-dash* (i.e., "–") when `\WordsOnly` is `\False.` This allows us to place a short rule within text in order place a chord earlier than a syllable; yet, that rule will not appear in the words-only book. The words-only version of this macro is a no-op. An example of its intended use is:

```
...flows like a ri\Ch{B/A}{\SBen ver,} flows...
```

`\STitle` `\STitle{`⟨*song title*⟩`}{`⟨*key*⟩`}` prints the ⟨*song title*⟩, preceded by the current `\SBSongCnt` value and followed by the ⟨*key*⟩ the song is given in. `\STitle` is most often used along with the `SBExtraKeys` environment. This command resets the `\SBVerseCnt` and `\SBSectionCnt` counters.

`\WBPageBrk` `\WBPageBrk` forces a new page if `\ifWordBk` is true.

`\WOPageBrk` `\WOPageBrk` forces a new page if `\ifWordsOnly` is true.

2.3 Miscellaneous Commands

Not all of the commands listed here are commonly used in songbooks written using one of the Song♭ook styles. The commands are listed alphabetically.

`\CpyRt` `\CpyRt{`⟨*copyright info.*⟩`}` prints the copyright information line. This command is not usually explicitly used in a songbook. It is called by the `song` environment and will normally only be used there.

`\FLineIdx` `\FLineIdx{`⟨*first line*⟩`}` make an entry in the *Title & First Line Index* file, "*`jobname`*`.tIdx`."

`\SBChorusMarkright` `\SBChorusMarkright` hook to allow `\SBSection`'s `\markright` to be overridden.

`\SBContinueMark` `\SBContinueMark` conditionally produce a continuation symbol. If the contents of `\rightmark` will result in nothing being typeset, then don't output the continuation mark; otherwise, output a continuation mark using the `\SBContinueTag` command.

`\SBSectionMarkright` `\SBSectionMarkright` hook to allow `\SBSection`'s `\markright` to be overridden.

`\SBVerseMarkright` `\SBVerseMarkright` hook to allow `\SBVerse`'s `\markright` to be overridden.

`\SongMarkboth` `\SongMarkboth` hook to allow the song environment's `\markboth` to be overridden.

`\STitleMarkboth` `\STitleMarkboth` hook to allow `\STitle`'s `\markboth` to be overridden.

`\ScriptRef` `\ScriptRef{`⟨*scripture address*⟩`}` is a scripture reference for the song. This command has its name because the Song♭ook style was written to produce songbooks for the church I am part of. This command is not usually explicitly used in a songbook. It is called by the `song` environment and will normally only be used there.

`\WAndM` `\WAndM{`⟨*lyricist & composer*⟩`}` prints a line telling who wrote the words and music for this song. The string "W&M:" precedes the listing of the ⟨*lyricist & composer*⟩ when it is printed. This command is not usually explicitly used in a songbook. It is called by the `song` environment and will normally only be used there.

2.4 Ifthen Commands

These `\if` tests are used to perform formatting that is dependent upon the type of songbook you are creating. It is these `\if` tests which allow a single source file to output the three songbook styles.

`\ifSBinSongEnv` `\ifSBinSongEnv` is true if we are inside of a song environment.

`\ifChordBk` `\ifChordBk` is true if we are processing a `chordbk` document.

`\ifOverhead` `\ifOverhead` is true if we are processing an `overhead` document.

`\ifWordBk` `\ifWordBk` are we processing a `wordbk` document?

`\ifWordsOnly` `\ifWordsOnly` is true when we are typesetting a words-only document (i.e., no chords).

`\ifNotWordsOnly` `\ifNotWordsOnly` is true if we are processing a document that displays chords.

`\ifCompactSongMode` `\ifCompactSongMode` is set to true if you want songs presented in a compact mode? It is initially set to false. Set this to true or false using the `\CompactSongModetrue` and `\CompactSongModefalse` commands, respectively.

`\ifSongEject` `\ifSongEject` is set to true if we want a new page generated at the end of every `song` environment? A value of true means eject after every `song` environment (default value is true).

Papersize tests have been provided in order to detect if a particular papersize has been specified. These are only documented in the *Detailed Documentation* section, below, since they are not generally needed.

2.5 Counters

These are the counters used in the various environments. Although you will generally not need to use them, they do sometimes come in handy; hence, they have been documented here.

`\theSBSongCnt` `\theSBSongCnt` counter is used for numbering the songs. When a song is listed multiple times (for multiple keys) the songs number must remain the same each time.

`\theSBSectionCnt` `\theSBSectionCnt` the section counter is used for numbering sections as they occur within a song.

`\theSBVerseCnt` `\theSBVerseCnt` the verse counter is used for numbering verses as they occur within a song.

2.6 Spacing Commands

These commands define the amount of space to leave in various situations. Change their values via LaTeX's `\renewcommand` command.

All of these spaces are defined as LaTeX commands to overcome limitations in LaTeX length evaluation. For example, if `\LeftMarginSBVerse` were to be defined as a length (i.e., using `\newlength`) and then immediately set to `4em`'s, the specific length would be evaluated with respect to the current font. This may not be what is desired; instead a length evaluated with respect to the font in place at the start of an `SBVerse` is probably what is desired. This can only be done by making these lengths LaTeX commands instead of lengths.

`\HangAmt` `\HangAmt` amount to indent when a line wraps.

`LeftMarginSBBracket` `\LeftMarginSBBracket` is the amount of left margin to leave when the `\SBBracket` environment is in effect.

`\LeftMarginSBChorus` `\LeftMarginSBChorus` is the amount of left margin to leave when the `\SBChorus` environment is in effect.

`LeftMarginSBSection` `\LeftMarginSBSection` is the amount of left margin to leave when the `\SBSection` environment is in effect.

`\LeftMarginSBVerse` `\LeftMarginSBVerse` is the amount of left margin to leave when the `\SBVerse` environment is in effect.

`\SBChordRaise` `\SBChordRaise` the distance to raise the chords above the baseline of the text they sit over.

`\SBRuleRaiseAmount` `\SBRuleRaiseAmount` the distance to raise the rule (as specified by `\SBIntersyllableRule`) which fills the space between adjoining syllables.

`\SpaceAboveSTitle` `\SpaceAboveSTitle` is the amount of vertical space left by the `STitle` command before it prints the song title line.

`\SpaceAfterTitleBlk` `\SpaceAfterTitleBlk` is the space inserted by the song environment between the *title block* and the versicles.

`\SpaceAfterChorus` `\SpaceAfterChorus` is the vertical space to leave after an `SBChorus`.

`\SpaceAfterOpGroup` `\SpaceAfterOpGroup` is the vertical space to leave after an `SBOpGroup`.

`\SpaceAfterSection` `\SpaceAfterSection` is the vertical space to leave after an `SBSection`.

`\SpaceAfterSBBracket` `\SpaceAfterSBBracket` is the vertical space to leave after an `SBBracket`.

`\SpaceAfterSong` `\SpaceAfterSong` is the vertical space to leave after a `song`.

`\SpaceAfterVerse` `\SpaceAfterVerse` is the vertical space to leave after an `SBVerse`.

`\SpaceBeforeSBBracket` `\SpaceBeforeSBBracket` is the vertical space to leave before an `SBBracket`.

It is worth noting that the `\SpaceAfterChorus`, `\SpaceAfterOpGroup`, `\SpaceAfterSection`, and `\SpaceAfterSong`, `\SpaceAfterVerse` macros all allow negative glue to be inserted; that is, the space may be shrunk as well as expanded. If this proves problematic (due to sections being visibly pushed into each other, the old spacing (as in versions 1.x and 2.x) can be restored by resetting these macros to 0ex. For example:

```
\renewcommand{\SpaceAfterChorus} {\vspace{0ex}}
\renewcommand{\SpaceAfterOpGroup}{\vspace{0ex}}
\renewcommand{\SpaceAfterSection}{\vspace{0ex}}
\renewcommand{\SpaceAfterSong}   {\vspace{0ex}}
\renewcommand{\SpaceAfterVerse}  {\vspace{0ex}}
```

2.7 String Constants

These constants are provided so that users may easily customize the appearance of formatted songs and songbooks. Use the `\renewcommand` command to change the value of these constants.

`\OHContPgFtrTag` `\OHContPgFtrTag` tag is inserted by the `\OHContPgFtr` command. The default value for this is “`continued on next page\ldots`”.

`\OHContPgHdrTag` `\OHContPgHdrTag` tag is inserted by the `\OHContPgHdr` command. The default value for this is “`\theSBSongCnt\ --- \theSongTitle, continued\ldots`”.

`\SBBaseLang` `\SBBaseLang` tag is the name of the language of all songs not specified within an `songTranslation` environment, and also as the default value of the `songTranslation` environment’s optional song language parameter. The default value for this is “`English`”.

`\SBBridgeTag` `\SBBridgeTag` the Bridge Tag to insert before the start of a bridge. The default value for this is “`Bridge:`”.

`\SBChorusTag` `\SBChorusTag` the Chorus Tag to insert before the first line of a chorus. The default value for this is “`Ch:`”.

`\SBContinueTag` `\SBContinueTag` the Continue Tag to insert in an `\SBContinueMark`. The default value for this is “`cont\ldots`”.

`\SBEndTag` `\SBEndTag` the End Tag to insert before the start of an ending (in an `\SBEnd` command). The default value for this is “`End:`”.

`\SBIntersyllableRule` `\SBIntersyllableRule` the command(s) to draw the rule between adjoining syllables.

`\SBIntroTag` `\SBIntroTag` the Intro Tag to insert before the start of an introduction (in an `\SBIntro` command). The default value for this is “`Intro:`”.

`\SBPubDom` `\SBPubDom` the string to insert which indicates song is in the public domain. The default value for this is “`Public Domain`”. If you want to localize this string in the song title block, be sure to use this public interface: the `\CpyRt` macro uses `\SBPubDom` to determine whether or not to print the copyright symbol (©).

`\SBUnknownTag` `\SBUnknownTag` the WAndM string to insert when either the author/artist or the copyright holder is unknown. The default value for this is “`Unknown`”.

`\SBWAndMTag` `\SBWAndMTag` the tag to insert before the words and music entry printed in the song header. The default value for this is “`W\&M:`”.

2.8 Font Handling

Of all the font selection Song♭ook macros, only one is commonly used by someone writing a songbook: `\SBLyricNoteFont`. All the other font macros are only used by an author to over-ride default behaviour, via the `\renewcommand` command.

`\ChBassFont` `\ChBassFont` sets the font for the bass note in chords as printed by the `\Ch`, `\Chr` and `\ChX` commands.

`\ChBkFont` `\ChBkFont` sets the font for square brackets typeset inside `\Ch` commands (and its variants).

`\ChFont` `\ChFont` sets the font for chords as printed by the `\Ch`, `\Chr`, and `\ChX` commands.

`\CpyRtFont` `\CpyRtFont` sets the font used to print the copyright line produced by the `\CpyRt` command.

`\CpyRtInfoFont` `\CpyRtInfoFont` sets the font used to print the ⟨*copyright licensing information*⟩ parameter of the `song` environment; which appears after the ⟨*copyright information*⟩ parameter under the ⟨*song title.*⟩

`\SBBracketTagFont` `\SBBracketTagFont` sets the font used to create the tag for an `SBBracket` environment.

`\SBBridgeTagFont` `\SBBridgeTagFont` sets the font used to create the tag for an `SBBridge` environment.

`\SBChorusTagFont` `\SBChorusTagFont` sets the font used to print the chorus tag, `\SBChorusTag`.

`\SBDefaultFont` `\SBDefaultFont` sets the default font for the songbook. As of version 4.0 there is no need for you to specify this command yourself.

`\SBEndTagFont` `\SBEndTagFont` sets the font used to print the tag, `\SBEndTag`, for the `\SBEnd` command.

`\SBIntroTagFont` `\SBIntroTagFont` sets the font used to print the introduction tag, `\SBIntroTag`.

`\SBLyricNoteFont` `\SBLyricNoteFont` sets the font used in comments placed within the lyrics giving musical direction. This is the only font command commonly used by the writer of a songbook.

`\SBMargNoteFont` `\SBMargNoteFont` sets the font used in the marginal reference printed by the `\SBMargNote` command.

`\SBOccursBrktFont` `\SBOccursBrktFont` sets the font used to create the large left and right square brackets which delimit an `SBOccurs` environment.

`\SBOccursTagFont` `\SBOccursTagFont` sets the font used to create the `\SBOccurs` tag.

`\SBRefFont` `\SBRefFont` sets the font used in the marginal reference printed by the `\SBRef` command.

`\SBVerseNumberFont`

`\SBVerseNumberFont` sets the font used to print the `\SBVerseCnt` in front of verses in an `SBVerse` environment.

`\SBSectionNumberFont`

`\SBSectionNumberFont` sets the font used to print the `\SBSectionCnt` in front of sections in an `SBSection` environment.

`\STitleFont`

`\STitleFont` sets the font used to print the song title, as generated by the `\STitle` command.

`\STitleKeyFont`

`\STitleKeyFont` sets the font used to print the key a song is written in, as generated by the `\STitle` command.

`\STitleNumberFont`

`\STitleNumberFont` sets the font used to print the `\SBSongCnt` in front of the song title, as generated by the `\STitle` command.

`\ScriptRefFont`

`\ScriptRefFont` sets the font used to print the scripture reference generated by the `\ScriptRef` command.

`\WandMFont`

`\WandMFont` sets the font used to print the lyricist and composer line generated by the `\WandM` command.

2.9 Deprecated Commands

The following commands will be discontinued in some future release of the Song♭ook style:

`\ChordBk` is set to `\True` if we're producing words and chord books. Set to `\False`, otherwise. Superceded by the `\ifChordBk` if.

`\False` is a constant used in TeX `\if` expressions. This command is now unnecessary.

`\Overhead` is set to `\True` if we're producing overhead transparencies. Set to `\False`, otherwise. Superceded by the `\ifOverhead` if.

`\SongEject` is a flag indicating whether or not the `\song` environment should end the current page when the environment ends: `\True` means end the page when the `\song` environment ends; `\False` means don't end the page. Superceded by the `\ifSongEject` if.

`\True` is a constant used in TeX `\if` expressions. This command is now unnecessary.

`\WordBk` is the flag which tells us whether we're producing a songbook with just words that is not a set of overhead masters. Superceded by the `\ifWordBk` if.

`\WordsOnly` is the flag which tells us whether we're producing a songbook with just words, or set of overhead masters. Superceded by the `\ifWordsOnly` if.

3 Usage Guidelines

This section gives some guidelines for use of the commands and environments offered by the Song♭ook style. These are not absolute standards, merely the suggestions that I have come up with after entering some 450 songs into a Song♭ook style based songbook. These guidelines rarely justify themselves, try things out and decide for yourself whether they're right or wrong.

1. Make each line of a song its own paragraph. This means that the songbook file is mostly double spaced. This allows the file to more easily survive encounters with users who edit the songbook source using a non-text-editor, such as WordPerfect.

2. Use of the `\Ch` command:

   - Always try to attach a chord to a single syllable. If you need to include more than one syllable with the chord then include extra text in units of syllables (whenever possible). For example:

     Do: `\Ch{G}{Halle}luia`

     Don't: `\Ch{G}{Hall}eluia`

   - Always include punctuation along with a syllable that has been included in a `\Ch` command. For example:

     Do: `\Ch{G}{Lord!}`

     Don't: `\Ch{G}{Lord}!`

   - Only place a single chord within a `\Ch` command. For example:

     Do: `\Ch{[}{}\Ch{G}{} \Ch{D}{}\Ch{]}{}`

     Don't: `\Ch{[G D]}{}`

3. Extension of syllables. Syllables may be extended at either/or both ends. Each end should be done in a different way:

   (a) One usually needs to make a syllable longer because the chord it is tied to is too long. This type of extension should be done using the `\Chr` command.

   Do: `\Chr{G\#m7/C}{Ho}\Ch{C}{ly}`

   Don't: `\Ch{G\#m7/C}{Ho\SBem}\Ch{C}{ly}`

   (b) Extending the beginning (i.e., delaying the start) of a syllable is generally required because the chord change needs to occur *between syllables.* For example, when the chord change is on the beat and the syllable is sung off-beat. Use `\SBen` and `\SBem` for this purpose.

   Do: `none Ho\Ch{D}{\SBen ly}`

4. Typographic conventions. LaTeX knows about certain ligatures; that is, it groups certain sequences of letters into a single character unit. `ff` is one of these ligatures and is typeset in a special way; however this cannot occur if the f's are split by a `\Ch` command. Therefore, if at all possible, never split up the following character sequences with the `\Ch` command: `ff, fi, ffi, fl, ffl`.

   Do: `\Ch{C}{diffi}cult`

   Don't: `\Ch{C}{dif}ficult`

5. Ordering of songs in the songbook. In order to allow LaTeX2e to fill pages in as natural a manner as possible, it is best to order the songs within the songbook based upon a `wordbk` formatted songbook. In that way, the words-only songbooks will contain optimally filled columns. Start by placing the longest songs first, only inserting shorter songs to cause page breaks at logical intervals.

6. Overheads that occupy more than one page. When in overhead mode, if a song spills over onto a second page (or beyond), it is helpful to print an extra header at the top of the page identifying which song the extra page belongs to. This is accomplished with the `\OHContPgHdr` macro. For example, one would insert the following lines where the new page is to occur:

   ```
   \OHContPgFtr
   \OHPageBrk
   \OHContPgHdr
   ```

4 Index/TOC Generation

The Song♭ook style provides facilities for title/first line index, song key index and table of contents generation. While this facility is not yet completely developed, it is much better than it was in early Song♭ook releases, and it produces *very* usable output!

4.1 Table of Contents Generation

Steps to follow in order to produce a table of contents:

1. Add a `\makeTitleContents` command to the preamble of your songbook.
2. Run LaTeX2e on the songbook source.
3. Make your own copy of `sampleToc.tex` and customize its header and footer definitions (so they match your songbook's). Then change the name of the file being `\input`ed to match your table of contents file.
4. Run LaTeX2e on your copy of `sampleToc.tex`.

4.2 Title & First Line Index Generation

Steps to follow in order to produce a title and first line index:

1. Add a `\makeTitleIndex` command to the preamble of your songbook.
2. Run LaTeX2e on the songbook source.
3. Run the `./mksbtdx` shell script on the `.tIdx` file that was produced by the previous step. Do this by typing "`mksbtdx` *jobname*" at a UNIX command line. For example, the index file for `sample-sb.tex` was produced by typing "`mksbtdx sample-sb`".
4. Make your own copy of `sampleTdx.tex` and customize its header and footer definitions (so they match your songbook's). Then change the name of the file being `\input`ed to match your index file. (`./mksbtdx` told you this file's name).
5. Run LaTeX2e on your copy of `sampleTdx.tex`.

4.3 Song Key Index Generation

Steps to follow in order to produce a song key index:

1. Add a `\makeKeyIndex` command to the preamble of your songbook.
2. Run LaTeX2e on the songbook source.
3. Run the `./mksbkdx` shell script on the `.kIdx` file that was produced by the previous step. Do this by typing "`mksbkdx` *jobname*" at a UNIX command line. For example, the key index file for `sample-sb.tex` was produced by typing "`mksbkdx sample-sb`".
4. Make your own copy of `sampleKdx.tex` and customize its header and footer definitions (so they match your songbook's). Then change the name of the file being `\input`ed to match your index file. (`./mksbkdx` told you this file's name).
5. Run LaTeX2e on your copy of `sampleKdx.tex`.

4.4 Song Artist Index Generation

To produce an index by song artist (composer) follow the same steps as for song key index generation, above, with the following exceptions:

- use `\makeArtistIndex` instead of `\makeKeyIndex`.
- use `./mksbadx` instead of `./mksbkdx`.
- use `sampleAdx.tex` instead of `sampleKdx.tex`.

5 Example

Here is an example songbook; where the the songbook contains exactly one song.

```
\documentstyle[12pt]{book}
\usepackage[chordbk]{songbook}            %% Words & Chords edition.

%%
% C.C.L.I. license number definition; for copyright licensing info.
%%
\newcommand{\CCLInumber}{\#999999}
\newcommand{\CCLIed}{(CCLI \CCLInumber)}
\newcommand{\NotCCLIed}{}
\newcommand{\PGranted}{}
\newcommand{\PPending}{(Permission Pending)}

%%
% Turn on index and table of contents.
%%
\makeTitleIndex       %% Title and First Line Index.
\makeTitleContents    %% Table of Contents.
\makeKeyIndex         %% Song Key Index.
\makeArtistIndex      %% Index by Artist.

\begin{document}
%%
% Songbook begins.
%%
\begin{song}{What A Mighty God We Serve}{C}
 {\SBPubDom}
 {\SBUnknownTag}
 {Isaiah 9:6}
 {\NotCCLIed}

  \renewcommand{\RevDate}{February~11,~1993}
  \SBRef{Give Thanks}{Hosanna! Music Tape HM-7}
  \SBRef{Hosanna! Music Book~I}{\#93}

  \begin{SBOpGroup}
    \Ch{C}{What} a mighty God we serve,

    What a mighty God we \Ch{G7}{serve},

    \Ch{C}{An}gels bow before Him,

    \Ch{C}{Hea}ven and earth adore Him,

    \Ch{C}{What} a mighty \Ch{G7}{God} we \Ch{C}{serve!}\Ch{[}{}\Ch{F}{}
      \Ch{C}{}\Ch{]}{}
  \end{SBOpGroup}

  \begin{SBVerse}
    O \Ch{C}{Zion,} O \Ch{F}{Zion,} that \Ch{G7}{bring}est good \Ch{C}{tid}ings,

    Get thee \Ch{F}{up} into the \Ch{G7}{High} Moun\Ch{C}{tains}

    Je\Ch{C}{ru}salem, Je\Ch{F}{ru}salem, that \Ch{G7}{bring}est good \Ch{C}{tid}ings
    Lift up thy \Ch{F}{voice} with \Ch{G7}{all} thy \Ch{C}{strength}

    Lift it \Ch{F}{up,} be not afraid;

    Lift it \Ch{C}{up,} be not afraid

    Say \Ch{Am}{unto} the \Ch{C}{ci}ties of \Ch{G7}{Judah,}

    ‘‘Behold your \Ch{C}{God,}\Ch{C7}{} Behold your \Ch{F}{God,}

    Be\Ch{C}{hold} \Ch{G7}{your} \Ch{C}{God!’’}
  \end{SBVerse}

\CBPageBrk
  \begin{SBExtraKeys}{%
    \STitle{What A Mighty God We Serve}{D}

    \begin{SBOpGroup}
      \Ch{D}{What} a mighty God we serve,

      What a mighty God we \Ch{A7}{serve},

      \Ch{D}{An}gels bow before Him,

      \Ch{D}{Hea}ven and earth adore Him,

      \Ch{D}{What} a mighty \Ch{A7}{God} we \Ch{D}{serve!}\Ch{[}{}\Ch{G}{}
        \Ch{D}{}\Ch{]}{}
    \end{SBOpGroup}

    \begin{SBVerse}
      O \Ch{D}{Zion,} O \Ch{G}{Zion,} that \Ch{A7}{bring}est good \Ch{D}{tid}ings,

      Get thee \Ch{G}{up} to into the \Ch{A7}{High} Moun\Ch{D}{tains}

      Je\Ch{D}{ru}salem, Je\Ch{G}{ru}salem, that \Ch{A7}{bring}est good
        \Ch{D}{tid}ings

      Lift up thy \Ch{G}{voice} with \Ch{A7}{all} thy \Ch{D}{strength}

      Lift it \Ch{G}{up} be not afraid,

      Lift it \Ch{D}{up} be not afraid

      Say \Ch{Bm}{unto} the \Ch{D}{ci}ties of \Ch{A7}{Judah,}

      ‘‘Behold your \Ch{D}{God,}\Ch{D7}{} Behold your \Ch{G}{God,}

      Be\Ch{D}{hold} \Ch{A7}{your} \Ch{D}{God!’’}
    \end{SBVerse}
  }\end{SBExtraKeys}
\end{song}
\end{document}
\bye
```

6 Dependencies

The Song♭ook style is dependent upon four other LaTeX2e styles: `conditionals.sty`, `calc.sty`, `ifthen.sty`, `multicol.sty`, and `xstring.sty`. `Conditionals.sty` is supplied with this package. `Calc.sty`, `ifthen.sty`, and `multicol.sty` are part of the LaTeX2e distribution. `xstring.sty` is available from CTAN.

Embedding guitar chord fingering charts within a songbook can be accomplished with the `texchord.sty` package; which is supplied in the contrib directory of the Song♭ook distribution.

7 Files

conditionals.sty Donald Arseneau's conditional tests; included with Donald's kind permission.

mksbadx A shell script around makeindex to sort the song artist index.

mksbkdx A shell script around makeindex to sort the song key index.

mksbtdx A shell script around makeindex to sort the title & first line index.

relnotes.txt The Song♭ook package release notes.

sample-sb.tex A sample songbook.

sampleAdx.tex Song artist index for the sample songbook.

sampleKdx.tex Song key index for the sample songbook.

sampleTdx.tex Title & first line index for the sample songbook.

sampleToc.tex TOC for the sample songbook.

songbook.ist The Song♭ook package makeindex `.ist` file.

songbook.dtx The base style file.

songbook.inx The install script used to create `songbook.sty`.

8 See Also

Some resources you will find helpful when coding songbooks:

- *LaTeX A Document Preparation System,* by Leslie Lamport
- *The LaTeX Companion,* by Goossens, Mittlebach, & Samarin
- The Song♭ook homepage, at URL `http://rath.ca/Misc/Songbook/`
- *The TeX book,* by Donald Knuth

8.1 Contributed Resources

A couple of Song♭ook users have created additional resources intended to be used with the Song♭ook style. If you have written anything which you would like to contribute to Song♭ook style's distribution, please let me know.

CarolBook a Song♭ook formatted book containing words for all the Christmas songs I've been able to find where the words are now in the public domain. PDF versions of the file are included for quick and easy use.

crd2sb a perl script which converts Chord files into Songbook files. Contributed by Abel Chow <`abel@g2networks.com`>. Note that a postscript formatter for Chord songs can be ftp'ed from: `ftp://ftp.uu.net/doc/music/guitar/resources/misc/CHORD/`.

modulate a perl script for modulating a song from one key to another. Contributed by Christopher Rath <`christopher@rath.ca`>.

LyX Integration files for use of the Song♭ookstyle with LyX. Christian Ridderström <`chr@md.kth.se`> has put together the necessary files to allow Song♭ooks to be edited using LyX. While these files are not distributed in the Song♭ook's `contrib` files, they are available from `http://www.md.kth.se/~chr/lyx/songbook/Songbook.shtml`.

texchord.sty LaTeX macros for printing guitar fingering charts. Contributed by Joel M. Hoffman <`joel@wam.umd.edu`>. Note, this style is *no longer* actively supported by Joel.

8.2 Other Similar Packages

There are a number of song and songbook formatting packages available which attempt to provide similar functionality to the Song♭ook package (although, IMHO, my package is better). Similar LaTeX2e packages (of which the author is aware) include:

chord.sty a song formatting package based on LaTeX's article style; written by Olivier Biot (`http://www.biot.yucom.be/`).

Chordpack a utility for typesetting *chordpro* chord files in TeX; written by Daniel Polansky (`http://www.fi.muni.cz/~xpolansk/home.html`) and available at `http://www.fi.muni.cz/~xpolansk/chordpack`.

gchords.sty a TeX packages for typesetting guitar chord diagrams; written by Kasper Peeters (`http://www.damtp.cam.ac.uk/user/kp229/`) and available at `http://www.damtp.cam.ac.uk/user/kp229/gchords/`.

Guitar.sty LaTeX macros for typesetting guitar chords over song texts; written by Martin Vth (`http://www.mathematik.uni-wuerzburg.de/~vaeth/`) and available from `http://www.mathematik.uni-wuerzburg.de/~vaeth/download/`.

GuitarTeX a graphical tool for editing *chordpro* chord files and printing them in TeX; written by Joachim Miltz and available from `http://www.rz-home.de/~jmiltz/guitartex/`.

song.sty a song formatting package based on LaTeX's book style; written by Jens T. Berger Thielemann (`http://www.stud.ifi.uio.no/~jensthi/`).

9 Bugs

In the specific case where a `\Ch`, `\Chr`, or `\ChX` macro begins a paragraph that isn't inside one of Song♭ook's versicle environments, that line may not indent properly in the `chordbk` substyle (specifically, a long, wrapped line won't have its extra indentation). I have been unable to identify the reason for the problem, although it is easily reproducible. The best way to avoid this problem is through use of the `\SBOpGroup` environment. If that isn't possible, the problem may often be overcome by starting such lines with an `\mbox{}` command; this inserts an empty (i.e., zero width) mbox at the start of the line. For example:

```
\mbox{}\Ch{G}{Great} is the Lord \Ch{A}{even} beyond the
  \Ch{D}{borders} of I\Chr{F#m}{srae}\Ch{Bm7}{l;}
```

The `\emph` macro is not completely compatible with `\Ch` and its friends. The specific problem is that sharps can not be specified via '#' within an `\emph` macro. The following snippet,

```
\emph{for the \Ch{G/A}{King} of \Ch{F#}{kings.}}
```

will fail with the LaTeX2e message,

```
! Illegal parameter number in definition of \\reserved\@a.
<to be read again>
```

The error message can be supressed by replacing '#' with '##', however this results in a double-sharp being typeset. The problem can be worked-around by replacing the snippet with:

```
\emph{for the \Ch{G/A}{King} of} \Ch{F#}{\emph{kings.}}
```

10 Special Thanks

Thanks to Donald Arseneau for writing the `conditionals.sty` file, and for helping write the `\Chord` macro. Donald, you are one of the faithful who is always quick to reply with correct answers to questions posted to `comp.text.tex`. Thanks again.

Thanks also to Philip Hirschhorn whose `\Chord` macro I ultimately used in versions 1.0–2.3 of the Song♭ook style, and to Olivier Boit who constructed a similar chord macro which I used to enhance Philip's code for version 3.0

A quick thank you to Herbert Martin Dietze <`herbert@fh-wedel.de`> for noting that `SBVerse*` and its cousins were missing from the `.sty` file, and then coding up an acceptable `SBVerse*` which I could quickly use as a model for the other two missing environments.

For version 4.1, I am grateful to Mark Wooding for suggesting the method I ultimately used for implementing the `song` environment's ⟨*Include?*⟩ option (although I did not use his preferred method).

I am grateful to Adam Fletcher for prodding me to add the per-song `compactsong` implementation; which took many years more than it should have taken for me to get it coded.

11 Author

Christopher Rath   `christopher@rath.ca`   (613) 824-4584
1371 Major Rd.
Orleans, ON
Canada   K1E 1H3

12 `.dtx` Documentation Driver

There is one last administrative detail to take care of before beginning the detailed review: the insertion of the documentation driver (i.e., the code that builds the documentation `.dvi` file.

```
⟨*driver⟩
\documentclass{ltxdoc} \RequirePackage{calc} \EnableCrossrefs
\CodelineIndex
\RecordChanges                  % Gather update information
 %\OnlyDescription % comment out for implementation details
 %\OldMakeindex % use if your MakeIndex is pre-v2.9
\setlength\hfuzz{15pt}  % dont make so many
\hbadness=7000          % over and under full box warnings
\def\MacroFont{\fontencoding\encodingdefault
   \fontfamily\ttdefault
   \fontseries\mddefault
   \fontshape\updefault
   \footnotesize}%

\voffset=-1.00in
\topmargin=0.5in
\headheight=0.0in
\headsep=0.20in
\textheight=9.4in
\footskip=0.4in

\newenvironment{ParameterList}
  {\par\hskip 1.5em Parameters:\begin{list}{}
    {\setlength{\topsep}{0pt}
      \setlength{\parsep}{0pt}
      \setlength{\itemsep}{0pt}
      \setlength{\leftmargin}{\leftmargin + 1.5em}
      \setlength{\parsep}{0pt}
    }
  }
  {\end{list}\vskip 0.5ex
  }
\newcommand{\parm}[1]{\texttt{[}\meta{#1}\texttt{]}}
\begin{document}
   \setcounter{IndexColumns}{1}
   \DocInput{songbook.dtx}
\end{document}
⟨/driver⟩
```

Part II
Detailed Documentation

This section contains style implementation details along with the detailed descriptions and documentation for the Song♭ook commands and environments. It is strongly recommended that these detailed descriptions be reviewed at least once as part of becoming familiar with the Song♭ook style.

This coding style has been structured in a top down fashion which assume that macros and environments must be declared before they are first used. TeX doesn't require this to be so, but since I've been coding software this way for 20+ years, it's easier for me to also maintain this structure here too.

13 Identification Part

The first section in `songbook.sty` is what LaTeX2e calls the *Implementation Part.* This is where Song♭ook identifies itself to the outside world. As part of this section an RCS "Id:" variable has been included as a TeX comment; the intent is that this may assist with reporting problems later.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                      %%
%%        I D E N T I F I C A T I O N   P A R T         %%
%%                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% rcsid = @(#)$Id: songbook.dtx,v 1.16 2010-04-12 18:10:15 rathc Exp $
%%
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{songbook}[2010/04/30 v4.5 All purpose Songbook style]
\typeout{Document Subclass: songbook 2010/04/30 v4.5 All purpose Songbook style}
```

14 Initial Code Part

The next section is called the *Initial Code Part.* This is where any dependencies in the early sections of `songbook.sty` has are contained. In the case of the Song♭ook style we must declare our dependence on `calc.sty` here because some of Song♭ook's declarative sections themselves contain calculations. In this section we also declare the `\if` constructs used in the package.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                      %%
%%          I N I T I A L   C O D E   P A R T           %%
%%                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%========================================================
%%  E A R L Y   P A C K A G E   D E P E N D E N C I E S  %
%%========================================================
```

Page layout calculations have become overly complex and so as of version 4.0 we now require `calc.sty` to make them readable once again. In every instance we could probably find a way to get along without `calc.sty`; however, since the package is a part of the LaTeX2e Base there is no logical reason to avoid its use.

```
\RequirePackage{calc}

```

14.1 If Constructs

Most of these `\if` contructs are needed for use in the *Delaration Of Options* section of `songbook.sty`. In each case, we create the if statement (a.k.a. the flag) and then immediately set it to a known value. Where there are several flags which act as sort of radio buttons, all of the flags are set so that none of them is selected; which has been done so that if we forget to deal with them properly in the *Declaration Of Options* code it will eventually manifest itself as an error.

Since the majority of the `\if`s have to be declared in this section, we will go ahead and declare the remaining `\if`s as well. It's simpler to maintain them when they are all in one place.

```
%%=====================================================
%%            I F   C O N S T R U C T S              %
%%=====================================================
```

14.1.1 Song♭ook Types

At any time, only one of `\ifChordBk`, `\ifOverhead`, or `\ifWordBk` may be true. These `\if`s correspond directly to the `chordbk`, `overhead`, and `wordbk` options; one of which *must* be used in the `\usepackage{}` statement used to invoke the Song♭ook style. All three flags are set to `\false`, and this fact is use later in order to confirm that the user had specified one of the 3 options in their document.

`\ifChordBk` `\ifChordBk` is true if the user specified the `chordbk` option.

`\ifOverhead` `\ifOverhead` is true if the user specified the `overhead` option.

`\ifWordBk` `\ifWordBk` is true if the user specified the `wordbk` option.

```
\newif\ifChordBk        \ChordBkfalse
\newif\ifOverhead       \Overheadfalse
\newif\ifWordBk         \WordBkfalse
```

14.1.2 Song♭ook Subtypes

A pair of `\if`s are declared to indicate whether we are only typesetting words on the page (i.e., the flag is false if we are typesetting words *and* chords). We are in words only mode when the user has declared either the `overhead` or `wordbk` options. When these flags are first declared they are set to the same value, false.

`\ifWordsOnly` `\ifWordsOnly` is true if we're in words-only mode.

`\ifNotWordsOnly` `\ifNotWordsOnly` always has a value oposite the of `\ifWordsOnly`. `\ifNotWordsOnly` is false if we are in words-only mode.

```
\newif\ifWordsOnly      \WordsOnlyfalse
\newif\ifNotWordsOnly   \NotWordsOnlyfalse
```

14.1.3 Song Indicator

`\ifSBinSongEnv` The `\ifSBinSongEnv` flag is provided to the style or a songbook's author to detect if the current text is inside of a `song` environment. This flag hasn't proven to be useful, but it doesn't hurt anything to leave it around; so, it hasn't been removed—who knows, there may well be a user somewhere making use of it! The `song` environment takes care of setting this flag's status.

```
\newif\ifSBinSongEnv    \SBinSongEnvfalse
```

14.1.4 Behaviour Flags

There are three flags which can be set in order to effect certain behaviours from the Song♭ook style. They are not related to one another but have been grouped together since they are they all `\if`s used to control Song♭ook behaviour.

`\ifCompactSongMode` `\ifCompactSongMode` is set to true if you want songs presented in a compact mode. It is initially set to false. This flag will *only* be set to true by the user; the style itself does not toggle this flag. Set this to true by specifying the `compactsong` option in the `\usepackage` statement.

`\ifExcludeSong` `\ifExcludeSong` is set to true if you want to have the current song excluded from the songbook. It is initially set to false, and would only be set to true inside a `song` environment, during processing of a song to be excluded. Its value is set to true when you pass a value of "`N`" as the first (optional) parameter to a `song` environment.

`\ifPrintAllSongs` `\ifPrintAllSongs` is set to true if you want to have Song♭ook print all a songbook's songs regardless of what option may have been specified in each song.

`\ifSamepageMode` `\ifSamepageMode` indicates we want the Song♭ook style to try and keep each song together on the same page. Set this true or false using the `\SamepageModetrue` and `\SamepageModefalse` commands, respectively. Important note: this command has *not* been documented in the *High Level Documentation* section, above; `\ifSamepageMode` is very unreliable. The LaTeX2e page breaking algorithms are not happy when this mode is used. The the `song` environment description, below, for a further explanation.

`\ifSongEject` `\ifSongEject` is set to true if we want a new page generated at the end of every `song` environment. A value of true means eject after every `song` environment (default value is true). Set this true or false using the `\SongEjecttrue` and `\SongEjectfalse` commands, respectively.

```
\newif\ifCompactAllMode \CompactAllModefalse
\newif\ifCompactSongMode\CompactSongModefalse
\newif\ifExcludeSong    \ExcludeSongfalse
\newif\ifPrintAllSongs  \PrintAllSongsfalse
\newif\ifSamepageMode   \SamepageModefalse
\newif\ifSongEject      \SongEjecttrue
```

14.1.5 Papesize Indicators

This next set of flags are needed to track the papersize specified by the user in then processed in the *Declaration Of Options* section. This set of `\if`s are mutually exclusive and only one of them should be true at any one time. They are all initially set to false; setting of a default value is done via an `\ExecuteOptions{}` clause, below. These flags were created for use by the Song♭ook style itself, but have been made part of the public interface to simplify page layout coding related to paper handling in a user's own songbook.

`\ifSBpaperA4` `\ifSBpaperA4` is true if papersize is A4.

`\ifSBpaperA5` `\fSBpaperA5` is true if papersize is A5.

`\ifSBpaperB5` `\ifSBpaperB5` is true if papersize is B5.

`\ifSBpaperLtr` `\ifSBpaperLtr` is true if papersize is US Letter.

`\ifSBpaperLgl` `\ifSBpaperLgl` is true if papersize is US Legal.

`\ifSBpaperExc`

`\ifSBpaperExc` is true if papersize is US Executive Letter.

```
\newif\ifSBpaperAfour    \SBpaperAfourfalse
\newif\ifSBpaperAfive    \SBpaperAfivefalse
\newif\ifSBpaperBfive    \SBpaperBfivefalse
\newif\ifSBpaperLtr      \SBpaperLtrfalse
\newif\ifSBpaperLgl      \SBpaperLglfalse
\newif\ifSBpaperExc      \SBpaperExcfalse
```

14.2 Fonts

Fonts are specified up-front in this section in order to simplify the `\DeclareOption{}` clauses that follow (i.e., those clauses need only make changes against these baseline settings). The fonts sizes and selections initially declared herein are those necessary for `chordbk` songbooks.

Fonts are handled by way of LaTeX2e commands defined using the `\newcommand` command. This was done specifically so that traditional LaTeX2e font selection occurs in the context the Songbook font command is used. I may have completely misunderstood how LaTeX2e does its font selection, in which case my implementation choice here is pointless; however, until proven otherwise... here it is.[1] Change these font specifiers via LaTeX2e's `\renewcommand.`

```
%%==========================================================
%%                       F O N T S                         %
%%==========================================================
```

14.2.1 Chord Fonts

These font selectors are used to determine how chords are printed in words and chords songbooks:

`\ChBassFont`

`\ChBassFont` sets the font for the bass note in chords as printed by the `\Ch`, `\Chr`, and `\ChX` commands.

`\ChBkFont`

`\ChBkFont` sets the font for square brackets typeset by `\Ch`, `\Chr`, and `\ChX` commands.

`\ChFont`

`\ChFont` sets the font for chords as printed by the `\Ch`, `\Chr`, and `\ChX` commands. This used to be set to `\bf\sf` (i.e., cmss12 at 14.4pt).

```
\newcommand{\ChBassFont}{\normalsize\bf\sf}       % = cmss12 at 12.0pt
\newcommand{\ChFont}{\large\fontfamily{\sfdefault}%
  \fontseries{sbc}\fontshape{n}\selectfont}      %=cmssbc12 at 14.4pt
\newcommand{\ChBkFont}{\ChFont\fontseries{m}      %
  \selectfont}                                   % =cmssm12 at 14.4pt
```

14.2.2 Title Block Fonts

These font selectors are used to select the fonts used in the Title Block that occurs that the start of each song:

`\CpyRtFont`

`\CpyRtFont` sets the font used to print the copyright symbol produced by the `\CpyRt` command.

`\CpyRtInfoFont`

`\CpyRtInfoFont` sets the font used to print the copyright licensing information parameter of the `\song` environment; which appears after the copyright information parameter under the song title.

`\STitleFont`

`\STitleFont` sets the font used to print the song title, as generated by the `\STitle` command.

`\STitleKeyFont`

`\STitleKeyFont` sets the font used to print the key a song is written in, as generated by the `\STitle` command.

---

[1] Given that `doc.dtx` uses fonts in this fashion, I feel I'm in pretty good company.

`\STitleNumberFont`

`\STitleNumberFont` sets the font used to print the `\SBSongCnt` in front of the song title, as generated by the `\STitle` command. This is one of two Song♭ook font commands that are implemented using a real LaTeX2e `\font` command; this turned out to be the easiest manner in which to obtain the desired fonts. In order to make the `\STitleNumberFont`'s behaviour the the same as the other Song♭ook font commands, the implementation is done indirectly; whereby the `\font` command is inserted into the `\STitleNumberFont` command so that it may be changed by the user in the same way as the other font commands in this package.

`\ScriptRefFont`

`\ScriptRefFont` sets the font used to print the scripture reference generated by the `\ScriptRef` command.

`\WandMFont`

`\WandMFont` sets the font used to print the lyricist and composer line generated by the `\WandM` command.

```
\newcommand{\CpyRtFont}{\footnotesize}          % = cmr10  at 10pt
\newcommand{\CpyRtInfoFont}{\tiny}              % = cmss8  at  8pt
\newcommand{\STitleFont}{\large\bf\sf}          % = cmss12 at 14.4pt
\newcommand{\STitleKeyFont}{\large}             % = cmr12  at 14.4pt
\font\STNFont=cmtt12 at 20pt
\newcommand{\STitleNumberFont}{\STNFont}        % = cmtt12 at 20pt
\newcommand{\ScriptRefFont}{\footnotesize}      % = cmr10  at 10pt
\newcommand{\WandMFont}{\footnotesize}          % = cmr10  at 10pt
```

14.2.3 Versicle Tag Fonts

These font selectors are used to select the fonts used to tag verses, choruses, bridges, and other elements with which a song is constructed (e.g., verse numbers, "Ch:" chorus indicator, etc.):

`\SBBracketTagFont`

`\SBBracketTagFont` sets the font used to create the tag for an `SBBracket` environment.

`\SBBridgeTagFont`

`\SBBridgeTagFont` sets the font used to create the tag for an `SBBridge` environment.

`\SBChorusTagFont`

`\SBChorusTagFont` sets the font used to print the chorus tag, `\SBChorusTag`.

`\SBEndTagFont`

`\SBEndTagFont` sets the font used to print the tag, `\SBEndTag`, for the `\SBEnd` command.

`\SBIntroTagFont`

`\SBIntroTagFont` sets the font used to print the introduction tag, `\SBIntroTag`.

`\SBOccursBrktFont`

`\SBOccursBrktFont` sets the font used to create the large left and right square brackets used to delimit the `\SBOccurs` environment.

`\SBOccursTagFont`

`\SBOccursTagFont` sets the font used to create the `\SBOccurs` tag.

`\SBVerseNumberFont`

`\SBVerseNumberFont` sets the font used to print the `\SBVerseCnt` in front of verses in an `SBVerse` environment.

`\SBSectionNumberFont`

`\SBSectionNumberFont` sets the font used to print the `\SBSectionCnt` in front of sections in an `SBSection` environment.

```
\newcommand{\SBBracketTagFont}{\small\bf\sf}    % = cmss10 at 10.0pt
\newcommand{\SBBridgeTagFont}{\SBEndTagFont}    % = cmss10 at 10.9pt
\newcommand{\SBChorusTagFont}{\small\bf\sf}     % = cmss10 at 10.9pt
\newcommand{\SBEndTagFont}{\small\bf\sf}        % = cmss10 at 10.9pt
\newcommand{\SBIntroTagFont}{\SBEndTagFont}     % = cmss10 at 10.9pt
\font\SBOBFont=cmss17 at 30pt
\newcommand{\SBOccursBrktFont}{\SBOBFont}       % = cmss17 at 30pt
\newcommand{\SBOccursTagFont}{\small\bf\sf}     % = cmss10 at 10.0pt
\newcommand{\SBVerseNumberFont}{\small\bf\sf}   % = cmss10 at 10.9pt
\newcommand{\SBSectionNumberFont}{\small\bf\sf} % = cmss10 at 10.9pt
```

14.2.4 Marginal Notes Fonts

These font selectors are used to select the fonts used when Song♭ook commands make notations in the margin of the songbook:

`\SBMargNoteFont` `\SBMargNoteFont` sets the font used in the marginal reference printed by the `\SBMargNote` command.

`\SBRefFont` `\SBRefFont` sets the font used in the marginal reference printed by the `\SBRef` command.

```
\newcommand{\SBMargNoteFont}{\scriptsize}       % = cmti8  at  8pt
\newcommand{\SBRefFont}{\SBMargNoteFont}        % = cmti8  at  8pt
```

14.2.5 Song Body Fonts

These font selector command are used to select fonts which are used within the body of songs:

`\SBDefaultFont` `\SBDefaultFont` sets the default font for the songbook. We will insert an occurrence of this command at the top of the songbook using the `\AtBeginDocument{}` clause, below.

`\SBLyricNoteFont` `\SBLyricNoteFont` sets the font used in comments placed within the lyrics giving musical direction. This is the only font command commonly used by the writer of a songbook. For example, to tag a line to be sung only by the Cantor and another by everyone, one would write:

```
{\SBLyricNoteFont (Cantor)} Give thanks to the Lord.

{\SBLyricNoteFont (All)} His love endures forever.
```

```
\newcommand{\SBDefaultFont}{\fontfamily{\rmdefault}%
  \large}                                       % = cmr12  at 14.4pt
\newcommand{\SBLyricNoteFont}{\footnotesize\sf} % = cmss10 at 10pt
```

14.2.6 Other Fonts

The remaining font selector commands:

`\SBOHContTagFont` `\SBOHContTagFont` sets the font used to print the `\OHContPgFtr` and `\OHContPgHdr`.

```
\newcommand{\SBOHContTagFont}{\small\bf\sf\itshape} % = cmss10 at 10.9pt

```

14.2.7 Compact Song Fonts

Downsized fonts to allow song to fit into half the space (i.e., two column mode) for `compactsong` printing; although the title will not be reset since it will be presented unchanged from normal chordbk mode.

`\ChBassFontCS`
`\ChFontCS`
`\ChBkFontCS`
`\SBDefaultFontCS`
`\SBOccursBrktFontCS`

```
\newcommand{\ChBassFontCS}{\small\bf\sf}         % = cmss12 at 11.0pt
\newcommand{\ChFontCS}{\normalsize\fontfamily{\sfdefault}%
      \fontseries{sbc}\fontshape{n}\selectfont}  % = cmssbc12 at 12.0pt
\newcommand{\ChBkFontCS}{\ChFont\fontseries{m}   %
      \selectfont}                               % = cmssm12 at 12.0pt
\newcommand{\SBDefaultFontCS}{\normalsize}       % = cmr12  at 12.0pt
\newcommand{\SBOccursBrktFontCS}{\large\bf\sf}   % = cmss10 at 10.9pt

```

14.2.8 Fonts Saving Variables

Variables in which to save the current fonts before we make changes for compact-song mode. We'll use them restore the original values again after leaving compact song mode.

`\ChBassFontSav`
`\ChFontSav`
`\ChBkFontSav`
`\SBDefaultFontSav`
`\SBOccursBrktFontSav`
`\SBFontSavVar`

```
\newcommand{\ChBassFontSav}{\relax}%
\newcommand{\ChFontSav}{\relax}%
\newcommand{\ChBkFontSav}{\relax}%
\newcommand{\SBDefaultFontSav}{\relax}%
\newcommand{\SBOccursBrktFontSav}{\relax}%
\newcommand{\SBFontSavVar}{\relax}%

```

14.3 Configurable Dimensions

In this section we define the spaces to leave in various situations.

All of these spaces are defined as LaTeX2e commands to overcome limitations in length evaluation. For example, if `\LeftMarginSBVerse` were to be defined as a length, and then immediately set to `4em`s the specific length would be evaluated with respect to the current font. This is not be what is desired; instead a length evaluated with respect to the font in place at the start of an `SBVerse` is what is desired. This can only be done by making these lengths LaTeX2e commands.

```
%%=======================================================
%%    C O N F I G U R A B L E   D I M E N S I O N S     %
%%=======================================================
```

14.3.1 Published Dimensions

While the bulk of the declared dimensions have been created to make the Song♭ook style more user configurable, there are also some dimensions which were created for internal use. This first section describes the user configurable dimensions:

`\HangAmt`

`\HangAmt` is the amount to indent when a line wraps. This has been defined using `\newcommand` instead of `\newlength` so that any unit definitions are evaluated at the time the `\HangAmt` command is used.

`\LeftMarginSBBracket`

`\LeftMarginSBBracket` is the amount of left margin left in front of `SBBracket`s and `SBBracket*`s in the songbook. The value for this variable has been chosen such that the song-words for `SBVerse`s, `SBChorus`es, and `SBBracket`s all align against the same left margin when printing standard words & chords songbooks.

`\LeftMarginSBChorus`

`\LeftMarginSBChorus` is the amount of left margin left in front of named choruses in the songbook. In most cases `\LeftMarginSBChorus`, `\LeftMarginSBSection`, and `\LeftMarginSBVerse` should all be the same value.

`\LeftMarginSBSection`

`\LeftMarginSBSection` is the amount of left margin left in front of sections in the songbook.

`\LeftMarginSBVerse`

`\LeftMarginSBVerse` is the amount of left margin left in front of verses in the songbook.

`\SBChordRaise`

`\SBChordRaise` is the distance to raise the chords above the baseline of the text they sit over.

`\SBRuleRaiseAmount`

`\SBRuleRaiseAmount` is the distance to raise the rule (as specified by `\SBIntersyllableRule`) which fills the space between adjoining syllables.

`\SpaceAboveSTitle`

`\SpaceAboveSTitle` is the space skipped by the `\STitle` macro before it prints the song title.

`\SpaceAfterTitleBlk` `\SpaceAfterTitleBlk` is the space inserted by the song environment between the *title block* and the versicles.

`\SpaceAfterChorus` `\SpaceAfterChorus` is the vertical space to leave after an `SBChorus` environment.

`\SpaceAfterOpGroup` `\SpaceAfterOpGroup` is the vertical space to leave after an `SBOpGroup` environment.

`\SpaceAfterSBBracket` `\SpaceAfterSBBracket` is the vertical space to leave after an `SBBracket` environment. This has proven troublesome to choose (see also `\SpaceBeforeSBBracket` because the `list` environment that produces the versicle inside of the `SBBracket` environment is itself enclosed inside of a math construct (which requires the `list` to have its vertical spacing supressed—otherwise the vertical line forming the left bracket encloses unnecessary whitespace). The vertical spacing around a list is created by way of some nontrivial macros and can't simply be copied into some other context. Thus, the choice of values for `\SpaceAfterSBBracket` and `\SpaceBeforeSBBracket` have been rather arbitrarily chosen.

`\SpaceAfterSection` `\SpaceAfterSection` is the vertical space to leave after an `SBSection` environment.

`\SpaceAfterSong` `\SpaceAfterSong` is the vertical space to leave after a song.

`\SpaceAfterVerse` `\SpaceAfterVerse` is the vertical space to leave after an `SBVerse` environment.

`\SpaceBeforeSBBracket` `\SpaceBeforeBBracket` is the vertical space to leave before an `SBBracket` environment. None of the other versicles have an extra space inserted before them. See `\SpaceAfterSBBracket` for further explanation.

```
\newcommand{\HangAmt}            {1.5em}
\newcommand{\LeftMarginSBBracket}{2.85em}
\newcommand{\LeftMarginSBChorus} {4em}
\newcommand{\LeftMarginSBSection}{\LeftMarginSBChorus}
\newcommand{\LeftMarginSBVerse}  {\LeftMarginSBChorus}
\newcommand{\SBChordRaise}       {2.25ex}
\newcommand{\SBOldChordRaise}    {2.90ex}
\newcommand{\SBRuleRaiseAmount}  {0.57ex}
\newcommand{\SpaceAboveSTitle}   {0.5in}
\newcommand{\SpaceAfterTitleBlk} {-1.75ex}
\newcommand{\SpaceAfterChorus}   {\vspace{0ex plus0ex  minus3ex}}
\newcommand{\SpaceAfterOpGroup}  {\vspace{0ex plus0ex  minus3ex}}
\newcommand{\SpaceAfterSBBracket}{\vspace{2ex plus1ex  minus1ex}}
\newcommand{\SpaceAfterSection}  {\vspace{0ex plus0ex  minus3ex}}
\newcommand{\SpaceAfterSong}     {\vspace{0ex plus10ex minus3ex}}
\newcommand{\SpaceAfterVerse}    {\vspace{0ex plus0ex  minus3ex}}
\newcommand{\SpaceBeforeSBBracket}{\vspace{1ex plus1ex  minus1ex}}
```

14.3.2 Internal Dimensions

These variables are used internally within Song♭ook macros. They are not part of the published `songbook.sty` interface; but a few of them can be used to tune some of its functions.

`\chSpaceTolerance`
`\chMiniSpace`
The `\chSpaceTolerance` and `\chMiniSpace` lengths are used in the `\Chr` macro.

`\sbBaselineSkipAmt` `\sbBaselineSkipAmt` is used internally in `SBVerse`, `SBChorus`, and all the other versicle environments; where hanging indentation has been accomplished using a specially defined list environment. The value of `\sbBaselineSkipAmt` is recalculated immediately before each being used in each *hanging indent* list.

```
\newlength{\chSpaceTolerance}   \setlength{\chSpaceTolerance}{1.5mm}
\newlength{\chMiniSpace}        \setlength{\chMiniSpace}     {0.3mm}
\newlength{\sbBaselineSkipAmt}  \setlength{\sbBaselineSkipAmt}{0pt}
```

\textwidthSav
\evensidemarginSav
\marginparwidthSav
\marginparsepSav
\chSpaceToleranceSav
\HangAmtSav
\LeftMarginSBChorusSav
\LeftMarginSBSectionSav
\LeftMarginSBVerseSav

The `\textwidthSav`, `\evensidemarginSav`, `\marginparwidthSav`, `\chSpaceToleranceSav`, `\HangAmtSav`, `\LeftMarginSBChorusSav`, `\LeftMarginSBSectionSav`, and `\LeftMarginSBVerse` lengths are used in `compactsong` processing; to save and then restore values from.

```
\newlength{\textwidthSav}
\newlength{\evensidemarginSav}
\newlength{\marginparsepSav}
\newlength{\marginparwidthSav}
\newlength{\chSpaceToleranceSav}
\newcommand{\HangAmtSav}{}
\newcommand{\LeftMarginSBChorusSav}{}
\newcommand{\LeftMarginSBSectionSav}{}
\newcommand{\LeftMarginSBVerseSav}{}

```

14.4 Declaration Of Non-Core Options

In the *Declaration Of Options* section of the `.sty` file we deal with the various options which a user may specify in the options part of the `\usepackage{songbook}` command. Since the Song♭ook style accepts standard LaTeX2e papersize options, we deal with those in addition to the style's own options. The documentation of these options is broken into two parts: the core options (`chordbk`, `wordbk`, & `overhead`), and the non-core options (all the rest).

The LaTeX2e documentation specifies that the options will be processed in the order in which they are listed in the `.sty` file. We take advantage of this fact and cause all of the options except the core three (`chordbk`, `wordbk`, & `overhead`) to simply set flags which indicate they were user-specified. The core options then do all the work.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                      %%
%%    D E C L A R A T I O N   O F   O P T I O N S      %%
%%                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

14.4.1 Papersize Options

Paper selection options inherited from Book Class. We process these first in order to remember what paper size the user has selected; before processing the Song♭ook's own options.

The code in each of these `\DeclareOption{}` clauses sets the `\SBpaper`... flags to unambiguously indicate which papersize the user specified.

```
%%=====================================================%
%%         P A P E R S I Z E   O P T I O N S           %
%%=====================================================%
```

a4paper

```
\DeclareOption{a4paper}{% Paper size: 210mm x 297mm
  \SBpaperAfourtrue
  \SBpaperAfivefalse
  \SBpaperBfivefalse
  \SBpaperLtrfalse
  \SBpaperLglfalse
  \SBpaperExcfalse
}

```

a5paper

```
\DeclareOption{a5paper}{% Paper size: 148mm x 210mm
  \SBpaperAfourfalse
  \SBpaperAfivetrue
  \SBpaperBfivefalse
  \SBpaperLtrfalse
  \SBpaperLglfalse
  \SBpaperExcfalse
}

```

b5paper

```
\DeclareOption{b5paper}{% Paper size: 176mm x 250mm
  \SBpaperAfourfalse
  \SBpaperAfivefalse
  \SBpaperBfivetrue
  \SBpaperLtrfalse
  \SBpaperLglfalse
  \SBpaperExcfalse
}

```

letterpaper

```
\DeclareOption{letterpaper}{% Paper size: 8.5in x 11in
  \SBpaperAfourfalse
  \SBpaperAfivefalse
  \SBpaperBfivefalse
  \SBpaperLtrtrue
  \SBpaperLglfalse
  \SBpaperExcfalse
}

```

legalpaper

```
\DeclareOption{legalpaper}{% Paper size: 8.5in x 14in
  \SBpaperAfourfalse
  \SBpaperAfivefalse
  \SBpaperBfivefalse
  \SBpaperLtrfalse
  \SBpaperLgltrue
  \SBpaperExcfalse
}

```

executivepaper

```
\DeclareOption{executivepaper}{% Paper size: 7.25in x 10.5in
  \SBpaperAfourfalse
  \SBpaperAfivefalse
  \SBpaperBfivefalse
  \SBpaperLtrfalse
  \SBpaperLglfalse
  \SBpaperExctrue
}

```

14.4.2 Compactallsongs Option

This option tells the Song♭ook style to present all the songs in a compact form, regardless of what has been specified in each song. For `chordbk` mode this means presenting the songs in two columns per page using a smaller font. When I can figure out what this option should mean for the other modes I'll code them up. In the mean time, `wordbk` and `overhead` modes simply ignore the `compactallsongs` option. Like the papersize options, the `compactallsongs` processing here simply sets a flag; the actual code required to implement `compactallsongs` mode is embedded below inside the three core options.

```
%%=====================================================%
%%     C O M P A C T A L L S O N G S   O P T I O N     %
%%=====================================================%
```

compactallsongs

```
\DeclareOption{compactallsongs}{%
  %%%
  % Set flag to indicate the user wants compact song mode
  % for all songs.
  \CompactAllModetrue
}

```

14.4.3 Compactsong Option

This option has been replaced with the combination of a new `compactallsongs` option and per-song specification of `compactsong` mode (see the documentation for the `song` environmnet.

So, we now stop processing and print an error message telling the user that the option has been removed.

```
%%=====================================================%
%%        C O M P A C T S O N G   O P T I O N         %
%%=====================================================%
```

compactsong

```
\DeclareOption{compactsong}{%
  \errmessage{The compactsong Songbook option has been
removed and replaced with a combination of a global
compactallsongs Songbook option and a per-song
environment compactsong option.  See the song
environment's documentation}
}

```

14.4.4 Printallsongs Option

This option tells the Song♭ook style to print all songs in the songbook, regardless of what has been specified in each song. Like the papersize options, the `printallsongs` processing here simply sets a flag; the actual code required to implement `printallsongs` mode is embedded below inside the `song` environment.

```
%%=====================================================%
%%      P R I N T A L L S O N G S   O P T I O N       %
%%=====================================================%
```

printallsongs

```
\DeclareOption{printallsongs}{%
  %%%
  % Set flag to indicate the user wants to print all songs.
  \PrintAllSongstrue
}

```

14.5 Declaration Of Core Options

Now we deal with the Options which set up the songbook instances appropriately; i.e., a “words-only”, “chords & words”, or “overhead master” book (`wordbk`, `chordbk`, & `overhead`). These option declarations take advantage of the fact that we have already been told what paper size to design for.

The style has been constructed on the underlying assumption that the user *must* specify one of the core options. To that end, we will later throw an error if none of these three options was executed (done at `\AtBeginDocument` time, see the top of the *Main Code Part* for details).

```
%%=====================================================%
%%     S O N G B O O K   C O R E   O P T I O N S      %
%%=====================================================%
```

14.5.1 `chordbk` Option

`chordbk` The `chordbk` option is executed here.

Each of the core options is structured similarly. As as result, the documentation for the first one, `chordbk`, will be more detailed, and the other two subsections will refer to this one.

```
\DeclareOption{chordbk}{%
```

Set flags to indicate that we *are* in `chordbk` mode. Set flags to indicate we are *not* in words-only mode. Indicate that we *do* want a page eject after every song.

```
  \ChordBktrue
  \WordBkfalse
  \Overheadfalse
  \WordsOnlyfalse
  \NotWordsOnlytrue
  \SongEjecttrue

```

Page Layout This first part specifies the page layout considerations.

Page layout usage recommendation: copy the appropriate page layout commands to the preamble of your own document and customize them appropriately. This will over-ride the default layout specified herein. Use a structure like this one to handle the three songbook types automatically for your songbooks:

```
\ifChordBk
  <page layout for Words & Chords books>
\else\ifWordBk
  <page layout for Words-Only books>
\else\ifOverhead
  <page layout for Overhead masters>
\fi\fi\fi
```

The only way I found to get these page layouts successfully built was to draw the various frames in a drawing package and then use a combination of page measurements and hand calculations to ensure I had everything done correctly. One of the key concepts that had not been evident to me until just recently was that on even pages the `\marginparsep` and `\marginparwidth` variables exist *inside* the `\evensidemargin`; this fact is not explicitly mentioned in any LaTeX manual I have read, not even in “The LaTeX Companion”!

The negative `\hoffset` and `\voffset` are to overcome the DVI driver default left and top margins of 1in, and all page layout commands herein assume these offsets have been “unset” in this fashion.

```
  \voffset=-1.00in
  \hoffset=-1.00in

```

Papersize-dependant processing. In general we don’t change anything except the page layout, however for smaller page sizes the some of the fonts are reduced to ensure that the songs fit reasonably onto the page.

```
  \ifSBpaperAfour
    \topmargin=0.5in
    \headheight=0.21in
    \headsep=0.2in
    \textheight=10.0in
    \footskip=0.19in
    %
    \oddsidemargin=0.618in
    \evensidemargin=1.4in
    \textwidth=6.25in
    \marginparsep=0.2in
    \marginparwidth=0.8in
  \else\ifSBpaperAfive
    \topmargin=6.0mm
```

```
   \headheight=5.334mm
   \headsep=2.666mm
   \textheight=185.17mm
   \footskip=4.826mm
   %
   \oddsidemargin=12.0mm
   \evensidemargin=30.0mm
   \textwidth=106.0mm
   \marginparsep=3.68mm
   \marginparwidth=20.32mm
```

Downsize the fonts to allow song to fit into the smaller A5 papersize.

```
   \renewcommand{\ChBassFont}{\small\bf\sf}        % = cmss12 at 11.0pt
   \renewcommand{\ChFont}{\normalsize\fontfamily{\sfdefault}%
     \fontseries{sbc}\fontshape{n}\selectfont}     %=cmssbc12 at 12.0pt
   \renewcommand{\ChBkFont}{\ChFont\fontseries{m}  %
     \selectfont}                                  % =cmssm12 at 12.0pt
   \renewcommand{\SBDefaultFont}{\normalsize}      % = cmr12  at 12.0pt
   \renewcommand{\SBOccursBrktFont}{\large\bf\sf}  % = cmss10 at 10.9pt
 \else\ifSBpaperBfive
   \topmargin=10.0mm
   \headheight=5.334mm
   \headsep=5.0mm
   \textheight=214.84mm
   \footskip=4.826mm
   %
   \oddsidemargin=20.0mm
   \evensidemargin=34.0 mm
   \textwidth=122.0mm
   \marginparsep=3.68mm
   \marginparwidth=20.32mm
```

Downsize the fonts to allow song to fit into the smaller B5 papersize.

```
   \renewcommand{\ChBassFont}{\small\bf\sf}        % = cmss12 at 11.0pt
   \renewcommand{\ChFont}{\normalsize\fontfamily{\sfdefault}%
     \fontseries{sbc}\fontshape{n}\selectfont}     %=cmssbc12 at 12.0pt
   \renewcommand{\ChBkFont}{\ChFont\fontseries{m}  %
     \selectfont}                                  % =cmssm12 at 12.0pt
   \renewcommand{\SBDefaultFont}{\normalsize}      % = cmr12  at 12.0pt
   \renewcommand{\SBOccursBrktFont}{\large\bf\sf}  % = cmss10 at 10.9pt
 \else\ifSBpaperLtr
   \topmargin=0.5in
   \headheight=0.21in
   \headsep=0.20in
   \textheight=9.4in
   \footskip=0.19in
   %
   \oddsidemargin=0.75in
   \evensidemargin=1.5in
   \textwidth=6.25in
   \marginparsep=0.2in
   \marginparwidth=0.8in
 \else\ifSBpaperLgl
   \topmargin=0.5in
   \headheight=0.21in
   \headsep=0.20in
   \textheight=12.4in
   \footskip=0.19in
   %
   \oddsidemargin=0.75in
   \evensidemargin=1.5in
   \textwidth=6.25in
   \marginparsep=0.2in
   \marginparwidth=0.8in
 \else\ifSBpaperExc
   \topmargin=0.25in
   \headheight=0.21in
   \headsep=0.165in
   \textheight=9.435in
   \footskip=0.19in
   %
```

```
    \oddsidemargin=0.5in
    \evensidemargin=1.25in
    \textwidth=5.5in
    \marginparsep=0.2in
    \marginparwidth=0.8in
  \fi\fi\fi\fi\fi\fi

```

Enable ragged bottom.

```
  \raggedbottom
}

```

14.5.2 `wordbk` Option

`wordbk` The `wordbk` option is executed here.

```
\DeclareOption{wordbk}{%
```

Set flags to indicate we *are* in wordbk mode. Set flags to indicate we *are* in words-only mode. Indicate that we do *not* want a page eject after every song.

```
  \ChordBkfalse
  \WordBktrue
  \Overheadfalse
  \WordsOnlytrue
  \NotWordsOnlyfalse
  \SongEjectfalse

```

Set fonts for `wordbk` use.

```
  \renewcommand{\SBDefaultFont}{\normalsize}
  \font\mySTNFont=cmtt12 at 17pt
  \renewcommand{\STitleNumberFont}{\mySTNFont}
  \renewcommand{\CpyRtFont}{\scriptsize}
  \renewcommand{\WandMFont}{\scriptsize}
  \renewcommand{\ScriptRefFont}{\scriptsize}
  \renewcommand{\SBOccursBrktFont}{\large\bf\sf}

```

Reset a few of the song spacing amounts.

```
  \renewcommand{\SpaceAboveSTitle}   {0.25in}
  \renewcommand{\LeftMarginSBChorus} {1.5em}
  \renewcommand{\LeftMarginSBSection}{\LeftMarginSBChorus}
  \renewcommand{\LeftMarginSBVerse}  {\LeftMarginSBChorus}

```

See the page layout comment in the `\DeclareOption{chordbk}` section, above, for usage recommendations w.r.t. page layout commands.

The negative `\hoffset` and `\voffset` are to overcome the DVI driver default left and top margins of 1in, and all page layout commands herein assume these offsets have been "unset" in this fashion.

```
  \voffset=-1.00in
  \hoffset=-1.00in

```

Papersize-dependant processing.

```
  \ifSBpaperAfour
    \topmargin=0.5in
    \headheight=0.21in
    \headsep=0.2in
    \textheight=10.0in
    \footskip=0.19in
    %
    \oddsidemargin=0.618in
    \evensidemargin=0.4in
    \textwidth=7.25in
    \marginparsep=0.0in
    \marginparwidth=0.0in
```

```
\else\ifSBpaperAfive
  \topmargin=6.0mm
  \headheight=5.334mm
  \headsep=2.666mm
  \textheight=185.17mm
  \footskip=4.826mm
  %
  \oddsidemargin=12.0mm
  \evensidemargin=6.0mm
  \textwidth=130.0mm
  \marginparsep=0.0mm
  \marginparwidth=0.0mm
\else\ifSBpaperBfive
  \topmargin=10.0mm
  \headheight=5.334mm
  \headsep=5.0mm
  \textheight=214.84mm
  \footskip=4.826mm
  %
  \oddsidemargin=20.0mm
  \evensidemargin=10.0mm
  \textwidth=146.0mm
  \marginparsep=0.0mm
  \marginparwidth=0.0mm
\else\ifSBpaperLtr
  \topmargin=0.5in
  \headheight=0.21in
  \headsep=0.10in
  \textheight=9.4in
  \footskip=0.29in
  %
  \oddsidemargin=0.75in
  \evensidemargin=0.5in
  \textwidth=7.25in
  \marginparsep=0.0in
  \marginparwidth=0.0in
\else\ifSBpaperLgl
  \topmargin=0.5in
  \headheight=0.21in
  \headsep=0.20in
  \textheight=12.4in
  \footskip=0.19in
  %
  \oddsidemargin=0.75in
  \evensidemargin=0.5in
  \textwidth=7.25in
  \marginparsep=0.0in
  \marginparwidth=0.0in
\else\ifSBpaperExc
  \topmargin=0.25in
  \headheight=0.21in
  \headsep=0.165in
  \textheight=9.435in
  \footskip=0.19in
  %
  \oddsidemargin=0.5in
  \evensidemargin=0.25in
  \textwidth=6.5in
  \marginparsep=0.0in
  \marginparwidth=0.0in
\fi\fi\fi\fi\fi\fi

```

Set ragged-right margins.

```
\raggedright

```

Do CompactSong processing, which at this time is nothing except resetting the `compactallsongs` flag back to false; to ensure that no `compactallsongs` processing occurs. We take time to print a warning message for the user to remind them that the `compactallsongs` option will not have any effect at this time.

```
  \ifCompactAllMode
    \typeout{``compactallsongs'' mode not implemented for Wordbk mode.}
    \CompactAllModefalse
  \fi
}

```

14.5.3 `overhead` Option

`overhead` The `wordbk` option is executed here.

```
\DeclareOption{overhead}{%
```

Set flags to indicate we *are* in overhead mode. Set flags to indicate we *are* in words-only mode. Indicate that we *do* want a page eject after every song.

```
  \ChordBkfalse
  \WordBkfalse
  \Overheadtrue
  \WordsOnlytrue
  \NotWordsOnlyfalse
  \SongEjecttrue

```

Set fonts for `overhead` use. Before doing any font stuff, change the regular sans serif font to demi-bold condensed.

```
  \def\@mss{cmssdc10}
  \renewcommand{\SBDefaultFont}{\LARGE\bf\sf}
  \renewcommand{\STitleNumberFont}{\Large\sf}
  \renewcommand{\STitleFont}{\LARGE\sf}
  \renewcommand{\CpyRtFont}{\normalsize\rm}
  \renewcommand{\CpyRtInfoFont}{\normalsize\rm}
  \renewcommand{\WandMFont}{\normalsize\rm}
  \renewcommand{\ScriptRefFont}{\normalsize\rm}
  \renewcommand{\SBLyricNoteFont}{\normalsize\rm}
  \renewcommand{\SBChorusTagFont}{\Large\sf}
  \renewcommand{\SBVerseNumberFont}{\Large\sf}
  \renewcommand{\SBSectionNumberFont}{\Large\sf}
  \renewcommand{\SBOccursTagFont}{\Large\sf}
  \renewcommand{\SBOccursBrktFont}{\huge\sf}
  \renewcommand{\SBBracketTagFont}{\Large\sf}
  \renewcommand{\SBOHContTagFont}{\Large\sf\itshape}

```

Reset a few of the song spacing amounts.

```
  \renewcommand{\SpaceAboveSTitle}   {0.25in}
  \renewcommand{\LeftMarginSBBracket}{2.25em}
  \renewcommand{\LeftMarginSBChorus} {1.5em}
  \renewcommand{\LeftMarginSBSection}{\LeftMarginSBChorus}
  \renewcommand{\LeftMarginSBVerse}  {\LeftMarginSBChorus}

```

Reset the . For some reason I'm not getting good results with the default value.

```
  \renewcommand{\baselinestretch}{.9}

```

See the page layout comment in the `\DeclareOption{chordbk}` section, above, for usage recommendations w.r.t. page layout commands.

General note re: `\textwidth` and overhead tranparencies: it is my personal experience that with font sizes used in overhead mode, a `\textwidth` of greater than 6in produces too wide an image for use in all situations. Depending upon how you intend to use your overheads, you may be able to use a wider image, however if you are uncertain I strongly recommend you stick with the 6in `\textwidth` that is specified herein.

The negative `\hoffset` and `\voffset` are to overcome the DVI driver default left and top margins of 1in, and all page layout commands herein assume these offsets have been "unset" in this fashion.

```
\voffset=-1.00in
\hoffset=-1.00in

```

Papersize-dependant processing.

```
\ifSBpaperAfour
  \topmargin=0.25in
  \headheight=0.25in
  \headsep=0.0in
  \textheight=10.3in
  \footskip=0.2in
  %
  \oddsidemargin=1.134in
  \evensidemargin=1.134in
  \textwidth=6.0in
  \marginparsep=0.0in
  \marginparwidth=0.0in
\else\ifSBpaperAfive
  \topmargin=0.0mm
  \headheight=5.334mm
  \headsep=0.0mm
  \textheight=193.666mm
  \footskip=4.826mm
  %
  \oddsidemargin=9.0mm
  \evensidemargin=9.0mm
  \textwidth=130.0mm
  \marginparsep=0.0mm
  \marginparwidth=0.0mm
\else\ifSBpaperBfive
  \topmargin=0.666mm
  \headheight=5.334mm
  \headsep=0.0mm
  \textheight=229.0mm
  \footskip=4.826mm
  %
  \oddsidemargin=15.0mm
  \evensidemargin=15.0mm
  \textwidth=146.0mm
  \marginparsep=0.0mm
  \marginparwidth=0.0mm
\else\ifSBpaperLtr
  \topmargin=0.25in
  \headheight=0.25in
  \headsep=0.0in
  \textheight=9.75in
  \footskip=0.2in
  %
  \oddsidemargin=1.25in
  \evensidemargin=1.25in
  \textwidth=6.0in
  \marginparsep=0.0in
  \marginparwidth=0.0in
\else\ifSBpaperLgl
  \topmargin=0.25in
  \headheight=0.25in
  \headsep=0.0in
  \textheight=12.8in
  \footskip=0.2in
  %
  \oddsidemargin=1.25in
  \evensidemargin=1.25in
  \textwidth=6.0in
  \marginparsep=0.0in
  \marginparwidth=0.0in
\else\ifSBpaperExc
  \topmargin=0.25in
```

```
    \headheight=0.21in
    \headsep=0.0in
    \textheight=9.6in
    \footskip=0.19in
    %
    \oddsidemargin=0.625in
    \evensidemargin=0.625in
    \textwidth=6.0in
    \marginparsep=0.0in
    \marginparwidth=0.0in
  \fi\fi\fi\fi\fi\fi

```

Set ragged-botton and ragged-right margins.

```
  \raggedright
  \raggedbottom

```

Do compactallsongs processing, which at this time is nothing except resetting the `compactallsongs` flag back to false; to ensure that no `compactallsongs` processing occurs. We take time to print a warning message for the user to remind them that the `compactallsongs` option will not have any effect at this time.

```
  \ifCompactAllMode
    \typeout{``compactallsongs'' mode not implemented for Overhead mode.}
    \CompactAllModefalse
  \fi
}

```

14.6 Execution Of Options

Here we tell the the Song♭ook style to execute the user's declared options.

First set up a default paper size, just in case the user didn't specify one. Then process the user specified options. It is mandatory for one of the songbook type options to be declared, but rather than delare a default we will throw an error (see below at the top of the *Main Code Part*).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                      %%
%%       E X E C U T I O N   O F   O P T I O N S        %%
%%                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\ExecuteOptions{letterpaper}
\ProcessOptions

```

14.7 Package Loading Part

In this section of the style we load the remaining styles upon which the Song♭ook style is dependant.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                      %%
%%       P A C K A G E   L O A D I N G   P A R T        %%
%%                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Donald Arseneau's `conditionals.sty`. This style is bundled with the Song♭ook style (Donald has often posted the macros to the USENET comp.text.tex newsgroup, but they haven't been formally submitted to CTAN.

```
\RequirePackage{conditionals}

```

Leslie Lamport's & David Carlilse's `ifthen.sty`. This style is part of the LaTeX2e distribution.

```
\RequirePackage{ifthen}

```

We load Christian Tellechea's `xtring` package to enable the new optional song format string available for the `song` environment. This style is available on CTAN.

```
\RequirePackage{xstring}

```

We load Frank Mittelbach's `multicol` package to enable use of `compactsong` mode. We specify the date of the 1.5u release; since we make use of the `\columnbreak` command which was only added in 1.5u.

```
\RequirePackage{multicol}[1999/05/25]

```

14.8 Main Code Part

The *Main Code Part* is the main part of the style. All of the "hard working" macros are detailed below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                      %%
%%              M A I N   C O D E   P A R T             %%
%%                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

The user *must* specify at least one of the `chordbk`, `wordbk`, or `overhead` options; otherwise we throw an error. This bit of code performs that check at the start of the users document.

We check to see if at least one of the core options have been specified by the user by populating an `\hbox{}` with the digit "1" if an option was specified. If no core option was specified then the `\hbox{}` will be empty and we throw an error.

`\AtBeginDocument`

```
\AtBeginDocument{%
  \setbox0=\hbox{}
  %
  \ifChordBk\setbox0=\hbox{1}\fi
  \ifWordBk\setbox0=\hbox{1}\fi
  \ifOverhead\setbox0=\hbox{1}\fi
  %
  \ifthenelse{\wd0 = 0}
    {\errmessage{No songbook option (i.e., type) specified.
        Specify a songbook mode in your usepackage
        statement; one of: [chordbk], [wordbk], or [overhead]}}
    {\relax}

```

If the user had specified one of the required options then we continue with setting things up. At present, the only housekeeping item that needs attention is to set the default font for the songbook. We do this by inserting the `\SBDefaultFont` here; this lifts from the user the burden of having to remember to specifying inside the songbook.

```
  \SBDefaultFont
}

```

14.8.1 Constants & Variables

Define Counters used herein.

```
%%======================================================
%%      C O N S T A N T S   &   V A R I A B L E S       %
%%======================================================

```

`\theSBSongCnt` The `\theSBSongCnt` counter is used for numbering the songs. When a song is listed multiple times (for multiple keys) the songs number must remain the same each time.

`\theSBSectionCnt` The `\theSBSectionCnt` counter is used for numbering sections as they occur within a song.

`\theSBVerseCnt` The `\theSBVerseCnt` counter is used for numbering verses as they occur within a song.

```
\newcounter{SBSongCnt}
\newcounter{SBSectionCnt}
\newcounter{SBVerseCnt}

```

String Constants Declare string constants.

These constants are provided so that users may easily customize the appearance of formatted songs and songbooks. Use the `\renewcommand` command to change the value of these constants.

`\OHContPgFtrTag` The `\OHContPgFtrTag` tag is inserted by the `\OHContPgFtr` command. The default value for this is “`continued on next page\ldots`”.

`\OHContPgHdrTag` The `\OHContPgHdrTag` tag is inserted by the `\OHContPgHdr` command. The default value for this is “`\theSBSongCnt\ --- \theSongTitle, continued\ldots`”.

`\SBBaseLang` The `\SBBaseLang` tag is the name of the language of all songs not specified within an `songTranslation` environment, and also as the default value of the `songTranslation` environment’s optional song language parameter. The default value for this is “`English`”.

`\SBBridgeTag` The `\SBBridgeTag` tag is inserted before the start of a bridge. The default value for this is “`Bridge:`”.

`\SBChorusTag` The `\SBChorusTag` tag is inserted before the first line of a chorus. The default value for this is “`Ch:`”.

`\SBContinueTag` The `\SBContinueTag` tag is inserted in an `\SBContinueMark`. The default value for this is “`cont\ldots`”.

`\SBEndTag` The `\SBEndTag` tag is inserted before the start of an ending (in an `\SBEnd` command). The default value for this is “`End:`”.

`\SBIntersyllableRule` The `\SBIntersyllableRule` tag is actually the command(s) used to draw the rule between adjoining syllables.

`\SBIntroTag` The `\SBIntroTag` tag is inserted before the start of an introduction (in an `\SBIntro` command). The default value for this is “`Intro:`”.

`\SBPubDom` The `\SBPubDom` tag is used to indicate that a song is in the public domain. The default value for this is “`Public Domain`”. If you want to localize this string in the song title block, be sure to use this public interface: the `\CpyRt` macro uses `\SBPubDom` to determine whether or not to print the copyright symbol (©).

`\SBUnknownTag` The `\SBUnknownTag` tag is used with the `\WAndM` command and is the string to insert when either the author/artist or the copyright holder is unknown. The default value for this is "`Unknown`".

`\SBWAndMTag` The `\SBWAndMTag` the tag is insert before the words and music entry printed in the song header. The default value for this is "`W&M:`".

`\Songbook` The macro used to print this style's name. The 'b' in the word songbook has been replace with a flat (♭).

```
\newcommand{\OHContPgFtrTag}     {continued on next page\ldots}
\newcommand{\OHContPgHdrTag}     {\theSBSongCnt\ --- \theSongTitle, continued\ldots}
\newcommand{\SBBaseLang}  {English}
\newcommand{\SBBridgeTag}        {Bridge:}
\newcommand{\SBChorusTag}        {Ch:}
\newcommand{\SBContinueTag}      {cont\ldots}
\newcommand{\SBEndTag}           {End:}
\newcommand{\SBIntersyllableRule}{\hrulefill}
\newcommand{\SBIntroTag}         {Intro:}
\newcommand{\SBPubDom}           {Public Domain}
\newcommand{\SBUnknownTag}       {Unknown}
\newcommand{\SBWAndMTag}         {W\&M:}
\newcommand{\Songbook}           {\textrm{Song$\flat$ook}}

```

Internal Song Variables Declare song attribute variables.

These variables are intended for consumption within the songbook style itself, so they will *not* be documented in the *High Level Documentation* section, above.

`\theSongComposer` `\theSongComposer` is the composer and lyricist of the last song.

`\theSongComposerU` `\theSongComposerU` takes the value of `\theSongComposer` except when the song composer parameter was left empty in the songbook; in which case this variable will be assigned the value of `\SBUnknownTag`.

`\theSongKey` `\theSongKey` is the key of the last song. This variable must be reset within the `\STitle` command, as well as at the start of the song environment, because of the way in which extra keys are handled.

`\theSongLicense` `\theSongLicense` is the copyright license info.

`\theSongTitle` `\theSongTitle` is the title of the last song.

`\theCopyRtInfo` `\theCopyRtInfo` is the copyright information of the last song. This includes the copyright licensing information.

`\theScriptureRef` `\theScriptureRef` is the scripture reference of the last song.

`\theXlatnBy` `\theXlatnBy` is who translated the song.

`\theXlatnLang` `\theXlatnLang` is the language the song has been translated into.

`\theXlatnPerm` `\theXlatnPerm` is the permission details for the last song translation. This variable is reset to an empty string at the start of each song environment.

`\theXlatnTitle` `\theXlatnTitle` is the title of the last song-translation. This variable is reset to an empty string at the start of each song environment.

```
\newcommand{\theSongComposer}{the Composer}
\newcommand{\theSongComposerU}{the ComposerU}
\newcommand{\theSongCopyRt}{the Copyright}
\newcommand{\theSongKey}{the Key}
\newcommand{\theSongLicense}{the License}
\newcommand{\theSongScriptRef}{the Scripture}
\newcommand{\theSongTitle}{the Title}
\newcommand{\theXlatnBy}{the Translator}
\newcommand{\theXlatnLang}{the Language}
\newcommand{\theXlatnPerm}{the Permission}
\newcommand{\theXlatnTitle}{the Translation Title}

```

14.8.2 Special Characters

Some macros to ease the entry of special characters in songbooks.

```
%%=====================================================%
%%         S P E C I A L   C H A R A C T E R S         %
%%=====================================================%

```

`\SBem` `\SBem` — em-dash macro definition.

Parameters:

None.

Generate an em-dash within a songbook. This macro is used to place in em-dash within text when we're *not* in words-only mode. This allows us to place dashes within text in order place a chord earlier than a sylable; yet, that dash will not appear in the words-only book. The words-only version of this macro is a no-op. Example of intended use:

```
\newcommand{\SBem}{\ifWordsOnly\relax\else---\fi}

```

`\SBen` `\SBen` — en-dash macro definition.

Parameters:

None.

Generate an en-dash within a songbook. This macro is used to place in en-dash within text when we're *not* in words-only mode; just like `\SBem`. The words-only version of this macro is a no-op.

```
\newcommand{\SBen}{\ifWordsOnly\relax\else--\fi}

```

`\SBContinueMark` `\SBContinueMark` — conditionally produce a continuation symbol.

Parameters:

None.

If the contents of `\rightmark` will result in nothing being typeset, then don't output the continuation mark; otherwise, output a continuation mark using the `\SBContinueTag` command.

```
\newcommand{\SBContinueMark}{%
  \setbox0=\hbox{\rightmark}
  \ifthenelse{\lengthtest{\wd0 = 0pt}}
  {\relax}%
  {\SBContinueTag}%
  }

```

`\OHContPgFtr` `\OHContPgFtr` — macro to print page footing continuation headers on overheads.

Parameters:

None.

This macro must be manually inserted where needed. It is generally used in conjunction with the `\OHPageBrk` and `\OHPageHdr` macros. `\OHContPgFtr` is a no-op, except when `\ifOverhead` is true.

```
\newcommand{\OHContPgFtr}{%
  \ifOverhead
    \vskip .25in
    \centerline{\SBOHContTagFont\OHContPgFtrTag}
  \else%
    \relax%
  \fi}
```

`\OHContPgHdr` `\OHContPgHdr` — macro to print page heading continuation headers on overheads.

Parameters:

None.

This macro must be manually inserted where needed. It is generally used in conjunction with the `\OHPageBrk` macro. `\OHContPgHdr` is a no-op, except when `\ifOverhead` is true.

```
\newcommand{\OHContPgHdr}{%
  \ifOverhead
    \centerline{\SBOHContTagFont\OHContPgHdrTag}
    \vskip .25in
  \else%
    \relax%
  \fi}

```

14.8.3 Table Of Contents & Indices

The macros used to create the *Key Index*, the *Title & First Line Index*, and the `Table Of Contents`. Planned enhancements are the addition of a *Scripture Index* and a *Artist Index*; i.e., an index of the `\ScriptRef{}` and `\WandM{}` entries, respectively.

Most of the specific code involved in managing the index files and writing the entries was copied from `latex.tex` (version 2.09) and then modified to suit our purposes here.

```
%%=======================================================%
%%          T A B L E   O F   C O N T E N T S           %
%%                                                       %
%%                A N D   I N D I C E S                  %
%%=======================================================%
```

`\makeArtistIndex`

`\makeArtistIndex` starts the creation of an index of artists.

Parameters:

None.

```
\def\makeArtistIndex{\if@filesw \newwrite\@artistIndexfile
  \immediate\openout\@artistIndexfile=\jobname.aIdx
  \def\artistIndex{\@bsphack\begingroup
            \def\protect####1{\string####1\space}\@sanitize
            \@wrArtistIndex}\typeout
  {Writing index file \jobname.aIdx }\fi}

```

`\artistIndex`

`\artistIndex[⟨1⟩][⟨2⟩]` makes an entry in the index of songs by artist.

Parameters:

⟨*1*⟩ Song artist.

⟨*2*⟩ Song title and number.

```
\def\@wrArtistIndex#1#2{\let\thepage\relax
   \xdef\@gtempa{\write\@artistIndexfile{\string
      \indexentry{#1}{#2}}}\endgroup\@gtempa
   \if@nobreak \ifvmode\nobreak\fi\fi\@esphack}

\def\artistIndex{\@bsphack\begingroup \@sanitize\@artistIndex}

\def\@artistIndex#1#2{\endgroup\@esphack}

```

`\makeKeyIndex`

`\makeKeyIndex` starts the creation of an index of songs by key.

Parameters:

None.

```
\def\makeKeyIndex{\if@filesw \newwrite\@keyIndexfile
  \immediate\openout\@keyIndexfile=\jobname.kIdx
  \def\keyIndex{\@bsphack\begingroup
            \def\protect####1{\string####1\space}\@sanitize
            \@wrKeyIndex}\typeout
  {Writing index file \jobname.kIdx }\fi}

```

`\keyIndex` `\keyIndex[`⟨*1*⟩`][`⟨*2*⟩`]` makes an entry in the index of songs by key.

Parameters:

⟨*1*⟩ Song key and title.
⟨*2*⟩ Song number.

```
\def\@wrKeyIndex#1#2{\let\thepage\relax
   \xdef\@gtempa{\write\@keyIndexfile{\string
      \indexentry{#1}{#2}}}\endgroup\@gtempa
   \if@nobreak \ifvmode\nobreak\fi\fi\@esphack}

\def\keyIndex{\@bsphack\begingroup \@sanitize\@keyIndex}

\def\@keyIndex#1#2{\endgroup\@esphack}

```

`\makeTitleIndex` `\makeTitleIndex` starts creation of a title & first line index.

Parameters:

None.

```
\def\makeTitleIndex{\if@filesw \newwrite\@titleIndexfile
  \immediate\openout\@titleIndexfile=\jobname.tIdx
  \def\titleIndex{\@bsphack\begingroup
             \def\protect####1{\string####1\space}\@sanitize
             \@wrTitleIndex}\typeout
  {Writing index file \jobname.tIdx }\fi}

```

`\titleIndex` `\titleIndex[`⟨*1*⟩`][`⟨*2*⟩`]` makes an entry in the title & first line index.

Parameters:

⟨*1*⟩ Song title or first line.
⟨*2*⟩ Song number.

```
\def\@wrTitleIndex#1#2{\let\thepage\relax
   \xdef\@gtempa{\write\@titleIndexfile{\string
      \indexentry{#1}{#2}}}\endgroup\@gtempa
   \if@nobreak \ifvmode\nobreak\fi\fi\@esphack}

\def\titleIndex{\@bsphack\begingroup \@sanitize\@titleIndex}

\def\@titleIndex#1#2{\endgroup\@esphack}

```

`\makeTitleContents` `\makeTitleContents` starts creation of a table of contents.

Parameters:

None.

```
\def\makeTitleContents{\if@filesw \newwrite\@titleContentsfile
  \immediate\openout\@titleContentsfile=\jobname.toc
  \def\titleContents{\@bsphack\begingroup
             \def\protect####1{\string####1\space}\@sanitize
             \@wrTitleContents}\typeout
  {Writing table of contents file \jobname.toc }\fi}

```

`\titleContents` `\titleContents[`⟨*1*⟩`][`⟨*2*⟩`]` makes an entry in the table of contents file.

Parameters:

⟨*1*⟩ Song number.
⟨*2*⟩ Song title.

```
\def\@wrTitleContents#1#2{\let\thepage\relax
   \xdef\@gtempa{\write\@titleContentsfile{\string
       \item\ \theSBSongCnt. #1\protect\hbox{, \thepage}}}\endgroup\@gtempa
   \if@nobreak \ifvmode\nobreak\fi\fi\@esphack}

\def\titleContents{\@bsphack\begingroup \@sanitize\@titleContents}

\def\@titleContents#1#2{\endgroup\@esphack}

```

`\SBtocSEntry`

`\SBtocSEntry` is the macro that encloses each skipped song TOC entry. The intent is that when you format your skipped TOC list you redefine `\SBtocSEntry` appropriately (assuming you are not happy with the default value).

Parameters:

⟨*1*⟩ Song number.
⟨*2*⟩ Song title.
⟨*3*⟩ Page number.

```
\newcommand{\SBtocSEntry}[3]{#1. \textit{#2}\hbox{, #3}}

```

`\makeTitleContentsSkip`

`\makeTitleContentsSkip` starts creation of a table of contents of songs excluded from the songbook.

Parameters:

None.

```
\def\makeTitleContentsSkip{\if@filesw \newwrite\@titleContentsSkipfile
  \immediate\openout\@titleContentsSkipfile=\jobname.tocS
  \def\titleContentsSkip{\@bsphack\begingroup
            \def\protect####1{\string####1\space}\@sanitize
            \@wrTitleContentsSkip}\typeout
  {Writing table of contents (skipped) file \jobname.tocS }\fi}

```

`\titleContentsSkip`

`\titleContentsSkip[`⟨*1*⟩`][`⟨*2*⟩`]` makes an entry in the table of contents file.

Parameters:

⟨*1*⟩ Song number.
⟨*2*⟩ Song title.

```
\def\@wrTitleContentsSkip#1#2{\let\thepage\relax
   \xdef\@gtempa{\write\@titleContentsSkipfile{\string
       \item\ \protect\SBtocSEntry{\theSBSongCnt}{#1}{\thepage}}}\endgroup\@gtempa
   \if@nobreak \ifvmode\nobreak\fi\fi\@esphack}

\def\titleContentsSkip{\@bsphack\begingroup \@sanitize\@titleContentsSkip}

\def\@titleContentsSkip#1#2{\endgroup\@esphack}

```

`\FLineIdx`

`\FLineIdx[`⟨*1*⟩`]` adds a first line of song entry to the song & title index file (`.idx`).

Parameters:

⟨*1*⟩ First line of song.

```
\newcommand{\FLineIdx}[1]{\titleIndex{#1@{\it #1\/}}{\theSBSongCnt}}

```

14.8.4 Some Other Hooks

The macros have been provided to allow the user additional control of songbooks created by the Songbook package.

```
%%======================================================%
%%          S O M E   O T H E R   H O O K S             %
%%======================================================%

```

`\SBChorusMarkright`

The `\SBChorusMarkright[`⟨*1*⟩`]` hook to allow `\SBSection`'s `\markright` to be overridden.

```
\newcommand{\SBChorusMarkright}[1]{\markright{#1}}

```

`\SBVerseMarkright`

The `\SBVerseMarkright[`⟨*1*⟩`]` hook to allow `\SBVerse`'s `\markright` to be overridden.

```
\newcommand{\SBVerseMarkright}[1]{\markright{#1}}

```

`\SBSectionMarkright` The `\SBSectionMarkright[`⟨*1*⟩`]` hook to allow `\SBSection`'s `\markright` to be overridden.

```
\newcommand{\SBSectionMarkright}[1]{\markright{\alph{#1}}}

```

`\SongMarkboth` The `\SongMarkboth[`⟨*1*⟩`][`⟨*2*⟩`]` hook to allow the song environment's `\markboth` to be overridden.

```
\newcommand{\SongMarkboth}[2]{\markboth{#1}{#2}}

```

`\STitleMarkboth` The `\STitleMarkboth[`⟨*1*⟩`][`⟨*2*⟩`]` hook to allow `\Stitle`'s `\markboth` to be overridden.

```
\newcommand{\STitleMarkboth}[2]{\markboth{#1}{#2}}

```

14.8.5 Miscellaneous Macros

This section contains a few miscellaneous macros used by the main macros that then follow.

```
%%=====================================================%
%%      M I S C E L L A N E O U S   M A C R O S       %
%%=====================================================%

```

`\CpyRt` The `\CpyRt[`⟨*1*⟩`][`⟨*2*⟩`][`⟨*3*⟩`]` copyright info. macro definition.

Parameters:

⟨*1*⟩ Centre this line Y/N? (optional)
⟨*2*⟩ Copyright information.
⟨*3*⟩ Copyright licensing information.

This command is not usually explicitly used in a songbook. It is called by the `song` environment and will normally only be used there.

The first parameter to this macro is optional and is used to surpress the centering of the Scripture reference (i.e., if the parameter is specified, and that value is *not* '`Y`' then the center environment will not be created around the reference.

```
\newcommand{\CpyRt}[3][Y]{%
  \if#1Y\begin{center}\fi
    \if\blank{#2}%
      \if\blank{#3}%
        {\CpyRtFont\copyright \SBUnknownTag{} \CpyRtInfoFont}%
      \else
        {\CpyRtFont\copyright \SBUnknownTag{} \CpyRtInfoFont #3}%
      \fi%
    \else%
      \ifthenelse{\equal{#2}{\SBPubDom}}
      {%then
        {\CpyRtFont #2 \CpyRtInfoFont #3}%
      }{%else
        {\CpyRtFont\copyright #2 \CpyRtInfoFont #3}%
      }%fi
    \fi%
  \if#1Y\end{center}\fi
}

```

`\ScriptRef` The `\ScriptRef[`⟨*1*⟩`][`⟨*2*⟩`]` macro indicates a scripture reference.

Parameters:

⟨*1*⟩ Centre this line Y/N? (optional)
⟨*2*⟩ Address of scripture reference for the song.

Used to indicate a scripture reference for the song. May either be the scripture being quoted in the song, or a scripture which supports the theology presented in the song.

The first parameter to this macro is optional and is used to surpress the centering of the Scripture reference (i.e., if the parameter is specified, and that value is *not* ‘`Y`’ then the center environment will not be created around the reference.

```
\newcommand{\ScriptRef}[2][Y]{%
  \if#1Y\begin{center}\fi
    {\ScriptRefFont #2}%
  \if#1Y\end{center}\fi
}

```

`\WAndM`

The `\WAndM[`⟨*1*⟩`][`⟨*2*⟩`]` macro indicates Words and Music authorship.

Parameters:
⟨*1*⟩ Centre this line Y/N? (optional)
⟨*2*⟩ Name(s) of the composer and lyricist.

This command is not usually explicitly used in a songbook. It is called by the song environment and will normally only be used there.

The first parameter to this macro is optional and is used to surpress the centering of the composer & lyricist (i.e., if the parameter is specified, and that value is *not* ‘`Y`’ then the center environment will not be created around the composer & lyricist.

```
\newcommand{\WAndM}[2][Y]{%
  \if#1Y\begin{center}\fi
    \if\blank{#2}%
      {\WandMFont\SBWAndMTag ~\SBUnknownTag}%
    \else
      {\WandMFont\SBWAndMTag ~#2}%
    \fi
  \if#1Y\end{center}\fi
}

```

`\sbSetsbBaselineSkipAmt`

`\sbSetsbBaselineSkipAmt` sets the `\sbBaselineSkipAmt` length.

Parameters:
None.

This command is only used internally within the songbook style. It is invoked just prior to any use of the sbBaselineSkipAmt length and it calculated the proper value based upon all the fonts chosen at that particular moment in time. It does this by creating an `\hbox{}` that contains one letter with a chord overtop of it; the height and depth of that `\hbox{}` added together then become the baseline skip.

```
\newcommand{\sbSetsbBaselineSkipAmt}{
  \ifChordBk%
    \setbox0=\hbox{\strut\raise\SBChordRaise\hbox{\ChFont\sbChord{}A\relax\strut}A}%
    \setlength{\sbBaselineSkipAmt}{\ht0 + \dp0}%
  \else%
    \setlength{\sbBaselineSkipAmt}{\baselineskip}%
  \fi%
}

```

14.8.6 Primary Songbook Macros

The macros in this section comprise those most often used by Song♭ook users.

```
%%=====================================================%
%%    P R I M A R Y   S O N G B O O K   M A C R O S    %
%%=====================================================%

```

`\STitle`

`\STitle[`⟨*1*⟩`][`⟨*2*⟩`][`⟨*3*⟩`]` is the song title macro.

Parameters:
⟨*1*⟩ Centre this line Y/N? (optional)
⟨*2*⟩ Song’s title.

⟨*3*⟩ Song's Key.

Before printing the title we reset the `\SBVerseCnt` and `\SBSectionCnt` counters back to zero. This is for songs which are printed in more than one key, because the verse count always begins at "`1.`" for each key.

The first parameter to this macro is optional and is used to surpress the centering of the title (i.e., if the parameter is specified, and that value is *not* '`Y`' then the center environment will not be created around the title.

This macro also makes an entry in the key index file; except in the case where a song is not being included, in which case no entry is made.

```
\newcommand{\STitle}[3][Y]{%
  \setcounter{SBVerseCnt}{0}%
  \setcounter{SBSectionCnt}{0}%
  \ifExcludeSong\relax%
    \else\keyIndex{{\protect\sbChord#3\protect\relax} -- #2}{\theSBSongCnt}\fi%
  \vspace{\SpaceAboveSTitle}%
  \if#1Y\begin{center}\fi
    {\STitleNumberFont\theSBSongCnt}{\STitleFont\ --- #2}%
    \ifWordsOnly\relax\else{\STitleKeyFont\ [{\sbChord#3\relax}]}\fi%
  \if#1Y\end{center}\fi
  \STitleMarkboth{#2}{\relax}%
  }

```

`song` `song[`⟨*1*⟩`]`... `[`⟨*7*⟩`]` is the environment within which a song is entered.

Parameters:

⟨*1*⟩ Song format string (optional).
⟨*2*⟩ Title of song.
⟨*3*⟩ Key song is written in.
⟨*4*⟩ Copyright information.
⟨*5*⟩ Name(s) of composer and lyricist.
⟨*6*⟩ Scripture reference for the song.
⟨*7*⟩ Copyright licensing information.

The song environment encapsulates a song, including multiple appearances for multiple keys and translations. We increment the song counter and then cause the title and other parameter information to be displayed.

Song Format String (options) The first parameter is optional and its default value is "`YF`". Each letter in the format string controls one aspect of a song's formating. The format string options are:

Parameters:

`Y N` : Include song in book; specify "`Y`" or "`N`"?
`C F` : `compactsong` or not (full size); specify "`C`" or "`F`"?

YN: Include Song in Book? When the value of ⟨*YN: Include Song in Book?*⟩ is "`Y`" then all processing is done normally. If the value of ⟨*YN: Include Song in Book?*⟩ is "`N`" then:

- the songcounter is incremented;
- a TOC index entry is written to a skipped-entry files, with each entry bracketted by some extra code (compared to the non-skipped-entry files);
- consider making ⟨*Include?*⟩ look for several values to allow exclusions/inclusions to only happen for certain types of songbooks.

If both "`Y`" and "`N`" are specified, then "`Y`" is taken as the specified value. Don't forget that a value of "`N`" will be overridden by the global `printallsongs` option.

The skipped-entry TOC file is named `*.tocS`. The purpose of creating a separate file is twofold: (1) to allow normal songbook processing to simply omit these not-included files; (2) to allow the skipped entries to be easily added back into the TOC processing process through simple appending of the files to the standard TOC file.

CF: Compactsong or Full Size? When the value of ⟨*CF: Compactsong or Full Size?*⟩ is “`F`” then all processing is done normally. If the value is “`C`” then the song is printed in `compactsong` mode. If both “`F`” and “`F`” are specified, then “`F`” is taken as the specified value. Don’t forget that a value of “`F`” will be overridden by the global `compactallsongs` option.

```
\newenvironment{song}[7][YF]{                 % Comment markers to negate
  \IfSubStr{\uppercase{#1}}{N}{\ExcludeSongtrue}{\relax}% newlines.
  \IfSubStr{\uppercase{#1}}{Y}{\ExcludeSongfalse}{\relax}%
  \ifPrintAllSongs\ExcludeSongfalse\fi        %
  \IfSubStr{\uppercase{#1}}{C}{\CompactSongModetrue}{\relax}% newlines.
  \IfSubStr{\uppercase{#1}}{F}{\CompactSongModefalse}{\relax}%
  \ifCompactAllMode\CompactSongModetrue\fi    %
  \SongMarkboth{\relax}{\relax}               %
  \SBinSongEnvtrue                            %
  \renewcommand{\SBinSongEnv}{\True}          %
  \ifWordsOnly                                %
    \setlength{\parindent}{0pt}               %
  \fi                                         %
```

Store each of the parameters in a macro to make them easily accessible later. This isn’t as useful as it should be due to my inability to properly detect in the title block macros whether or not the parameter is nil or blank when one of these `\the` macros is passed instead of the native parameter itself.

We *clear* the translation macros now, since the `songTranslation` environment is only valid inside a `song` environment, and we are now declaring a new `song`.

```
  \renewcommand{\theSongComposer}{#5}         %
  \if\blank{#5}                               %
    \renewcommand{\theSongComposerU}{\SBUnknownTag}%
  \else                                       %
    \renewcommand{\theSongComposerU}{#5}      %
  \fi                                         %
  \renewcommand{\theSongCopyRt}{#4}           %
  \renewcommand{\theSongKey}{#3}              %
  \renewcommand{\theSongLicense}{#7}          %
  \renewcommand{\theSongScriptRef}{#6}        %
  \renewcommand{\theSongTitle}{#2}            %
  \renewcommand{\theXlatnBy}{}                %
  \renewcommand{\theXlatnLang}{\SBBaseLang}   %
  \renewcommand{\theXlatnPerm}{}              %
  \renewcommand{\theXlatnTitle}{}             %
  %
  \addtocounter{SBSongCnt}{1}                 %
  %
```

Write table of contents and index entries in reponse to the user’s ⟨*Include?*⟩ directive.

```
  \ifExcludeSong                              %
    \titleContentsSkip{\theSongTitle}{\theSongKey}%
  \else                                       %
    \titleIndex{\theSongTitle}{\theSBSongCnt}  %
    \titleContents{\theSongTitle}{\theSongKey} %
    \artistIndex{\theSongComposerU+\theSongTitle}{\theSBSongCnt}%
  \fi                                         %
```

Now we deal with the user’s ⟨*Include?*⟩ directive. If ⟨*Include?*⟩ is “`Y`” then we will cause normal songbook processing to occur; otherwise we’ll simply insert a `\relax` macro. I have implemented this feature using a memory hungry method: when excluding a song, put the lyrics into `box2` and then discard it without using it. Although Mark Wooding suggested using this method, he also provided a pointer to a more robust method: using the `sverb` package that is part of 'mdwtools' collection (specifically, the `\ignoreenv{}` command).

```
  \ifExcludeSong\setbox2=\vbox\bgroup\fi%
```

Try to keep the song title and all its contents on the same page; if that is what is desired.

```
  \ifSamepageMode%
    \begin{samepage}%
  \fi%
```

CompactSong Font Processing Downsize fonts to allow song to fit into half the space (i.e., two column mode); although the title will not be reset since it will be presented unchanged from normal chordbk mode.

```
  \ifCompactSongMode
    \renewcommand{\ChBassFontSav}{\ChBassFont} %
    \renewcommand{\ChFontSav}{\ChFont} %
    \renewcommand{\ChBkFontSav}{\ChBkFont} %
    \renewcommand{\SBDefaultFontSav}{\SBDefaultFont} %
    \renewcommand{\SBOccursBrktFontSav}{\SBOccursBrktFont}%
    %
    \renewcommand{\ChBassFont}{\ChBassFontCS} %
    \renewcommand{\ChFont}{\ChFontCS} %
    \renewcommand{\ChBkFont}{\ChBkFontCS} %
    \renewcommand{\SBDefaultFont}{\SBDefaultFontCS} %
    \renewcommand{\SBOccursBrktFont}{\SBOccursBrktFontCS}%
%
%   Multicol specific changes.
%     \begin{macrocode}
    \setlength{\columnsep}{0.25in}

```

Remove side-margin, since marginal notes are not allowed when using multicol.sty; but, we save their values before changing them.

```
    \setlength{\textwidthSav}     {\textwidth}
    \setlength{\evensidemarginSav}{\evensidemargin}
    \setlength{\marginparsepSav}  {\marginparsep}
    \setlength{\marginparwidthSav}{\marginparwidth}
    %
    \addtolength{\textwidth}     {\marginparsep + \marginparwidth}
    \addtolength{\evensidemargin}{-\marginparsep - \marginparwidth}
    \setlength  {\marginparsep}  {0in}
    \setlength  {\marginparwidth}{0in}

```

Reduce minimum spacing amount used in `\Chr` macro (since we're now using a smaller font for lyrics and chords.

```
    \setlength{\chSpaceToleranceSav}{\chSpaceTolerance}
    %
    \setlength{\chSpaceTolerance}{1.0mm}

```

Remove the extra space before Verses, etc.

```
    \renewcommand{\HangAmtSav}            {\HangAmt}
    \renewcommand{\LeftMarginSBChorusSav} {\LeftMarginSBChorus}
    \renewcommand{\LeftMarginSBSectionSav}{\LeftMarginSBSection}
    \renewcommand{\LeftMarginSBVerseSav}  {\LeftMarginSBVerse}
    %
    \renewcommand{\HangAmt}            {1.5em}
    \renewcommand{\LeftMarginSBChorus} {2em}
    \renewcommand{\LeftMarginSBSection}{\LeftMarginSBChorus}
    \renewcommand{\LeftMarginSBVerse}  {\LeftMarginSBChorus}
  \fi
```

Whereever you see a parameter used directly, and not the parameter macro just set, above, it is because I haven't figured out how the receiving macro can deal with accepting its input via a macro (and not via the native parameter). In general this is because the receiving macro is attempting to detect and empty or blank parameter.

The second parameter is used directly here when `\STitle` is invoked (instead of `\theSongKey`), because I can't figure out how to cause the sharp and flat substitution to occur within the context of the `\renewcommand` statement, above.

```
  \begin{center}
    \STitle[N]{\theSongTitle}{#3}\\
    \vspace{-.5ex}
    \CpyRt[N]{#4}{#7}\\
    \vspace{-.5ex}
    \WAndM[N]{#5}\\
    \if\given{#6}%
      \vspace{-.75ex}
      \ScriptRef[N]{\theSongScriptRef}\\
    \fi%
  \end{center}%
  \vspace{\SpaceAfterTitleBlk}
```

If we're in `compactsong` mode then put us into `multicols{2}` mode.

```
  \ifCompactSongMode
    \begin{multicols*}{2}
      \raggedcolumns
  \fi
  \SBDefaultFont%
  }%
```

This brings the `song` environment's open clause to a close.

The close clause now starts. We begin by closing out the `SamepageMode` and `CompactSongMode` environments, as applicable. For `CompactSongMode` we need to restore the fonts and lengths.

```
{\ifSamepageMode%
    \end{samepage}%
  \fi%
  \ifCompactSongMode
    \renewcommand{\ChBassFont}{\ChBassFontSav} %
    \renewcommand{\ChFont}{\ChFontSav} %
    \renewcommand{\ChBkFont}{\ChBkFontSav} %
    \renewcommand{\SBDefaultFont}{\SBDefaultFontSav} %
    \renewcommand{\SBOccursBrktFont}{\SBOccursBrktFontSav}%
    %
    \setlength{\textwidth}     {\textwidthSav}
    \setlength{\evensidemargin}{\evensidemarginSav}
    \setlength{\marginparsep}  {\marginparsepSav}
    \setlength{\marginparwidth}{\marginparwidthSav}
    %
    \renewcommand{\HangAmt}            {\HangAmtSav}
    \renewcommand{\LeftMarginSBChorus} {\LeftMarginSBChorusSav}
    \renewcommand{\LeftMarginSBSection}{\LeftMarginSBSectionSav}
    \renewcommand{\LeftMarginSBVerse}  {\LeftMarginSBVerseSav}
    %
    \end{multicols*}
  \fi
  \ifSongEject%
    \vfill\pagebreak%
  \else%
    \SpaceAfterSong\pagebreak[1]%
  \fi%
```

Here's where we close out the ⟨*Include?*⟩ if-then-else. Note that we immediately clear `box2` before proceding (an attempt to free up the memory we've just consumed).

```
  \ifExcludeSong\egroup\setbox2=\hbox{}\fi%
  \renewcommand{\SBinSongEnv}{\False}%
  \SBinSongEnvfalse%
  }

```

`\CBExcl` `\OHExcl` `\WBExcl` `\WOExcl`

The `\CBExcl`, `\OHExcl`, `\WBExcl`, and `\WOExcl` macros exist to be passed as parameters to the `song` environment's ⟨*Include?*⟩ parameter. The parameters cause the song to be excluded when processing the particular Song♭ook type:

`CBExcl`	Exclude the song when in `chordk` mode
`OHExcl`	Exclude the song when in `overhead` mode
`WBExcl`	Exclude the song when in `wordbk` mode
`WOExcl`	Exclude the song when in either `wordbk` or `overhead` mode

Here's an example usage which shows a song to be excluded when in `chordbk` mode:

```
\documentclass{book}
\usepackage[chordbk]{songbook}

\begin{document}
  \begin{song}[\CBExcl]{title}{}{}{}{}{}
    some lyrics
  \end{song}
\end{document}
```

```
\newcommand{\CBExcl}{\ifChordBk N\else Y\fi}
\newcommand{\OHExcl}{\ifOverhead N\else Y\fi}
\newcommand{\WBExcl}{\ifWordBk N\else Y\fi}
\newcommand{\WOExcl}{\ifWordsOnly N\else Y\fi}
```

`xlatn`

`xlatn[`⟨*1*⟩`][`⟨*2*⟩`][`⟨*3*⟩`]` is the old song-translation environment.

Parameters:
⟨*1*⟩ Title of the translated song.
⟨*2*⟩ Translation permission.
⟨*3*⟩ Who performed the translation.

The `xlatn` environment is considered obsolete and will be removed from a future release of the Song♭ook macros.

The xlatn environment always occurs within a song environment. We reset the verse counter then cause the title and other parameter information to be displayed.

```
\newenvironment{xlatn}[3]{% Comment marker negates the newline.
        \renewcommand{\theXlatnBy}{#3}%
        \renewcommand{\theXlatnPerm}{#2}%
        \renewcommand{\theXlatnTitle}{#1}%
        %
        \titleIndex{\theXlatnTitle}{\theSBSongCnt}%
        \titleContents{\theXlatnTitle}{\theSongKey}%
        %
        \begin{center}
          \STitle[N]{\theXlatnTitle}{\theSongKey}\\
          \CpyRt[N]{\theSongCopyRt}{\theSongLicense}\\
          \if\nil{#2}%
            \relax%
          \else%
            \vspace{-.5ex}
            {\CpyRtFont\theXlatnPerm}\\
          \fi
          \if\nil{#3}%
            \relax%
          \else%
            \vspace{-.5ex}
            {\CpyRtFont\theXlatnBy}\\
          \fi
        \end{center}%
        %
        \setcounter{SBVerseCnt}{0}%
        \setcounter{SBSectionCnt}{0}%
}{\relax}
```

`songTranslation` `songTranslation[`⟨*1*⟩`][`⟨*2*⟩`][`⟨*3*⟩`]` is the song-translation environment.

Parameters:
- ⟨*1*⟩ Language of translated song.
- ⟨*2*⟩ Title of the translated song.
- ⟨*3*⟩ Translation permission.
- ⟨*4*⟩ Who performed the translation.

The `songTranslation` environment always occurs within a `song` environment. We reset the verse counter then cause the title and other parameter information to be displayed.

The `xlatn` environment was the original song translation environment, but with the addition of an additional parameter (the “Language of translated song.” parameter) it made most sense to create a new environment which simply deals with the new parameter and then calls the old environment. At some point in the future, when the `xlatn` environment is removed, the `xlatn` code will be moved here.

```
\newenvironment{songTranslation}[4]{% Comment marker negates the newline.
        \renewcommand{\theXlatnBy}{#4}%
\begin{xlatn}{#2}{#3}{#4}%
}{\end{xlatn}}
```

`\sbChord` `\sbChord` changes a sequence of characters into a chord.

Parameters:
- ⟨*1*⟩ Chord.

The original version of this function was written by Philip Hirschhorn <`psh@math.mit.edu`> or <`phirschhorn@lucy.wellesley.edu`>.

Scan the sequence of characters in Chord. Replace ‘`#`’ characters with ♯’s and ‘`b`’ characters with ♭. This produces more realistic looking chord symbols (which also take up less space than their phoney counterparts). We also look for ‘`/`’ characters, and insert a `\ChBassFont` command into the stream when a ‘`/`’ is found. This makes the bass note of the chord to appear in a smaller font.

```
\def\sbChord#1{%
  \ifx#1\relax%
    \let\next=\relax%
  \else%
    \ifx#1##% double sharp because we’re inside a \def
      $\sharp$%
    \else%
      \ifx#1b%
        $\flat$%
      \else%
        \ifx#1/%
          \ChBassFont /%
        \else%
          \ifx#1[%
            \bgroup\ChBkFont [\egroup%
          \else%
            \ifx#1]%
              \bgroup\ChBkFont ]\egroup%
            \else%
              #1%
            \fi%
          \fi%
        \fi%
      \fi%
    \fi%
    \let\next=\sbChord%
  \fi%
  \next%
}

```

`\Ch` `\Ch[`⟨*1*⟩`][`⟨*2*⟩`]` is the chord over lyrics macro.

`\ChX` `\ChX` is the Chord over lyrics macro, but deleting trailing spaces.

`\Chr` `\Chr` is the Chord over lyrics macro, but inserting a rule, when necessary.

Parameters:
⟨*1*⟩ Chord.
⟨*2*⟩ Syllable that chord is to be left justified over.

The words-only style file turns off the chord generation and just prints the second parameter.

The `\ChX` version of this macro is used for the benefit of the words-only style to ensure that spaces following the macro are removed. For example, an interword space containing a couple of extra chords would be written as (this is not usually necessary, but sometimes there is no other way to elliminate spurious white space from a words-only songbook):

```
\ChX{D7}{ing} \ChX{E}{} \ChX{D}{} \Ch{A}{You}
```

The `\Chr` version of this macro inserts a rule, at the height specified by the `\SBRuleRaiseAmount` macro, when the chord is wider than the syllable. The default value creates an extended em-dash-like rule; a value of 0pt creates an underbar-like rule. More details about the `\Chr` command follow below, just preceeding its definition.

This code is based on macros from Olivier Biot's (`http://www.biot.yucom.be/`) `chord.sty` file. Changes made by me:

- removed annoying space between `\SBIntersyllableRules` when they butt up against one another
- changed the default `\ChordRaise` value to something closer to what my previous version of the `\Ch` command used to set
- renamed the commands: `\@` to `\Ch`, and `\@@` to `\Chr`
- renamed the variables used to adjust `\Ch`'s behaviour, to ensure no conflict exists with Olivier's macro.

```
\newcommand{\Ch}[2]{{%
  \ifChordBk%
    \setbox1=\hbox{\ChFont\sbChord#1\relax\strut}%
    \setbox0=\hbox{#2}%
    \ifdim\wd1<\wd0%
      \strut\raise\SBChordRaise\copy1\kern-\wd1\copy0%
    \else%
      \strut\copy0\kern-\wd0\strut\raise\SBChordRaise\copy1%
    \fi%
  \else%
    #2%
  \fi}}

```

The `\ChX` code.

```
\newcommand{\ChX}[2]{%
  \ifWordsOnly%
    \if\nil{#2}%
      \ignorespaces%
    \else%
      #2%
    \fi%
  \else%
    \Ch{#1}{#2}%
  \fi}

```

The `\Chr` code and a detailed macro description & definition.

We start with some internal scratch variables. Any value they have prior to `\Chr`'s execution will be discarded each time.

```
\newlength{\chCriticDim}
\newlength{\chSpaceDim}
```

```
DEF\Chr#1#2
BEGIN
  \box1 == \hbox{... #1 --> Chord ...}
  \box0 == \hbox{... #2 --> Syllable ...}
  \chCriticDim == \wd0 - \chSpaceTolerance - 2 \chMiniSpace
  IFF \wd1 > \chCriticDim
    \chCriticDim == \wd1 - \wd0 - \chSpaceTolerance - 2 \chMiniSpace
    IFF \chCriticDim > 0mm
      \chSpaceDim == \wd1 - \wd0 + \chSpaceTolerance
    ELSE
      \chSpaceDim == \chSpaceTolerance
    FFI
    \chCriticDim == \chSpaceDim - 2 \chSpaceTolerance
    \raise \SBChordRaise \copy1 \kern - \wd1
    IFF \wd0 == 0mm
      \kern - 2 \chMiniSpace
    FFI
    \copy0
    \hbox to \chCriticDim{\hss\raise\SBRuleRaiseAmount
                          \hbox to \chSpaceDim{\SBIntersyllableRule}\hss}
  ELSE
    \raise \SBChordRaise \copy1 \kern - \wd1 \copy0
  FFI
END
```

```
\newcommand{\Chr}[2]{{%
  \ifChordBk
    \setbox1=\hbox{\ChFont\sbChord#1\relax\strut}%
    \setbox0=\hbox{#2}%
    \setlength{\chCriticDim}{\wd0 - \chSpaceTolerance}%
    \advance\chCriticDim by 2\chMiniSpace%
    \ifdim\wd1>\chCriticDim%
      \chCriticDim \wd1%
      \advance\chCriticDim by -\wd0%
      \advance\chCriticDim by -\chSpaceTolerance%
      \advance\chCriticDim by -2\chMiniSpace%
      \ifdim\chCriticDim>0mm%
        \chSpaceDim \wd1%
        \advance\chSpaceDim by -\wd0%
        \advance\chSpaceDim by \chSpaceTolerance%
      \else%
        \chSpaceDim\chSpaceTolerance%
      \fi%
      \chCriticDim \chSpaceDim%
      \advance\chCriticDim by 2\chMiniSpace%
      \strut\raise\SBChordRaise\copy1\kern-\wd1\ifdim\wd0=0mm\kern-2\chMiniSpace\fi%
      \copy0\hbox to\chCriticDim{\hss%
        \raise\SBRuleRaiseAmount\hbox to\chSpaceDim{\SBIntersyllableRule}\hss}%
    \else%
      \strut\raise\SBChordRaise\copy1\kern-\wd1%
      \copy0%
    \fi%
  \else%
    #2%
  \fi}%
}

```

`\SBMargNote` `\SBMargNote[⟨1⟩]` creates a Songbook marginal note.

Parameters:

⟨*1*⟩ Text of note to place in margin.

Used to place a note of some kind in the margin of a songbook, or within a footnote when in `CompactSong` mode. In words-only mode this macro is a no-op.

If we are excluding a song then we have `\SBMargNote` take no action. We do this to be sure that no footnotes are generated, and to prevent the error that will occur from attempting to use the `\marginpar` command within a `\vbox{}`.

```
\newcommand{\SBMargNote}[1]{%
  \ifExcludeSong%
```

```
    \relax%
  \else\ifWordsOnly%
    \relax%
  \else\ifCompactSongMode%
    \footnote{{\SBMargNoteFont{#1}}}%
  \else%
    \marginpar{{\begin{flushleft}\SBRefFont{#1}\end{flushleft}}}%
  \fi\fi\fi}

```

`\SBRef`

`\SBRef` creates a song reference in the margin.

Parameters:

- ⟨*1*⟩ Songbook/CD/tape name.
- ⟨*2*⟩ Page/Song number within book referenced by ⟨*1*⟩, or tape/CD publisher info.

Used to indicate a source for the full SATB music for this song, or what CD/cassette the song can be found on. In words-only mode this macro is a no-op. This normally appears in the margin of the songbook, but in `CompactSong` mode the information appears in a footnote that is always numbered '`0`' (even if there is more than one reference in a song.

If we are excluding a song then we have `\SBRef` take no action. We do this to be sure that no footnotes are generated, and to prevent the error that will occur from attempting to use the `\marginpar` command within a `\vbox{}`.

```
\newcommand{\SBRef}[2]{%
  \ifExcludeSong%
    \relax%
  \else\ifWordsOnly%
    \relax%
  \else\ifCompactSongMode%
    \footnotetext[0]{{\SBRefFont{\em #1}, {#2}.}}%
  \else%
    \marginpar{{\begin{flushleft}\SBRefFont{\em #1}, {#2}.\end{flushleft}}}%
  \fi\fi\fi}

```

`SBVerse`
`SBVerse*`

The `SBVerse` and `SBVerse*` environments encapsulate a verse.

Parameters:

None.

Very much like LaTeX's verse environment, except that here the verses are numbered. The indent amount for lines that are too long is set with the `\HangAmt` command (see the constant definitions at the top of this document).

A version of this command which indents but down not place an `\SBVerseCnt` before the chorus is available as `SBVerse*`. Similar to LaTeX's `\section*` command, the verse counter is not incremented either.

```
\newenvironment{SBVerse}{%
  \sbSetsbBaselineSkipAmt%
  \bgroup%
  \addtocounter{SBVerseCnt}{1}%
  \SBVerseMarkright{\theSBVerseCnt}%
  \begin{list}{{\SBVerseNumberFont\theSBVerseCnt .}}
    {\setlength {\leftmargin}   {\LeftMarginSBVerse + \HangAmt}
      \setlength{\itemindent}   {-\HangAmt}
      \setlength{\listparindent}{-\HangAmt}
      \setlength{\parsep}       {0pt}
      \setlength{\baselineskip} {\sbBaselineSkipAmt}
    }%
    \item}
{\end{list}%
 \egroup%
 \SpaceAfterVerse}

```

The `SBVerse*` code. Coding of this environment courtesy of Herbert Martin Dietze <`herbert@fh-wedel.de`>.

```
\newenvironment{SBVerse*}{%
  \sbSetsbBaselineSkipAmt%
  \bgroup%
  \begin{list}{{\SBVerseNumberFont }}
    {\setlength {\leftmargin}   {\LeftMarginSBVerse + \HangAmt}
      \setlength{\itemindent}   {-\HangAmt}
      \setlength{\listparindent}{-\HangAmt}
      \setlength{\parsep}       {0pt}
      \setlength{\baselineskip} {\sbBaselineSkipAmt}
      }%
    \item}
{\end{list}%
 \egroup%
 \SpaceAfterVerse}

```

`SBSection`
`SBSection*`

The `SBSection` and `SBSection*` environments encapsulate a section.

Parameters:

None.

Very much like LaTeX's verse environment, except that here the sections are numbered. The indent amount for lines that are too long is set with the `\HangAmt` command (see the constant definitions at the top of this file).

A version of this command which indents but doesn't place an `\SBSectionCnt` before the chorus is available as `SBSection*`. Similar to LaTeX's `\section*` command, the section counter is not incremented either.

```
\newenvironment{SBSection}{%
  \sbSetsbBaselineSkipAmt%
  \bgroup%
  \addtocounter{SBSectionCnt}{1}%
  \SBSectionMarkright{SBSectionCnt}
  \begin{list}{{\SBSectionNumberFont\alph{SBSectionCnt})}}
    {\setlength {\leftmargin}   {\LeftMarginSBSection + \HangAmt}
      \setlength{\itemindent}   {-\HangAmt}
      \setlength{\listparindent}{-\HangAmt}
      \setlength{\parsep}       {0pt}
      \setlength{\baselineskip} {\sbBaselineSkipAmt}
      }%
    \item}
{\end{list}%
 \egroup%
 \SpaceAfterSection}

```

The `SBSection*` code. Coding of this environment courtesy of Herbert Martin Dietze <`herbert@fh-wedel.de`>.

```
\newenvironment{SBSection*}{%
  \sbSetsbBaselineSkipAmt%
  \bgroup%
  \begin{list}{{\SBSectionNumberFont }}
    {\setlength {\leftmargin}   {\LeftMarginSBSection + \HangAmt}
      \setlength{\itemindent}   {-\HangAmt}
      \setlength{\listparindent}{-\HangAmt}
      \setlength{\parsep}       {0pt}
      \setlength{\baselineskip} {\sbBaselineSkipAmt}
      }%
    \item}
{\end{list}%
 \egroup%
 \SpaceAfterSection}

```

`\SBChorus`
`\SBChorus*`

The `SBChorus` and `SBChorus*` environments encapsulate a chorus.

Parameters:

None.

Very much like LaTeX's verse environment, except that here a `\SBChorusTag` tag is inserted to demark the start of the chorus. The indent amount for lines that are too long is set with the `\HangAmt` command (see the constant definitions at the top of this file).

A version of this command which indents but does not place a `\SBChorusTag` before the chorus is available as `SBChorus*`.

```
\newenvironment{SBChorus}{%
  \sbSetsbBaselineSkipAmt%
  \bgroup%
  \SBChorusMarkright{\SBChorusTag}
  \begin{list}{{\SBChorusTagFont\SBChorusTag}}
    {\setlength {\leftmargin}   {\LeftMarginSBChorus + \HangAmt}
      \setlength{\itemindent}   {-\HangAmt}
      \setlength{\listparindent}{-\HangAmt}
      \setlength{\parsep}       {0pt}
      \setlength{\baselineskip} {\sbBaselineSkipAmt}
      }%
    \item}
{\end{list}%
 \egroup%
 \SpaceAfterChorus%
}

```

The `SBChorus*` code. Coding of this environment courtesy of Herbert Martin Dietze <`herbert@fh-wedel.de`>.

```
\newenvironment{SBChorus*}{%
  \sbSetsbBaselineSkipAmt%
  \bgroup%
  \begin{list}{{\SBChorusTagFont }}
    {\setlength {\leftmargin}   {\LeftMarginSBChorus + \HangAmt}
      \setlength{\itemindent}   {-\HangAmt}
      \setlength{\listparindent}{-\HangAmt}
      \setlength{\parsep}       {0pt}
      \setlength{\baselineskip} {\sbBaselineSkipAmt}
      }%
    \item}
{\end{list}%
 \egroup%
 \SpaceAfterChorus}

```

`\SBOpGroup`

`\SBOpGroup` identifies an open chorus/verse.

Parameters:

None.

This environment is akin to SBChorus, except that no tag and no indentation is performed. This environment serves two purposes:

1. Identify a verse or chorus that is unmarked (by way of a tag) and the left margin of the block is not indented.
2. Puts the verse or chorus in a list environment so that wrapping lines are properly indented.

```
\newenvironment{SBOpGroup}{%
  \sbSetsbBaselineSkipAmt%
  \bgroup%
  \begin{list}{\hbox{}}
    {\setlength {\leftmargin}   {\HangAmt}
      \setlength{\itemindent}   {-\HangAmt}
      \setlength{\listparindent}{-\HangAmt}
      \setlength{\topsep}       {0pt}
      \setlength{\parsep}       {0pt}
      \setlength{\labelwidth}   {0pt}
      \setlength{\labelsep}     {0pt}
      \setlength{\baselineskip} {\sbBaselineSkipAmt}
      }%
    \item}
{\end{list}%
```

```
 \egroup%
 \SpaceAfterOpGroup}

```

`\SBBridge` `\SBBridge[`⟨*1*⟩`]` identifies a bridge.

Parameters:

⟨*1*⟩ The Bridge.

This command is used to encapsulate a bridge that occurs in a song. In words-only mode this command is a no-op.

```
\newcommand{\SBBridge}[1]{%
  \ifWordsOnly%
    \relax%
  \else%
    \sbSetsbBaselineSkipAmt%
    \bgroup%
    \begin{list}{{\SBBridgeTagFont\SBBridgeTag}}
      {\setlength {\leftmargin}  {\LeftMarginSBChorus}%
        \setlength{\parsep}      {0pt}
        \setlength{\baselineskip}{\sbBaselineSkipAmt}
        }%
      \item #1
    \end{list}%
    \egroup\par
  \fi}

```

`\SBEnd` `\SBEnd[`⟨*1*⟩`][`⟨*2*⟩`]` identifies a song ending.

Parameters:

⟨*1*⟩ Display in words-only? (optional)

⟨*2*⟩ The Ending.

This command is used to encapsulate the ending of a song. If the first parameter is not specified, or if it is '`N`', then in words-only mode this command is a no-op.

```
\newcommand{\SBEnd}[2][N]{%
  \ifthenelse{\equal{\ifWordsOnly Y\fi}{Y}
    \and \equal{N}{#1}}%
  {\relax}%
  {\sbSetsbBaselineSkipAmt%
   \bgroup%
    \begin{list}{{\SBEndTagFont\SBEndTag}}
      {\setlength {\leftmargin}  {\LeftMarginSBChorus}
        \setlength{\parsep}      {0pt}
        \setlength{\baselineskip}{\sbBaselineSkipAmt}
        }%
    \item #2
    \end{list}%
    \egroup\par}
  }

```

`\SBIntro` `\SBIntro[`⟨*1*⟩`][`⟨*2*⟩`]` identifiesd an introduction.

Parameters:

⟨*1*⟩ Display in words-only? (optional)

⟨*2*⟩ The Introduction.

This command is used to encapsulate an introduction to a song. If the first parameter is not specified, or if it is '`N`', then in words-only mode this command is a no-op.

```
\newcommand{\SBIntro}[2][N]{%
  \ifthenelse{\equal{\ifWordsOnly Y\fi}{Y}
    \and \equal{N}{#1}}%
  {\relax}%
  {\sbSetsbBaselineSkipAmt%
   \bgroup%
    \begin{list}{{\SBIntroTagFont\SBIntroTag}}%
```

```
      {\setlength {\leftmargin}  {\LeftMarginSBChorus}%
        \setlength{\parsep}      {0pt}
        \setlength{\baselineskip}{\sbBaselineSkipAmt}
        }%
      \item #2
      \vspace{-\topsep}%\vspace{-\partopsep}%
    \end{list}%
    \egroup\par}%
  }

```

SBBracket
SBBracket*

The `SBBracket`[⟨*1*⟩] and `SBBracket`[⟨*1*⟩] environments encapsulates a bracketed versicle.

Parameters:

⟨*1*⟩ Some tag is inserted before the bracket to indicate the significance of the bracketed area.

There are two versions of this environment: `SBBracket` and `SBBracket*`. They operate identically, except that the *ed version doesn't print its tag and bracket in words-only modes.

This is a more versatile, and better formatted version of `SBBridge`, `SBOccurs`, etc.; and it is recommended that this be used in the others place.

Starting in version 4.0 of the style, the left-hand indentation of this environment has been chosen such that the `SBVerse`, `SBChorus`, and `SBBracket` songwords all align against the same left margin when printing standard words & chords songbooks.

```
\newenvironment{SBBracket}[1]{%
  \SpaceBeforeSBBracket
  \sbSetsbBaselineSkipAmt%
  \setbox0=\hbox to \LeftMarginSBBracket{\parbox{\LeftMarginSBBracket}%
    {\flushright{\hspace{0pt}\SBBracketTagFont #1}}}%
  \hbox\bgroup%
    \rightskip=\LeftMarginSBBracket%
    $\raisebox{1.25ex}{\copy0}%
    \left\lbrack%
      \vcenter\bgroup%
        \begin{list}{\hbox{}}%                          %
          {\setlength {\leftmargin}   {\HangAmt + 0.5em}% This list
            \setlength{\rightmargin}  {\LeftMarginSBBracket}%
            \setlength{\itemindent}   {-\HangAmt}%      % been copied
            \setlength{\listparindent}{-\HangAmt}%      % verbatim from
            \setlength{\topsep}       {0pt}%            % the SBOpGroup
            \setlength{\parsep}       {0pt}%            % environment,
            \setlength{\labelwidth}   {0pt}%            % above and then
            \setlength{\labelsep}     {0pt}%            % modified slightly.
            \setlength{\baselineskip} {\sbBaselineSkipAmt}%
            }%                                          %
          \item%
}{%
        \end{list}%
      \egroup%
    \right.$%
    \rightskip=0pt
  \egroup
  \SpaceAfterSBBracket
}

```

The `SBBracket*` code.

```
\newenvironment{SBBracket*}[1]{%
  \SpaceBeforeSBBracket
  \sbSetsbBaselineSkipAmt%
  \ifNotWordsOnly
    \setbox0=\hbox to \LeftMarginSBBracket{\parbox{\LeftMarginSBBracket}%
      {\flushright{\hspace{0pt}\SBBracketTagFont #1}}}%
    \hbox\bgroup%
      \rightskip=\LeftMarginSBBracket%
      $\raisebox{1.25ex}{\copy0}%
      \left\lbrack%
        \vcenter\bgroup%
  \fi
        \begin{list}{\hbox{}}%                          %
          {\setlength {\leftmargin}   {\HangAmt + 0.5em}% This list
            \setlength{\rightmargin}  {\LeftMarginSBBracket}%
            \setlength{\itemindent}   {-\HangAmt}%      % been copied
            \setlength{\listparindent}{-\HangAmt}%      % verbatim from
            \setlength{\topsep}       {0pt}%            % the SBOpGroup
            \setlength{\parsep}       {0pt}%            % environment,
            \setlength{\labelwidth}   {0pt}%            % above and then
            \setlength{\labelsep}     {0pt}%            % modified slightly.
            \setlength{\baselineskip} {\sbBaselineSkipAmt}%
            }%                                          %
          \item%
}{%
        \end{list}%
  \ifNotWordsOnly
        \egroup%
      \right.$%
      \rightskip=0pt
    \egroup
  \fi
  \SpaceAfterSBBracket
}

```

`SBOccurs` The `SBOccurs[`⟨*1*⟩`]` environment encapsulates an occurance.

Parameters:

⟨*1*⟩ Occurance number(s). For example "`1,3`" would designate that this passage applies to the 1^{st} and 3^{rd} occurances.

```
\newenvironment{SBOccurs}[1]{%
  {\SBOccursTagFont #1\SBOccursBrktFont [}
  }
{{\SBOccursBrktFont ]}}

```

`SBExtraKeys` The `SBExtraKeys[`⟨*1*⟩`]` environment encapsulates extra song keys.

Parameters:

⟨*1*⟩ This parameter actually is used to either pass or not pass all the content of the environment on to the LaTeXprocessor.

Songs are frequently listed in more than one key. This is ok for books with chords, however the words-only edition should only print one occurance of a song. So, any extra keys are placed in a `SBExtraKey` environment. This allows them to be *shut off* when they're not needed.

This was coded some years ago and I probably wouldn't do it this way again; however, it works so I'm not inclined to *better* it.

```
\newenvironment{SBExtraKeys}[1]{%
  \ifWordsOnly%
    \relax%
  \else%
    #1
  \fi}
{}

```

`\CBPageBrk` `\CBPageBrk[`⟨*1*⟩`]` generates a page break here if we're in Chordbk mode.

Parameters:

⟨*1*⟩ Take effect in CompactSong mode too? (optional)

When we're also in CompactSong mode we will only execute the page break if a parameter other than '`N`' has been passed.

```
\newcommand{\CBPageBrk}[1][N]{%
  \ifChordBk%
    \ifCompactSongMode
```

```
      \ifthenelse{\equal{#1}{N}}
      {\relax}
      {\vfill\pagebreak}
    \else
      \vfill\pagebreak
    \fi
  \fi}

```

\CSColBrk

`\CSColBrk` generates a column break here if we're in `compactsong` mode.

Parameters:

None.

```
\newcommand{\CSColBrk}{%
  \ifCompactSongMode%
    \columnbreak%
  \fi}

```

\NotWOPageBrk

`\NotWOPageBrk` generates a page break here if we're *not* in words-only mode.

Parameters:

None.

```
\newcommand{\NotWOPageBrk}{%
  \ifWordsOnly%
    \relax%
  \else%
    \pagebreak
  \fi}

```

\OHPageBrk

`\OHPageBrk` generates a page break here if we're in `overhead` mode.

Parameters:

None.

```
\newcommand{\OHPageBrk}{%
  \ifOverhead%
    \pagebreak
  \fi}

```

\WBPageBrk

`\WBPageBrk` generates a page break here if we're in `workbk` mode.

Parameters:

None.

```
\newcommand{\WBPageBrk}{%
  \ifWordBk%
    \pagebreak
  \fi}

```

\WOPageBrk

`\WOPageBrk` generates a page break here if we're in words-only mode.

Parameters:

None.

```
\newcommand{\WOPageBrk}{%
  \ifWordsOnly%
    \pagebreak
  \fi}

```

14.8.7 Obsolete Macros

The macros in this section are no longer recommended, but will continue to exist in the next version of the style. Existing users of this style should upgrade their source files to make use of the new, replacement, mechanisms offered by the style.

```
%%=====================================================%
%%            O B S O L E T E   M A C R O S            %
%%=====================================================%

```

The `xlatn` environment is obsolete, but for the sake of code-clarity the code has not been moved into the Obsolete Macros section of this document.

14.8.8 Deprecated Macros

The macros in this section will be deleted in the next version of the style. Where these old macros conflict with new ones they have been renamed by placing a lowercase ‘o’ at the start of each macro name; this makes them easily accessible yet out of the way.

```
%%=====================================================%
%%          D E P R E C A T E D   M A C R O S          %
%%=====================================================%

```

Boolean Contants In the early releases, before I *knew* about LaTeX’s `\newif` command I had coded `\if`s using these contants. These should have been removed some time ago, but I had neglected placing them into this *Deprecated Macros* section and so hadn’t given proper notice. Consider this *notice.*

`\False`
`\True`
`\ChordBk`
`\Overhead`
`\SongEject`
`\WordBk`
`\WordsOnly`
`\SBinSongEnv`

`\False` is defined for use in `\if` macro contructs and the other constants in this style.
`\True` is defined for use in `\if` macro contructs and the other constants in this style.
`\ChordBk` tells if we are processing a `chordbk.sty` document.
`\Overhead` tells if we are processing an `overhead.sty` document.
`\SongEject` specifies if we want to end the current page at the end of every `song` environment. A value of `\True` means eject after every `song` environment.
`\WordBk` tells if we are processing a `wordbk.sty` document
`\WordsOnly` is equal to `\True` if we’re in words-only mode. The default value will be `\False`, as that is how all of the commands in this file will act.
`\SBinSongEnv` tells if we are inside of a song environment. This is re-defined as we enter and exit the song environment.

```
\newcommand{\False}{0}
\newcommand{\True}{1}
\newcommand{\ChordBk}{\False}
\newcommand{\Overhead}{\False}
\newcommand{\SongEject}{\True}
\newcommand{\WordBk}{\False}
\newcommand{\WordsOnly}{\False}
\newcommand{\SBinSongEnv}{\False}

```

End of songbook.sty file.

```
\endinput

```