

Il pacchetto `songs`*

Kevin W. Hamlen

26 ottobre 2015

Sommario

Il pacchetto `songs` permette di produrre canzonieri contenenti testo e accordi, ma non spartiti musicali. Permette di compilare, a partire dallo stesso sorgente \LaTeX , libri con accordi, libri di canzoni e slide. Il pacchetto gestisce la trasposizione automatica degli accordi, le tablature per chitarra e una serie di indici tematici.

1 Introduzione

Il pacchetto `songs` per \LaTeX permette di creare canzonieri contenenti i testi delle canzoni e, eventualmente, gli accordi delle stesse. Lo stesso documento permette di creare un canzoniere per i cantanti, un libro con accordi per i musicisti e delle slide per la proiezione all'assemblea. Il software è pensato in particolare per la scrittura di musica religiosa, ma può essere efficacemente impiegato per ogni tipo di canzone.

2 Condizioni di utilizzo

Il pacchetto `songs` è un software libero; può essere distribuito e/o modificato secondo quanto previsto dalla GNU General Public License pubblicata dalla Free Software Foundation; si consideri la versione 2 della Licenza, o ogni versione successiva. Il testo della Licenza (in inglese) si trova nella sezione §15.

Questo pacchetto è distribuito nella speranza che possa rivelarsi utile, ma SENZA GARANZIE. Per avere maggiori dettagli si veda la GNU General Public License nella sezione §15. Una copia della licenza può anche essere richiesta alla Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

*Questo documento si riferisce alla versione v2.13 del pacchetto `songs`, © 2011 Kevin W. Hamlen, ed è distribuito secondo la versione 2 della GNU General Public License pubblicata dalla Free Software Foundation. La traduzione e i commenti per la versione italiana sono stati curati da Francesco Endrici. È stata tradotta soltanto la prima parte della documentazione. La parte relativa alla spiegazione del codice è rimasta in inglese. È stata rilasciata la versione 2.18 del pacchetto `Songs`. Sono state introdotte alcune nuove funzionalità, ma relative a comandi poco usati.

Questo software è sotto il copyright di © 2011 Kevin W. Hamlen. Per avere informazioni o scaricare l'ultima versione del programma si veda la pagina web del progetto:

<http://songs.sourceforge.net>

3 Un documento base

Per chi vuole iniziare subito a scrivere le proprie canzoni riportiamo un esempio di codice con una canzone e un indice. Partendo da questo esempio è possibile creare un intero canzoniere. Le istruzioni per la compilazione sono riportate dopo il codice.

```
\documentclass{article}
\usepackage[chorded]{songs}

\newindex{titleidx}{titleidx}
\noversenumbers

\begin{document}
\showindex{Complete Index of Songs}{titleidx}
\songsection{Worship Songs}

\begin{songs}{titleidx}
\begin{song}{Doxology}[by={Louis Bourgeois and Thomas Ken},
    sr={Revelation 5:13},
    cr={Public domain.},
    index={Praise God, from Whom all blessings flow}]
\beginverse
\[G]Praise God, \[D]from \[Em]Whom \[Bm]all \[Em]bless\[D]ings \[G]flow;
\[G]Praise Him, all \[D]crea\[Em]tures \[C]here \[G]be\[D]low;
\[Em]Praise \[D]Him \[G]a\[D]bove, \[G]ye \[C]heav'n\[D]ly \[Em]host;
\[G]Praise Fa\[Em]ther, \[D]Son, \[Am]and \[G/B G/C]Ho\[D]ly \[G]Ghost.
\[C]A\[G]men.
\endverse
\endsong
\end{songs}

\end{document}
```

Per compilare questo codice bisogna eseguire tre comandi. Prima si usa \LaTeX (si raccomanda `pdflatex`) per compilare il documento:

```
pdflatex mybook.tex
```

(dove `mybook.tex` è il nome dato al file che abbiamo scritto). Poi si usa il programma `songidx` (fornito col pacchetto `songs`) per generare gli indici:

```
songidx titleidx.sxd titleidx.sbx
```

Infine si ricompila il documento usando \LaTeX in modo da includere i dati appena creati con `songidx`.

```
pdflatex mybook.tex
```

Il documento creato avrà nome `mybook.pdf` se si è usato `pdflatex` o `mybook.dvi` se si è usato `latex`.

La figura 1 riporta come esempio la prima pagina di un canzoniere con accordi. Quando si genera un canzoniere con i soli testi, tutti gli accordi vengono omessi. Si veda la §4 per informazioni su come generare più canzonieri a partire dallo stesso codice.

4 Inizializzazione e opzioni

Ogni documento \LaTeX che usi il pacchetto **songs** deve contenere nel preambolo una riga come quella che segue:

```
\usepackage[\langle options \rangle]{songs}
```

Le possibili *\langle options \rangle* sono:

<code>lyric</code>	Tipi di output. Il pacchetto songs può produrre quattro tipi di output: libri di soli testi, libri con accordi, slide e testo grezzo. Si può specificare il risultato desiderato scegliendo fra le opzioni <code>lyric</code> , <code>chorded</code> , <code>slides</code> o <code>rawtext</code> . Se non viene specificato nulla, <code>chorded</code> è l'opzione di default.
<code>chorded</code>	
<code>slides</code>	
<code>rawtext</code>	
<code>\chordson</code>	La stampa degli accordi può essere attivata e disattivata anche all'interno del documento usando le macro <code>\chordson</code> e <code>\chordsoff</code> .
<code>\chordsoff</code>	
<code>\slides</code>	La modalità slide può essere attivata anche all'interno di un documento con la macro <code>\slides</code> . Per ottenere un buon risultato però questa operazione dovrebbe essere effettuata nel preambolo o all'inizio di una nuova pagina.
<code>nomeasures</code>	Stanghette. Il pacchetto songs permette di inserire nel testo delle stanghette verticali per segnare l'inizio e la fine di una battuta musicale (si veda §7.7). Per non stampare le stanghette si deve usare l'opzione <code>nomeasures</code> ; per mostrare le stanghette si deve usare l'opzione <code>showmeasures</code> (di default). Le stanghette possono essere attivate e disattivate all'interno di un documento con le macro <code>\measureson</code> e <code>\measuresoff</code> .
<code>showmeasures</code>	
<code>\measureson</code>	
<code>\measuresoff</code>	
<code>transposecapos</code>	Trasposizione L'opzione <code>transposecapos</code> modifica l'effetto della macro <code>\capo</code> . Solitamente, usando <code>\capo{<n>}</code> all'interno di una canzone si produce una nota testuale che suggerisce l'uso di un capotasto per chitarra da mettere sul tasto <i><n></i> . Se l'opzione <code>transposecapos</code> è attiva queste note vengono tralasciate e l'effetto di <code>\capo{<n>}</code> è lo stesso di <code>\transpose{<n>}</code> . Cioè tutti gli accordi compresi fra <code>\capo</code> e la fine della canzone vengono automaticamente trasposti di <i><n></i> semitoni. Questo può tornare utile per creare canzonieri che siano usabili da chitarristi e, con una semplice modifica al codice, dai pianisti, che non hanno la possibilità di

Worship Songs

1

Doxology

Revelation 5:13
Louis Bourgeois and Thomas Ken

G *D* *Em* *Bm* *Em* *D*
Praise God, from Whom all blessings
G
flow;
G *D* *Em* *C* *G* *D*
Praise Him, all creatures here below;
Em *D* *G* *D* *G* *C* *D* *Em*
Praise Him a - bove, ye heav'nly host;
G *Em* *D* *Am*
Praise Father, Son, and
G/B *G/C* *D* *G*
Ho - ly Ghost.
C *G*
A - men.

Public domain.

*The LORD is my rock and my fortress
and my deliverer,
my God, my rock, in whom I take
refuge,
my shield, and the horn of my
salvation, my stronghold.
I call upon the LORD, who is worthy to
be praised,
and I am saved from my enemies.
The cords of death encompassed me;
the torrents of destruction assailed
me;
the cords of Sheol entangled me;
the snares of death confronted me.
In my distress I called upon the LORD;
to my God I cried for help.
From his temple he heard my voice,
and my cry to him reached his ears.*

Psalm 18:2-6

2

A Mighty Fortress Is Our God

Martin Luther

A *C#m* *B7* *E*
A mighty Fortress is our God,
D *A* *E7* *A*
A bulwark never fail - ing.
C#m *B7* *E*
Our helper He, amid the flood
D *A* *E7* *A*
Of mortal ills prevailing.
B7sus4 *B7* *E*
For still our an - cient foe
A *E/G#* *F#m*
Doth seek to work us woe;
B7 *E*
His craft and pow'r are great,
Bm *C#*
And, armed with cruel hate,
D *A* *E7* *A*
On earth is not his e - qual.
A *C#m* *B7* *E*
Did we in our own strength confide,
D *A* *E7* *A*
Our striving would be los - ing.
C#m *B7* *E*
Were not the right Man on our side,
D *A* *E7* *A*
The Man of God's own choosing.
B7sus4 *B7* *E*
Dost ask who that may be?
A *E/G#* *F#m*
Christ Jesus, it is He;
B7 *E*
Lord Sabaoth His Name,
Bm *C#*
From age to age the same;
D *A* *E7* *A*
And He must win the bat - tle.

Public Domain.

Figura 1: Esempio di pagina di un canzoniere con accordi.

usare un capotasto. Si vedano §7.8 and §10 per informazioni più dettagliate sulle macro `\capo` e `\transpose`.

`noindexes` **Indici.** L'opzione `noindexes` inibisce la creazione degli indici del documento.
`\indexeson` La visualizzazione degli indici può essere attivata e disattivata con `\indexeson` e
`\indexesoff` `\indexesoff`.
`nopdfindex` L'opzione `nopdfindex` inibisce la creazione dei segnalibri nel file `.pdf`. Se non
si genera un file `.pdf` questa opzione non ha effetto.

`noscripture` **Citazioni bibliche.** L'opzione `noscripture` omette le citazioni bibliche dal file
`\scriptureon` di output.(si veda §8.2). Le citazioni bibliche possono essere attivate e disattivate
`\scriptureoff` anche all'interno di un documento usando `\scriptureon` e `\scriptureoff`.

`noshading` **Celle colorate.** L'opzione `noshading` inibisce la creazione delle ombreggiature
nei numeri di canzone e nelle note testuali. Può essere comodo qualora queste
finestre diano problemi con la stampa o facciano consumare troppo inchiostro.

`\includeonlysongs` **Includere liste parziali di canzoni.** Spesso può essere comodo estrarre soltan-
to alcune canzoni da un documento master, per esempio per creare un libretto per
una particolare liturgia. Per fare ciò basta digitare `\includeonlysongs{<songlist>}`
nel preambolo del documento (cioè prima di `\begin{document}`), dove `<songlist>`
è una lista separata da virgole dei numeri delle canzoni che si vogliono includere.
Per esempio,

```
\includeonlysongs{37,50,2}
```

crea un documento contenente solo le canzoni 37, 50, e 2, in quest'ordine.

I documenti generati con `\includeonlysongs` non contengono le citazioni bi-
bliche (§8.2) e ignorano eventuali `\nextcol`, `\brk`, `\sclearpage`, e `\scleardpage`
presenti fra le canzoni, a meno che essi non siano seguiti da un asterisco (per
esempio, `\nextcol*`). Per indicare un'interruzione di colonna o di pagina in un
punto specifico di un documento creato con `\includeonlysongs` bisogna scri-
vere `nextcol`, `brk`, `sclearpage` o `scleardpage` nel punto corrispondente nella
`<songlist>`.

Il comando `\includeonlysongs` riordina le canzoni solo all'interno di un am-
biente `songs` (si veda §7), non in diversi ambienti `songs`. Non può essere usato
con l'opzione `rawtext`.

5 Le sezioni di un canzoniere

`\songsection` **Titoli di sezione.** I titoli di sezione di un canzoniere possono essere creati con
`\songchapter` `\songsection{<title>}`

che lavora come il comando `\section` di \LaTeX , se non che esso crea un `<title>`
centrato, in carattere senza grazie e non indica il numero di sezione. Se si usa la
classe `book`, si può usare `\songchapter` al posto di `\songsection`.

`\newindex` **Indici.** Il pacchetto **songs** può generare tre tipi di indice: indici per titolo, indici
`\newauthorindex` per autore e per riferimento biblico. Per creare un indice occorre prima dichiarare
`\newscripindex` l'indice stesso nel preambolo del documento, come segue:

```
\newindex{<id>}{<filename>}
\newauthorindex{<id>}{<filename>}
\newscripindex{<id>}{<filename>}
```

Il campo `<id>` è un identificatore alfabetico che verrà utilizzato per identificare l'indice in altre macro che lo richiamano. Il campo `<filename>` è una stringa di testo che possa essere usata come un nome file valido nel proprio sistema operativo. Durante la generazione degli indici vengono automaticamente generati i file ausiliari `<filename>.sxd` e `<filename>.sbx`. Ad esempio:

```
\newindex{mainindex}{idxfile}
```

crea un indice intitolato “mainindex” i cui dati sono salvati in due file nominati `idxfile.sxd` e `idxfile.sbx`.

`\showindex` Per inserire un indice in un documento si usa il comando:

```
\showindex[<columns>]{<title>}{<id>}
```

dove `<id>` è lo stesso identificatore usato in `\newindex`, `\newauthorindex` o `\newscripindex`, e `<title>` è il titolo dell'indice, che deve essere una stringa di solo testo (senza comandi di formattazione, che non possono essere usati nei segnalibri dei pdf).

La parte `[<columns>]` è opzionale; se specificata indica il numero di colonne da utilizzare se l'indice non sta tutto in una sola colonna. Ad esempio per un indice dei titoli a due colonne si scriverà:

```
\showindex[2]{Index of Song Titles}{mainindex}
```

6 Compilazione

Come buona parte dei documenti L^AT_EX, la compilazione di un canzoniere richiede tre passaggi. Primo: si usa L^AT_EX (preferibilmente `pdflatex`) per generare i file ausiliari dal file `.tex`:

```
pdflatex mybook.tex
```

Secondo: si usa il programma `songidx` per generare un indice per ogni indice dichiarato con `\newindex`, `\newauthorindex` o `\newscripindex`. La sintassi del comando `songidx` è:

```
songidx [-b <canon>.can] <filename>.sxd <filename>.sbx
```

dove $\langle filename \rangle$ è lo stesso $\langle filename \rangle$ usato nei comandi `\newindex`, `\newauthorindex` o `\newscripindex`. Se l'indice è stato dichiarato con `\newscripindex` occorre usare l'opzione `-b` per specificare a quale versione della Bibbia si vuol far riferimento per generare l'indice. La parte $\langle canon \rangle$ è uno dei vari file `.can` fornito con la distribuzione `songidx`. Per la Bibbia Protestante, Cattolica e Greca Ortodossa in inglese il file `bible.can` dovrebbe funzionare adeguatamente. Per altre bibbie occorre crearsi il proprio file `.can` copiando e modificando gli esistenti file `.can`.

Ad esempio, se il file `.tex` contiene le righe:

```
\newindex{titleidx}{titlfile}
\newauthorindex{authidx}{authfile}
\newscripindex{scripidx}{scrpfile}
```

i comandi per generare gli indici secondo la Bibbia Cristiana inglese sono:

```
songidx titlfile.sxd titlfile.sbx
songidx authfile.sxd authfile.sbx
songidx -b bible.can scrpfile.sxd scrpfile.sbx
```

Una volta generati gli indici si può generare il canzoniere finito richiamando \LaTeX per l'ultima volta:

```
pdflatex mybook.tex
```

7 Scrivere le canzoni

7.1 Iniziare una canzone

songs **Gruppi di canzoni.** Le canzoni devono essere contenute all'interno di un ambiente `songs`. Ogni ambiente `songs` inizia e finisce con:

```
\begin{songs}{\langle indexes \rangle}
:
\end{songs}
```

$\langle indexes \rangle$ è una lista separata da virgole di identificatori di indice (gli $\langle id \rangle$ specificati con `\newindex`)—un identificatore per ogni indice che si vuole includere nel canzoniere. Fra `\begin{songs}` e `\end{songs}` si possono trovare *solo* canzoni o ambienti `intersong` (si veda §8).

\beginsong **Le canzoni.** Una canzone inizia e finisce con:

```
\endsong \beginsong{\langle titles \rangle}[\langle otherinfo \rangle]
:
\endsong
```

Le canzoni possono essere scritte soltanto all'interno di ambienti `songs` a meno che non si decida di non usare il *page builder* (la serie di macro che in \LaTeX gestisce la distribuzione del testo e degli oggetti nella pagina) di `songs` e di usarne un altro (si veda §11.5).

Nella riga `\beginsong`, $\langle titles \rangle$ consiste in uno o più titoli separati da `\\`. Se vengono specificati più titoli, il primo viene stampato normalmente e gli altri vengono messi fra parentesi su righe separate

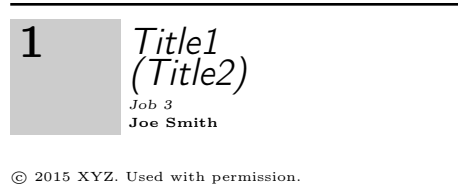
La parte $[\langle otherinfo \rangle]$ è opzionale. È una lista separata da virgole di valori nella forma $\langle key \rangle = \langle value \rangle$. Le possibili chiavi sono:

<code>by={\langle authors \rangle}</code>	<i>autori e compositori</i>
<code>cr={\langle copyright \rangle}</code>	<i>informazioni sul copyright</i>
<code>li={\langle license \rangle}</code>	<i>informazioni sulla licenza</i>
<code>sr={\langle refs \rangle}</code>	<i>riferimenti biblici della canzone</i>
<code>index={\langle lyrics \rangle}</code>	<i>una voce aggiuntiva per l'indice</i>
<code>ititle={\langle title \rangle}</code>	<i>una voce aggiuntiva per l'indice per un titolo nascosto</i>

Ad esmpio, una canzone che inizia e finisce con

```
\beginsong{Title1 \\ Title2}[by={Joe Smith}, sr={Job 3},
cr={\copyright~2015 XYZ.}, li={Used with permission.}]
\endsong
```

risulta



Le quattro chiavi usate nell'esempio sono descritte dettagliatamente nel seguito di questa sezione; le ultime due sono descritte nella §7.9. Si possono creare anche chiavi personalizzate (si veda §11.8).

by= Autori. La chiave `by={\langle authors \rangle}` permette di indicare uno o più autori, compositori, traduttori, ecc. A ogni indice degli autori associato all'ambiente `songs` corrente viene aggiunta una singola voce per ognuno degli autori inseriti. Il programma si aspetta che gli autori siano separati da virgola, punto e virgola o la parola `and`. Ad esempio:

```
by={Fred Smith, John Doe, and Billy Bob}
```

Le parole separate da uno spazio esplicito (`_`) o da una tilde (`~`) sono trattate dal programma come fossero parole singole. Ad esempio `The_Vienna_Boys'_Choir` viene indicizzato come "Choir, The Vienna Boys'" ma `The_Vienna_Boys'_Choir` viene indicizzato come "Vienna Boys' Choir, The".

cr= Informazioni sul Copyright. La chiave `cr={\langle copyright \rangle}` specifica chi è il detentore del copyright, se esiste. Ad esempio:

```
cr={\copyright~2000 ABC Songs, Inc.}
```

Le informazioni sul copyright vengono stampate in carattere sottile alla fine della canzone.

$$\begin{aligned}
\langle refs \rangle &\longrightarrow \langle nothing \rangle \mid \langle ref \rangle ; \sqcup \langle ref \rangle ; \dots ; \sqcup \langle ref \rangle \\
\langle ref \rangle &\longrightarrow \langle many-chptr-book \rangle \sqcup \langle chapters \rangle \mid \langle one-chptr-book \rangle \sqcup \langle verses \rangle \\
\langle many-chptr-book \rangle &\longrightarrow \text{Genesis} \mid \text{Exodus} \mid \text{Leviticus} \mid \text{Numbers} \mid \dots \\
\langle one-chptr-book \rangle &\longrightarrow \text{Obadiah} \mid \text{Philemon} \mid 2 \text{ John} \mid 3 \text{ John} \mid \text{Jude} \\
\langle chapters \rangle &\longrightarrow \langle chref \rangle , \sqcup \langle chref \rangle , \dots , \sqcup \langle chref \rangle \\
\langle chref \rangle &\longrightarrow \langle chapter \rangle \mid \langle chapter \rangle - \langle chapter \rangle \mid \langle chapter \rangle : \langle verses \rangle \mid \\
&\quad \langle chapter \rangle : \langle verse \rangle - \langle chapter \rangle : \langle verse \rangle \\
\langle verses \rangle &\longrightarrow \langle vref \rangle , \langle vref \rangle , \dots , \langle vref \rangle \\
\langle vref \rangle &\longrightarrow \langle verse \rangle \mid \langle verse \rangle - \langle verse \rangle
\end{aligned}$$

Figura 2: Sintassi per i riferimenti biblici.

li= **Informazioni sulla licenza d'uso.** Le informazioni sulla licenza vengono indicate con `li={\langle license \rangle}`, dove $\langle license \rangle$ è una stringa di testo. Le informazioni sulla licenza vengono stampate in corpo molto piccolo sotto alle informazioni sul copyright, se ci sono. Scrivere `\setlicense{\langle license \rangle}` ovunque fra `\beginsong` e `\endsong` equivale a usare `li={\langle license \rangle}` nella riga `\beginsong`.

Se molte canzoni sono coperte dalla stessa licenza può essere comodo creare una macro per abbreviare le informazioni. Ad esempio:

```
\newcommand{\CCLI}{(CCLI \#1234567)}
```

si può poi scrivere `li=\CCLI` nella riga `\beginsong`.

sr= **Riferimenti biblici.** Il pacchetto `songs` permette di gestire le citazioni bibliche e gli indici delle citazioni. Per citare un riferimento biblico di usa la chiave `sr={\langle refs \rangle}`, dove $\langle refs \rangle$ è una lista di citazioni. Per ogni citazione viene creata una voce in ogni indice delle citazioni Il programma `songidx` si aspetta che $\langle refs \rangle$ sia una lista di riferimenti in cui il punto e virgola sia usato per separare riferimenti di libri diversi e la virgola per separare riferimenti di diversi capitoli e versetti dello stesso libro. Ad esempio una citazione è

```
sr={John 3:16,17, 4:1-5; Jude 3}
```

Nella figura 2 viene mostrata la sintassi di alcuni riferimenti biblici.

Nella sintassi mostrata $\langle chapter \rangle$ e $\langle verse \rangle$ sono numeri arabi indicanti un capitolo di un libro e il versetto di un capitolo. Se si fa riferimento a un libro contenente un solo capitolo si devono inserire soltanto i numeri di versetto dopo il nome del libro (invece di `1:\langle verses \rangle`).

7.2 Strofe e ritornelli

\beginverse **Scrivere una strofa o un ritornello.** Fra `\beginsong` e `\endsong` ci può stare un numero qualsiasi di strofe e ritornelli. una strofa inizia e finisce con:

\endverse

\beginchorus

\endchorus

```

\beginverse
:
\endverse

```

e un ritornello inizia e finisce con:

```

\beginchorus
:
\endchorus

```

Le strofe sono numerate (a meno che non si usi `\noversenumbers` per impedirne la numerazione) e i ritornelli hanno una linea verticale posizionata alla loro sinistra.

Per creare strofe non numerate si deve usare il comando `\beginverse*`. Questo si può fare nel caso in cui si vogliano scrivere strofe che non sono realmente strofe, ma che devono essere stampate come tali (come introduzioni, finali, intermezzi). Una strofa che inizia con `\beginverse*` deve comunque finire con `\endverse` (non `\endverse*`).

Nelle strofe e nei ritornelli si può inserire una riga di testo per ogni riga della canzone (\LaTeX solitamente ignora gli "a capo" e li considera come uno spazio semplice). Ogni riga del documento sorgente produce una riga separata nel documento finale (come la macro `\obeylines` di \LaTeX). Le righe troppo lunghe vengono spezzate con un rientro pari a `\parindent`.

`\repchoruses` **Ripetere i ritornelli.** Se si preparano delle slide per proiezione è comodo ripetere il ritornello dopo la prima strofa di ogni pagina, in modo che l'operatore al proiettore non debba tornare indietro per poter mostrare il ritornello. Per automatizzare questo processo si può scrivere `\repchoruses`. Se il primo ritornello della canzone fa parte di un gruppo di due o più ritornelli consecutivi, allora tutti i ritornelli vengono ripetuti. I ritornelli non vengono inseriti automaticamente dopo le strofe non numerate, poiché si suppone che queste siano intermezzi musicali o finali.

`\norepchoruses` La macro sopra descritta copre buona parte dei casi, ma per una gestione più precisa può servire un approccio manuale. Se una canzone ha una struttura irregolare si può usare `\norepchoruses` per inibire la ripetizione automatica dei ritornelli. Poi, laddove si voglia ripetere il ritornello,

```

\ifslides
\beginchorus
:
\endchorus
\fi

```

facendo copia-incolla del ritornello desiderato. Questo codice permette di inserire il ritornello solo quando si generano le slide e non nei canzonieri. Dopo la fine della canzone si scriverà:

```

\ifslides\repchoruses\fi

```

per riattivare la ripetizione automatica dei ritornelli, se lo si desidera.

7.3 Accordi

\[Fra \beginverse e \endverse o fra \beginchorus e \endchorus gli accordi possono essere inseriti usando il comando \[*chordname*]. Gli accordi appariranno solo nel canzoniere con accordi (chord) e non in quello con i soli testi (lyric). *chordname* è un testo qualsiasi. Per produrre i simboli di diesis e bemolle si usa # e &.

Ogni testo che segue la macro \[senza spaziatura è ritenuto essere la sillaba su cui l'accordo deve essere suonato, e quindi posizionato. Ad esempio:

\[E&]peace and \[Am]joy	<i>produce</i>	E^b	Am
		peace and	joy

Se la macro è seguita da uno spazio (spazio o *return*) il nome dell'accordo verrà stampato senza alcun testo sotto di sé, indicando che l'accordo deve essere suonato fra le parole che lo circondano. Ad esempio:

\[E&]peace and \[Am] joy	<i>produce</i>	E^b	Am
		peace and	joy

Se il testo della canzone che segue l'accordo contiene un altro accordo e se la larghezza dell'accordo è maggiore di quella del testo, la parola viene automaticamente sillabata e **songs** inserisce un trattino di sillabazione. Ad esempio:

\[F#sus4]e\[A]ternal	<i>produce</i>	$F^\#sus4$	A
		e -	ternal

Una sequenza di accordi che si trova sulla stessa sillaba può essere scritta senza spazi o addirittura all'interno dello stesso comando come fosse un solo accordo: ¹

\[A]\[B]\[Em]joy	<i>produce</i>	A B Em
		joy

\[A B Em]joy	<i>produce</i>	A B Em
		joy

L'unica differenza fra i due esempi sopra riportati è che nel primo gli accordi possono essere ripetuti separatamente (si veda §7.4) mentre nel secondo possono essere ripetuti solo raggruppati.

Si può indicare esplicitamente quanto testo debba stare sotto a un accordo utilizzando le parentesi graffe. Per escludere del testo che normalmente starebbe sotto a un accordo si può usare una coppia di parentesi che racchiuda l'accordo. Ad esempio:

{\[G A]e}ternal	<i>produce</i>	G A
		e - ternal

¹Per la stesura del canzoniere online della Pastorale Giovanile di Trento sconsigliamo quest'ultima soluzione e chiediamo che ogni \[] contenga un solo accordo. Questo perché altrimenti si impedisce il corretto funzionamento di macro che permettono di convertire il codice di **songs** in codice compatibile con altri pacchetti o altri programmi.

(senza le parentesi la sillaba “ternal” non sarebbe stata spinta lontana dall’accordo.) Questo può servire a indicare che il cambio di accordo deve avvenire prima di cantare la seconda sillaba.

Al contrario, delle parentesi che non contengono un accordo possono servire a mettere sotto l’accordo del testo che altrimenti sarebbe stato escluso. Ad esempio:

`\[Gmaj7sus4]{th’ eternal}` *produce* $\overset{Gmaj7sus4}{th’\ eternal}$

Senza le parentesi la parola “eternal” sarebbe stata posta più a destra in modo che l’accordo apparisse solo sopra alla sillaba “th”.

`\nolyrics` **Accordi senza testo.** Talvolta capita di voler scrivere una riga di accordi senza testo, come nelle introduzioni o negli intermezzi strumentali. Per far stare gli accordi sulla linea di base e non sollevati si può usare il comando `\nolyrics`. Ad esempio:

`{\nolyrics Intro: \[G] \[A] \[D]}` *produce* Intro: $G\ A\ D$

Notare bene le parentesi che indicano dove l’effetto della macro termina. Nelle parentesi possono essere inserite più righe di accordi. Le parti strumentali solitamente non appaiono nei canzonieri di soli testi (lyric), quindi è opportuno inserire il comando `\nolyrics` fra `\ifchorded` e `\fi` (si veda §11.4).

`\DeclareLyricChar` **Simboli sotto agli accordi.**

Se si stanno scrivendo canzoni in una lingua il cui alfabeto contiene dei simboli che \LaTeX tratta come punteggiatura è possibile usare il comando `\DeclareLyricChar` per dire a **sons** di considerare quel simbolo come un normale carattere.

`\DeclareLyricChar{\langle token \rangle}`

Dove $\langle token \rangle$ può essere una macro di \TeX , un carattere attivo, una lettera (qualcosa a cui \TeX assegna il codice di categoria 11), o un simbolo di punteggiatura (qualcosa a cui \TeX assegna il codice di categoria 12). Ad esempio, di default,

`\[Fmaj7]s\dag range` *produce* $\overset{Fmaj7}{s\ -\ \dagger range}$

perché `\dag` non è riconosciuto come un simbolo alfabetico; ma se si scrive,

`\DeclareLyricChar{\dag}`

si ottiene:

`\[Fmaj7]s\dag range` *produce* $\overset{Fmaj7}{s\dagger range}$

`\DeclareNonLyric` Peraltro si può anche scrivere

`\DeclareNonLyric{\langle token \rangle}`

per invertire l'effetto sopra presentato e forzare un token a essere considerato un carattere di punteggiatura. Questi token vengono spinti alla destra dei nomi degli accordi in modo che non cadano mai sotto a un accordo; viene inoltre inserito un trattino di sillabazione.

`\DeclareNoHyphen` Per definire dei token che vengano spinti a destra e non cadano sotto agli accordi, ma a cui non venga aggiunto il trattino di sillabazione, si può scrivere: `\DeclareNoHyphen{token}`.

`\MultiwordChords` **Estendere gli accordi sopra a parole adiacenti.** Il comando `\MultiwordChords`² forza più parole a essere inserite sotto lo stesso accordo. Normalmente un accordo lungo sopra una parola corta spinge la seconda parola a destra:

`\[Gmaj7sus4]my life` *produce* *Gmaj7sus4* my life

Ma se prima si scrive `\MultiwordChords`, il tutto risulterà più compatto:

`\[Gmaj7sus4]my life` *produce* *Gmaj7sus4*
my life

Occorre fare attenzione quando si usa `\MultiwordChords` per evitare di produrre dei testi che siano di dubbia interpretazione da parte dei musicisti. Ad esempio,

`\[F G Am]me free` *produce* *F G Am*
me free

Un risultato del genere non aiuta a capire che tutti tre gli accordi dovrebbero essere suonati sulla parola “me.” Perché `\MultiwordChords` produca un buon risultato occorre talvolta fare un saggio uso delle parentesi. Per questo motivo non è caricato come default.

`\shrp` **Alterazioni al di fuori degli accordi.** I simboli di diesis e bemolle possono
`\flt` essere prodotti con # e & quando ci si trova nella macro di un accordo. Ma se si vuole inserire questi simboli in altre parti di un documento bisogna usare i comandi `\shrp` e `\flt`. Per esempio per definire un comando che produca un accordo *C#* si scriverà:

`\newcommand{\Csharp}{C\shrp}`

7.4 Ripetere gli accordi

- ~ Molte canzoni son composte da più strofe che usano gli stessi accordi. Il pacchetto **songs** semplifica notevolmente l'inserimento degli accordi fornendo un modo per memorizzare e ripetere gli accordi. per ripetere un accordo di una strofa precedente basta digitare il simbolo (^) al posto del comando (`\[]`) che si sarebbe utilizzato. Ad esempio,

²le canzoni del Canzoniere della Pastorale Giovanile di Trento sono state scritte considerando sempre attivo il comando `\MultiwordChords`.

```

\beginverse
\[G]This is the \[C]first \[G]verse.
\endverse
\beginverse
The ^second verse ^ has the same ^chords.
\endverse

```

produce

$$\overset{G}{\text{This is the first verse.}}$$

$$\overset{G}{\text{The second verse}} \overset{C}{\text{has the same}} \overset{G}{\text{chords.}}$$

Si possono inserire normalmente degli accordi senza alterare la sequenza di accordi che viene ripetuta. Così una terza strofa potrebbe essere,

```

\beginverse
The ^third verse ^has a \[Cm]new ^chord.
\endverse

```

per produrre

$$\overset{G}{\text{The third verse has a new}} \overset{C}{\text{chord.}}$$

La ripetizione degli accordi può essere usata in combinazione con la trasposizione automatica per generare strofe cambiate di tono. Si veda §10 per un esempio.

`\memorize` Di default gli accordi vengono ripetuti memorizzando quelli della prima strofa della canzone, ma è possibile ripetere gli accordi di qualsiasi strofa o ritornello digitando il comando `\memorize` all'inizio della strofa o del ritornello di cui si vogliono ripetere gli accordi. Le strofe o i ritornelli in cui è presente il carattere `^` ripetono gli accordi dell'ultima strofa o dell'ultimo ritornello memorizzati.

Memorizzazione selettiva. È anche possibile inserire degli accordi non memorizzabili all'interno di una strofa memorizzata, in modo che non possano essere poi ripetuti. Per inibire la memorizzazione di un accordo basta anteporre al nome dell'accordo il simbolo di apice. Ad esempio,

```

\beginverse\memorize
The \[G]third \[C]chord will \[^Cm]not be re\[G]played.
\endverse
\beginverse
When ^replaying, the ^unmemorized chord is ^skipped.
\endverse

```

produce

$$\overset{G}{\text{The third chord will not}} \overset{C}{\text{be replayed.}}$$

$$\overset{G}{\text{When replaying, the}} \overset{C}{\text{unmemorized chord is}} \overset{G}{\text{skipped.}}$$

Questo è utile quando la prima strofa della canzone contiene qualcosa di unico che non deve essere ripetuto.

Memorizzare sequenza multiple di accordi. Di default il pacchetto **songs** memorizza una sola sequenza di accordi alla volta e \sim li ripete attingendo all'ultima sequenza memorizzata. Tuttavia è possibile memorizzare e ripetere più sequenze utilizzando i comandi presentati nei seguenti capoversi.

\newchords Le sequenze di accordi vengono memorizzate in appositi registri, per creare un nuovo registro si scrive

```
\newchords{<regname>}
```

dove *<regname>* è un nome alfabetico.

Dopo aver creato il registro lo si può usare indicandone il nome come opzione di **\memorize**:

```
\memorize[<regname>]
```

Memorizzare in un registro comporta la cancellazione di tutto quello che esso conteneva.

\replay Per ripetere gli accordi contenuti in un registro si scrive,

```
\replay[<regname>]
```

Utilizzando \sim si ripetono gli accordi contenuti nel registro *<regname>*.

I contenuti dei registri hanno valore globale, quindi si può memorizzare una sequenza da una canzone e ripeterla in un'altra. È possibile usare **\replay** più di una volta anche all'interno della stessa strofa o ritornello.

7.5 Interruzioni di riga e di colonna.

\brk **Interruzioni di riga.** Per spezzare una riga molto lunga in un punto particolare si deve inserire il comando **\brk** nel punto desiderato. Questo non ha effetto su righe abbastanza corte da stare nella pagina senza essere spezzate. Ad esempio,

```
\beginverse
This is a \brk short line.
But this is a particularly long line of lyrics \brk that will
need to be wrapped.
\endverse
```

produce

```
This is a short line.
But this is a particularly long line of lyrics
that will need to be wrapped.
```

Interruzioni di colonna all'interno delle canzoni. Per indicare un'interruzione di colonna all'interno di una strofa o di un ritornello troppo lunghi per stare in una colonna si usa `\brk` su una riga a sé stante. Se in una strofa lunga non ci sono `\brk`, questa viene spezzata in corrispondenza di una riga di testo che non vada a capo. (Una riga spezzata e andata a capo non è mai divisa da un'interruzione di colonna). Se in un ritornello molto lungo non ci sono `\brk`, questo sfiora la colonna e L^AT_EX dà un'errore di overfull vbox.

`\nextcol` **Interruzioni di colonna fra canzoni.** Per inserire un'interruzione di colonna fra due canzoni si può scrivere `\nextcol`, `\brk`, `\sclearpage`, o `\sclcardpage` fra le canzoni. Il comando `\nextcol` termina una colonna lasciando dello spazio bianco in fondo. Il comando `\brk` termina una colonna in un canzoniere di soli testi (lyric) modificando la spaziatura verticale del documento in modo da riempire tutta la colonna. (In tutti gli altri formati (non-lyric) `\brk` è uguale a `\nextcol`.) Il comando `\sclearpage` è come `\nextcol` ma fa ricominciare il testo su una nuova pagina. Il comando `\sclcardpage` è come `\sclearpage` ma fa ricominciare il testo su una pagina dispari nei documenti fronte-retro. Spesso le interruzioni di colonna devono trovarsi in posizioni differenti nei differenti formati di un canzoniere (lyric e chorded). Per fare questo è possibile usare i blocchi condizionali (si veda §11.4). Ad esempio,

```
\ifchorded\else\ifslides\else\brk\fi\fi
```

crea un'interruzione di colonna solo in un canzoniere senza accordi e non ha effetto sugli altri formati.

Quando si usa una lista parziale di canzoni estratta con `\includeonlysongs`, `\brk`, `\nextcol`, `\clearpage` e `\cleardpage`, posti fra le canzoni, devono essere seguiti da un asterisco per avere effetto. Per inserire un'interruzione di colonna in una lista parziale di canzoni si possono inserire le parole `nextcol`, `brk`, `clearpage`, o `cleardpage` nel punto corrispondente dell'argomento di `\includeonlysongs`.

7.6 Risposte e ripetizioni

`\echo` **Risposte - Seconde voci.** Per scrivere delle risposte nelle canzoni (o delle seconde voci) si può usare il comando `\echo{<lyrics and chords>}`. Queste parti vengono messe fra parentesi e in corsivo. Ad esempio,

```
Alle\{G\}luia! \echo{Alle\{A\}luia!}      produce  AlleGluia! (AlleAluia!)
```

`\rep` **Ripetizione di righe.** Per indicare che una riga o un ritornello devono essere ripetuti più volte si usa il comando `\rep{<n>}` alla fine della riga o del Ad esempio,

```
Alleluia! \rep{4}                          produce  Alleluia! (×4)
```

`\lrep` per indicare esattamente dove inizia e dove finisce la parte da ripetere si possono usare i comandi `\lrep` e `\rrep`, che creano delle barre verticali. Ad esempio,

`\lrep \[G]Alleluia!\rrep \rep{4}` *produce* $\left\| \begin{array}{c} G \\ \hline \end{array} \right\| (\times 4)$

7.7 Stanghette e misure

`\measurebar` Se ci si trova di fronte a canzoni poco conosciute può essere comodo avere delle
| indicazioni sulla lunghezza delle battute. Per questo **songs** permette di inserire
delle stanghette verticali a delimitare le diverse misure di una canzone. Per inserire
una stanghetta si può digitare `\measurebar` o una semplice barra verticale (“|”).
Ad esempio,

`Alle|\[G]luia` *produce* $\left| \begin{array}{c} G \\ \hline \end{array} \right|$ Alleluia

Affinché le stanghette siano stampate nel file occorre che l’opzione `showmeasuressia` abilitata. Le stanghette sono stampate di default solo nei canzonieri con accordi.

`\meter` La prima stanghetta di una canzone riporta anche il tempo della canzone. Di
default è 4/4. Per cambiare questo valore si scrive `\meter{\langle n \rangle}{\langle d \rangle}` in qualsiasi
punto dopo `\beginsong`, ma prima della prima stanghetta.

`\mbar` Si può anche cambiare il tempo all’interno di una canzone usando `\meter` o
digitando `\mbar{\langle n \rangle}{\langle d \rangle}` per produrre un indicazione col tempo $\langle n \rangle / \langle d \rangle$. Ad
esempio,

```
\meter{6}{8}
\beginverse
|Sing to the |heavens, ye \mbar{4}{4}saints of |old!
\endverse
```

produces

$\frac{6}{8}$ | Sing to the | heavens, ye $\frac{4}{4}$ | saints of | old!

7.8 TNote di testo

`\textnote` Oltre a strofe e ritornelli le canzoni possono contenere anche delle note testuali
`\musicnote` che diano istruzioni a cantanti e musicisti. Per creare queste note, visualizzate sia
sui canzonieri con accordi che in quelli senza si usa il comando:

`\textnote{\langle text \rangle}`

Per creare una nota stampata solo nei canzonieri con accordi si scrive:

`\musicnote{\langle text \rangle}`

Entrambi questi comandi creano finestre ombreggiate contenenti il $\langle text \rangle$. Ad
esempio,

`\textnote{Sing as a two-part round.}`

produce

Sing as a two-part round.

Le note testuali possono essere inserite ovunque in una canzone, all'interno delle strofe, dei ritornelli, o fra essi.

`\capo` **Capotasti.** C'è un tipo particolare di nota testuale che indica al chitarrista su quale tasto posizionare il capotasto della chitarra. Il comando `\capo{<n>}` viene usato per questo scopo. Normalmente ha lo stesso effetto del comando `\musicnote{capo <n>}`; tuttavia se l'opzione `transposecapos` è attiva, allora ha lo stesso effetto del comando `\transpose{<n>}`. Si veda §10 per informazioni più dettagliate sulla trasposizione automatica degli accordi.

7.9 Voci degli indici

I titoli di ogni canzone vengono automaticamente inseriti nell'indice del documento. Tuttavia è possibile aggiungere altre indicizzazioni per una canzone.

`index=` **Indice delle frasi significative.** Ad esempio spesso vengono inseriti negli indici, assieme ai titoli, dei pezzi significativi della canzone. È possibile aggiungere all'indice questo testo aggiungendo il comando: `index={<lyrics>}` nella riga `\beginsong`. Ad esempio,

```
\beginsong{Doxology}
[index={Praise God from Whom all blessings flow}]
```

darà come risultato che la canzone sarà inserita nell'indice sia come “*Doxology*” sia come “Praise God from Whom all blessings flow”. Il comando `index=` può essere usato anche più di una volta all'interno della stessa riga `\beginsong` per generare più voci nell'indice. Le voci generate con il comando `index={<lyrics>}` vengono stampate in carattere tondo anziché corsivo per distinguerle dai titoli veri e propri.

`ititle=` **Aggiungere titoli di canzone secondari.** Per aggiungere all'indice delle voci che vengano stampate in corsivo come i titoli di canzone si usa:

```
ititle={<title>}
```

sempre nella riga di `\beginsong`. Come la chiave `index=`, anche `ititle=` può essere usato più volte per produrre più voci nell'indice. Non è raro infatti che una canzone sia conosciuta con diversi titoli.

`\indexentry` Si possono aggiungere voci all'indice scrivendo `\indexentry[<indexes>]{<lyrics>}`
`\indextitleentry` (which creates an entry like `index=`) or `\indextitleentry[<indexes>]{<title>}`
(che crea una voce come `ititle=`). Questi due comandi possono essere usati ovunque fra `\beginsong` e `\endsong` e possono essere usati più di una volta per generare quante voci si vuole. Se specificato, `<indexes>` è una lista separata da virgole degli identificatori degli indici a cui si vuole aggiungere la voce. Altrimenti ogni nuova voce è aggiunta automaticamente a tutti gli indici attivati per l'ambiente `songs` in uso.

7.10 Gli accordi nelle legature

Questa sottosezione tratta di un argomento molto avanzato e probabilmente può essere saltata da chi usa **songs** per produrre documenti non professionali.³

Gli accordi vengono solitamente inseriti tramite il comando `\[]`; tuttavia questo comando non può essere utilizzato nel caso in cui l'accordo cada sopra a una legatura e si voglia che L^AT_EX generi la legatura. Una legatura è una combinazione di lettere o simboli che T_EX normalmente stampa come un singolo carattere al fine di produrre un documento più leggibile e meglio composto. In inglese le uniche legature sono: ff, fi, fl, ffi, and ffl. Altre lingue hanno altre legature come æ e œ. Si noti che in ognuno di questi casi le lettere vengono "schiacciate" assieme per formare un solo simbolo.

`\ch` Quando un accordo cade sopra una legatura, L^AT_EX non riesce a utilizzare il carattere giusto, nemmeno nella versione senza accordi (lyric) di un canzoniere. Per evitare questo errore tipografico si può usare il comando `\ch` per inserire un accordo:


`\ch{⟨chord⟩}{⟨pre⟩}{⟨post⟩}{⟨full⟩}`

dove `⟨chord⟩` è il nome dell'accordo, `⟨pre⟩` è il testo che appare prima della sillabazione se la legatura viene spezzata, `⟨post⟩` è il testo che compare dopo la sillabazione se la legatura viene e `⟨full⟩` è il testo completo della legatura se questo non viene spezzato della sillabazione. Ad esempio per scrivere correttamente `\[Gsus4]dif\[G]ficult`, in cui l'accordo *G* cade sopra alla legatura "ffi" si deve scrivere:

`di\ch{G}{f}{fi}{ffi}cult` *produce*  difficult

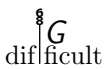
Così la legatura risulta correttamente stampata e l'accordo *G* è al posto giusto sopra alla seconda f. Per usare il comando `\ch` con un accordo ripetuto (si veda §7.4), basta usare `~` come `⟨chord⟩`.

`\mch` Il comando `\mch` è esattamente come `\ch` solo che inserisce anche una stanghetta nella legatura. Ad esempio,

`di\mch{G}{f}{fi}{ffi}cult` *produce*  difficult

posiziona una stanghetta e l'accordo *G* dopo la prima "f" nella parola "difficult", e genera una legatura completa "ffi" nelle versioni senza stanghette del canzoniere.

Nell'insolito caso in cui si debba inserire un cambio di tempo proprio su una legatura, si può scrivere:

`\meter{6}{8}di\mch{G}{f}{fi}{ffi}cult` *produce*  difficult

Il comando `\meter` imposta il nuovo tempo, che viene stampato sopra alla seguente stanghetta—in questo caso la stanghetta prodotta dal comando `\mch`.

³Tuttavia ne consigliamo la lettura perché mostra quante sottigliezze ci siano dietro a una corretta composizione tipografica!

Gli accordi e le stanghette creati con `^` o `|` possono essere tranquillamente usati nelle legature. Quindi `\difficult` non ha bisogno di particolari trattamenti; lascia intatta la legetura “ffi” quando le stanghette non vengono visualizzate.

8 Fra le canzoni

Non bisogna mai inserire del materiale al di fuori degli ambienti `song`, perché si causerebbe un cattivo funzinoamento del sistema di composizinoe delle pagine, il che di solito si manifesta in strane interruzioni di pagina e pagine vuote. Per inserire del materiale fra le canzoni si possono usare gli ambienti descritti di seguito.

8.1 L’ambiente `intersong`

intersong Per inserire fra le canzonei del materiale avente larghezza pari alla larghezza della colonna si usa l’ambiente `intersong`:

```
\begin{intersong}
:
\end{intersong}
```

Il materiale inserito in un ambiente `intersong` è soggetto alle stesse regole di interruzione di colonna delle canzoni, (si veda §11.5), ma tutto il resto della formattazione sta all’utente. Di default L^AT_EX inserisce una spaziatura espandibile (*glue* in linguaggio T_EXnico) dopo un ambiente `intersong`. Per eliminarla bisogna scrivere alla fine del contenuto dell’ambiente, `\par\nointerlineskip`.

intersong* Per inserire materiale avetne larghezza pari alla larghezza della pagina di usa l’ambiente `intersong*`:

```
\begin{intersong*}
:
\end{intersong*}
```

Questo inizia una nuova pagina, se la pagina corrente contiene già materiale avente larghezza pari alla larghezza della colonna.

songgroup Di default il contenuto degli ambienti `intersong` viene omesso quando si generano canzonieri a aprtire da liste parziali di canzoni con `\includeonlysongs`. Si può forzarne l’inserimento se la canzone è inclusa in un ambiente `songgroup`:

```
\begin{songgroup}
:
\end{songgroup}
```

Ogni ambiente `songgroup` può contenere un numero qualsiasi di ambienti `intersong`, `intersong*` o citazioni bibliche (si veda §8.2), ma deve contenere una sola canzone. Usando `\includeonlysongs` tutto il gruppo viene incluso se la canzone vinene inclusa, altrimenti tutto il gruppo viene omesso.

8.2 Citazioni bibliche

`\beginscripture` **Starting a Scripture Quotation.** Le citazioni bibliche sono un particolare tipo di materiale che sta al di fuori delle canzoni. Una citazione biblica inizia e termina con:

```
\beginscripture{<ref>}  
:  
:  
\endscripture
```

dove `<ref>` è un riferimento biblico che viene stampato alla fine della citazione. L'argomento `<ref>` deve avere la stessa sintassi dell'argomento `<ref>` dato al comando `\beginsong` (si veda §7).

Il testo della citazione biblica fra `\beginscripture` e `\endscripture` è stampato con un normale paragrafo. Ad esempio:

```
\beginscripture{James 5:13}  
Is any one of you in trouble? He should pray. Is anyone happy?  
Let him sing songs of praise.  
\endscripture
```

produce

*Is any one of you in trouble? He should
pray. Is anyone happy? Let him sing
songs of praise.* James 5:13

`\Acolon` **Tuplets.** Per stampare componimenti biblici in versi, come i salmi, si possono
`\Bcolon` iniziaer le righe con `\Acolon` o `\Bcolon`. Le righe A-colons sono stampate allineate
al margine sinistro, le B-colons sono rientrate. Ogni riga troppo lunga va a capo
con un rientro doppio. Ad esempio,

```
\beginscripture{Psalm 1:1}  
\Acolon Blessed is the man  
\Bcolon who does not walk in the counsel of the wicked  
\Acolon or stand in the way of sinners  
\Bcolon or sit in the seat of mockers.  
\endscripture
```

produce

*Blessed is the man
who does not walk in the counsel
of the wicked
or stand in the way of sinners
or sit in the seat of mockers.*
Psalm 1:1

`\strophe` **Versi.** Spesso i componimenti biblici sono raggruppati in strofe, separate solitamente da una sottile spaziatura verticale. Questa spaziatura può essere creata con il comando `\strophe`. Ad esempio,

```
\beginscripture{Psalm 88:2-3}
\Acolon May my prayer come before you;
\Bcolon turn your ear to my cry.
\strophe
\Acolon For my soul is full of trouble
\Bcolon and my life draws near the grave.
\endscripture
```

produce

*May my prayer come before you;
turn your ear to my cry.*

*For my soul is full of trouble
and my life draws near the grave.*

Psalm 88:2–3

`\scripindent` **Blocchi rientranti.** Alcuni passaggi biblici, come quelli che contengono sia prosa che poesia, contengono blocchi di testo rientrati. Con `\scripindent` si può aumentare il rientro all'interno di una citazione biblica, con `\scripoutdent` lo si può diminuire. Ad esempio,

```
\beginscripture{Hebrews 10:17-18}
Then he adds:
\scripindent
\Acolon ‘‘Their sins and lawless acts
\Bcolon I will remember no more.’’
\scripoutdent
And where these have been forgiven, there is no longer any
sacrifice for sin.
\endscripture
```

produce

Then he adds:

*‘‘Their sins and lawless acts
I will remember no more.’’*

*And where these have been forgiven,
there is no longer any sacrifice for sin.*

Hebrews 10:17–18

9 Tablature per chitarra

`\gtab` Le tablature per chitarra possono essere generate con la sintassi

`\gtab{⟨chord⟩}{⟨fret⟩:⟨strings⟩:⟨fingering⟩}`

dove le parti `⟨fret⟩` e `⟨fingering⟩` sono opzionali (e di conseguenza anche i due punti che seguono gli argomenti opzionali possono essere omessi).

`⟨chord⟩` è il nome dell'accordo che verrà posizionato sopra il diagramma.

`⟨fret⟩` è un numero opzionale (fra 2 e 9) che verrà posizionato alla sinistra del diagramma.

`⟨strings⟩` deve essere una sequenza di simboli, uno per ogni corda della chitarra, dalla più bassa alla più alta. Ogni simbolo può essere uno fra questi: **X** se la corda non deve essere suonata, 0 (zero o la lettera O) se la corda va suonata libera, o un numero fra 1 e 9 se la corda va suonata premendo il corrispondente tasto.

`⟨fingering⟩` è una sequenza opzionale di numeri, uno per ogni corda, dalla più bassa alla più alta. I numeri possono essere: 0 se non si vogliono mostrare informazioni sulla diteggiatura per quella corda (ad esempio se la corda non deve essere suonata o se va suonata aperta), o un numero fra 1 e 4 per indicare quale dito deve essere usato per suonare la corda.

Di seguito alcuni esempi:

<code>\gtab{A}{X02220:001230}</code>	<i>produce</i>	
<code>\gtab{C#sus4}{4:XX3341}</code>	<i>produce</i>	
<code>\gtab{B&}{X13331}</code>	<i>produce</i>	

`\minfrets` Di default le tablature riportano almeno 4 tasti (di più se l'argomento `⟨strings⟩` contiene un numero maggiore di 4). Per cambiare il numero minimo di tasti si può cambiare il valore di `\minfrets`. Ad esempio, scrivendo

`\minfrets=1`

le tablature avranno un numero di tasti pari al massimo valore dell'argomento `⟨strings⟩`.

10 Trasposizione automatica

`\transpose` Gli accordi di una canzone possono essere trasposti di $\langle n \rangle$ semitoni aggiungendo la riga

`\transpose{ $\langle n \rangle$ }`

in qualsiasi punto fra `\beginsong` e il primo accordo che deve essere trasposto. Ad esempio, se il primo accordo di una canzone è `\[D]` e prima di esso compare la riga `\transpose{2}`, nel documento finale l'accordo risultante sarà E . Se si specifica un numero negativo per $\langle n \rangle$, l'accordo verrà trasposto verso il basso.

Il comando `\transpose` ha effetto su tutti gli accordi fino alla riga `\endsong`. se nella stessa canzone vengono scritti due comandi `\transpose` il loro effetto si somma.

Quando l'opzione `transposecapos` è attiva, il comando `\capo` agisce come `\transpose`. Si veda §7.8 per ulteriori informazioni.

`\preferflats` **Enarmonici.** Quando si usa il comando `\transpose` per trasporre automaticamente gli accordi di una canzone, il pacchetto `songs` sceglie fra gli accordi enarmonici basandosi sul primo accordo della canzone. Ad esempio se scriviamo `\transpose{1}` e il primo accordo della canzone è un E , tutti i A diventeranno B^b e non A^\sharp , dal momento che la scala di F -major (il E trasposto di mezzo tono) ha una segnatura in bemolle. nella maggior parte dei casi questo algoritmo produce il risultato corretto, ma se ci dovessero essere dei problemi si può usare i comandi `\preferflats` o `\prefersharps` dopo `\transpose` per forzare la trasposizione usando accordi in bemolle o in diesis.

Strofe trasposte. La trasposizione automatica può essere usata assieme alla ripetizione degli accordi (si veda §7.3) per produrre strofe modificate di tonalità. Ad esempio,

```
\beginverse\memorize
\[F\#]This is a \[B/F\#]memorized \[F\#]verse. \[E\&7]
\endverse
\transpose{2}
\beginverse
^This verse is ^modulated up two ^half-steps.
\endverse
```

produce

$$\begin{array}{ccccccc} F^\sharp & & B/F^\sharp & & F^\sharp & & E^b7 \\ \text{This is a} & & \text{memorized} & & \text{verse.} & & \end{array}$$

$$\begin{array}{ccccccc} A^b & & D^b/A^b & & A^b & & \\ \text{This verse is} & & \text{modulated up two} & & \text{half-steps.} & & \end{array}$$

`\trchordformat` **Doppi accordi.** Di default quando gli accordi vengono trasposti automaticamente usando `\transpose`, vengono stampati solo gli accordi trasposti. Tuttavia in qualche caso può risultare comodo stampare sia gli accordi originali che quelli trasposti, così che musicisti che suonino strumenti trasposti e non trasposti possano usare lo stesso canzoniere. Questo può essere ottenuto ridefinendo il comando `\trchordformat`, che riceve due argomenti—il nome dell’accordo originale e il nome dell’accordo trasposto. Ad esempio, per stampare gli accordi originali sopra agli accordi trasposti si definisce

```
\renewcommand{\trchordformat}[2]{\vbox{\hbox{#1}\hbox{#2}}}
```

`\solfedge` **Cambiare i nomi delle note.** In molti Paesi si è soliti utilizzare (*LA, SI, DO, RE, MI, FA, SOL*) come nomi per gli accordi, al posto dei nomi alfabetici (*A, B, C, D, E, F, G*). Di default il motore di trasposizione lavora solo con i nomi alfabetici, ma gli si può dire di utilizzare la nomenclatura italiana scrivendo `\solfedge`. Per tornare alla nomenclatura alfabetica si scrive `\alphascale`.

`\notenames` Possono essere definiti altri nomi per le note. Lo si può fare scrivendo

```
\notenames{<nameA>}{<nameB>}\dots{<nameG>}
```

dove le voci da `<nameA>` a `<nameG>` devono essere una sequenza di lettere *maiuscole*. Ad esempio alcuni usano *TI* al posto di *SI*. Per trasporre automaticamente gli accordi con una notazione di questo tipo basta scrivere:

```
\notenames{LA}{TI}{DO}{RE}{MI}{FA}{SOL}
```

`\notenamesin` Il pacchetto **songs** può anche convertire automaticamente un set di nomi un un
`\notesnamesout` altro. Ad esempio, supponiamo di avere un canzoniere in cui gli accordi siano stati scritti con la notazione alfabetica e vogliamo stamparlo con la notazione italiana. Si può fare semplicemente usando il comando `\notesnamesin` per dire a **songs** quali sono i nomi delle note nell’input, e poi usando `\notesnamesout` per dire a **songs** quali sono i nomi degli accordi che si vogliono stampare. Il codice risulta così:

```
\notesnamesin{A}{B}{C}{D}{E}{F}{G}
```

```
\notesnamesout{LA}{SI}{DO}{RE}{MI}{FA}{SOL}
```

La sintassi di `\notesnamesin` e `\notesnamesout` è identica a quella di `\notenames` (si veda sopra), a parte il fatto che gli argomenti di `\notesnamesout` possono essere un qualsiasi codice \LaTeX che sia ammesso in modo orizzontale⁴, non solo lettere maiuscole.

Per bloccare la conversione dei nomi degli accordi si usa `\alphascale`, `\solfedge`, o `\notenames` per resettare tutti i nomi in modo che l’input e l’output siano uguali.

`\transposehere` **Trasporre gli accordi all’interno di comandi.** Il motore di trasposizione automatica non riconosce i nomi degli accordi che si trovano all’interno di comandi di \LaTeX .

⁴ \LaTeX lavora in diversi modi: orizzontale, verticale, matematico. Non spieghiamo in questa sede cosa significhi. Per semplicità diciamo che possiamo inserire come nomi degli accordi qualsiasi stringa alfanumerica.

```

\newcommand{\mychord}{F\shrp sus4/C\shrp}
\transpose{4}
\[\mychord]

```

il comando `\transpose` non riuscirà a trasporlo; l'accordo risultante sarà ancora *F#sus4/C#*. Per risolvere il problema si può usare `\transposehere` all'interno dei nuovi comandi, in modo da attivare esplicitamente il motore di trasposizione. L'esempio precedente può essere ridefinito correttamente:

```

\newcommand{\mychord}{\transposehere{F\shrp sus4/C\shrp}}

```

`\notrans` La trasposizione può essere localmente soppressa con il comando `\notrans`. Ad esempio, scrivendo

```

\transposehere{G = \notrans{G}}

```

si stampa un *G* trasposto seguito da un *G* non trasposto. Questo comando non blocca la conversione dei nomi degli accordi (si veda `\notenames`). Per bloccare anche la conversione dei nomi basta usare le parentesi (ad esempio, `{G}` invece di `\notrans{G}`).

`\gtabtrans` **Trasporre le tablature per chitarra.** Il pacchetto `songs` non traspone automaticamente le tablature per chitarra (si veda §9). Inoltre, quando avviene la trasposizione automatica, soltanto l'accordo inserito con `\gtab` viene mostrato (e trasposto); i diagrammi vengono tralasciati. Per cambiare questo comportamento di default bisogna ridefinire il comando `\gtabtrans` macro, i cui due argomenti sono i due argomenti di `\gtab`. Ad esempio, per mostrare le tablature non trasposte anche quando un documento è stato trasposto si scrive

```

\renewcommand{\gtabtrans}[2]{\gtab{\notrans{#1}}{#2}}

```

Per trasporre il nome dell'accordo ma non il diagramma sotto di esso si sostituisce `\notrans{#1}` con il solo `#1` nel codice sopra. Per ripristinare il comportamento di default si scrive

```

\renewcommand{\gtabtrans}[2]{\transposehere{#1}}

```

11 Personalizzare un canzoniere

11.1 Numerazione delle canzoni e delle strofe.

`songnum` **Numerazione delle canzoni.** Il contatore `songnum` definisce il numero della canzone che lo segue. È impostato a 1 all'inizio di ogni ambiente `songs` e viene incrementato di 1 dopo ogni `\endsong`. Può essere ridefinito ovunque tranne che all'interno di una canzone. Ad esempio,

```

\setcounter{songnum}{3}

```

imposta il numero della canzone successiva a 3.

`\thesongnum` Si può cambiare lo stile della numerazione per una sezione del canzoniere ridefinendo `\thesongnum`. Ad esempio, per fare in modo che la numerazione risulti A1, A2, etc., si può scrivere:

	<code>\renewcommand{\thesongnum}{A\arabic{songnum}}</code>
	L'espansione ⁵ di <code>\thesongnum</code> deve produrre soltanto testo senza comandi di formattazione o token non espandibile, dal momento che il testo deve essere esportato per la generazione degli indici.
<code>\printsongnum</code>	Per cambiare la formattazione dei numeri delle canzoni che compaiono all'inizio di ogni canzone si ridefinisce il comando <code>\printsongnum</code> , che si aspetta il testo fornito dal comando <code>\thesongnum</code> come unico argomento. Ad esempio per stampare i numeri in corsivo in cima a ogni canzone si scrive
	<code>\renewcommand{\printsongnum}[1]{\it\LARGE#1}</code>
<code>\songnumwidth</code>	La dimensione <code>\songnumwidth</code> indica la larghezza della cella colorata in grigio in cui vengono inseriti i numeri delle canzoni. Ad esempio per fare in modo che ogni cella sia larga 2 centimetri si scrive
	<code>\setlength{\songnumwidth}{2cm}</code>
<code>\nosongnumbers</code>	Se <code>\songnumwidth</code> è impostata a zero i numeri delle canzoni non vengono stampati. Per disattivare completamente la numerazione delle canzoni si usa il comando <code>\nosongnumbers</code> . Questo impedisce la visualizzazione dei numeri delle canzoni all'inizio delle canzoni stesse (ma i numeri vengono comunque mostrati ovunque siano richiamati, come ad esempio negli indici). Lo stesso effetto si ottiene impostando a zero la dimensione <code>\songnumwidth</code> .
<code>versenum</code>	Numerazione delle strofe. Il contatore <code>versenum</code> definisce il numero della strofa successiva. È impostato a 1 dopo ogni <code>\beginsong</code> e viene incrementato di 1 dopo ogni <code>\endverse</code> (tranne che se la strofa inizia con <code>\beginverse*</code>). Il contatore <code>versenum</code> può essere ridefinito ovunque all'interno di una canzone. Ad esempio,
	<code>\setcounter{versenum}{3}</code>
<code>\theversenum</code>	imposta il numero della strofa successiva a 3. Si può cambiare lo stile dei numeri di strofa ridefinendo <code>\theversenum</code> . Ad esempio per stampare i numeri delle strofe come numeri romani maiuscoli si scrive
	<code>\renewcommand{\theversenum}{\Roman{versenum}}</code>
<code>\printversenum</code>	Per cambiare la formattazione dei numeri delle strofe si ridefinisce il comando <code>\printversenum</code> , che si aspetta il testo fornito dal comando <code>\theversenum</code> come unico argomento. Per stampare i numeri in corsivo si scrive
	<code>\renewcommand{\printversenum}[1]{\it\LARGE#1.\ }</code>
<code>\versenumwidth</code>	La dimensione <code>\versenumwidth</code> definisce lo spazio orizzontale riservato per i numeri delle strofe alla sinistra di ogni strofa. Il testo delle strofe viene spostato verso destra di una lunghezza pari a <code>\versenumwidth</code> . Per dare ai numeri delle strofe mezzo centimetro di spazio si scrive

⁵Il concetto di *espansione* in T_EX non è semplice da spiegare. Possiamo limitarci a dire che non possiamo inserire in `\thesongnum` nulla se non testo.

`\setlength{\versenumwidth}{0.5cm}`

I numeri delle strofe che sono più larghi di `\versenumwidth` spostano la prima riga della strofa di una quantità sufficiente a garantirsi il giusto spazio, ma le righe successive hanno un rientro pari a `\versenumwidth`.

`\noversenumbers` Per inibire completamente la numerazione delle strofe si usa il comando `\noversenumbers`. Che equivale a dire

`\renewcommand{\printversenum}[1]{}`
`\setlength{\versenumwidth}{0pt}`

`\placeversenum` Il posizionamento orizzontale dei numeri delle strofe nella prima riga di ogni strofa è controllato dal comando `\placeversenum`. Di default ogni numero è allineato a sinistra. Gli autori interessati a modificare il posizionamento dei numeri delle strofe possono consultare §16.2 nella sezione implementazione per informazioni su questo comando.

11.2 Aspetto delle canzoni

`\lyricfont` **Scelta del font.** Le canzoni sono stampate utilizzando il font di default del documento (`\normalfont`) e con il corpo di default del documento (`\normalsize`). Questi parametri possono essere modificati ridefinendo `\lyricfont`. Ad esempio per scrivere le canzoni con un carattere senza grazie e in corpo `small` si scrive

`\renewcommand{\lyricfont}{\sffamily\small}`

`\stitlefont` I titoli delle canzoni sono stampati con un carattere senza grazie e inclinato (non inclinato se si producono delle slide). Queste impostazioni possono essere cambiate modificando `\stitlefont`. Ad esempio per stampare i titoli in carattere tondo (*roman*) si scrive

`\renewcommand{\stitlefont}{\rmfont\Large}`

`\versefont` Si può cambiare la formattazione di strofe, ritornelli e delle note generate con `\textnote` e `\musicnote` ridefinendo rispettivamente `\versefont`, `\chorusfont`, e `\notefont`. Ad esempio per scrivere i ritornelli in corsivo si definisce:

`\renewcommand{\chorusfont}{\it}`

`\notebgcolor` Il colore delle celle colorate che contengono le note testuali e i numeri delle canzoni può essere modificato ridefinendo `\notebgcolor` e `\snumbgcolor`. Ad esempio:

`\renewcommand{\notebgcolor}{red}`

`\printchord` Di default gli accordi vengono stampati con un carattere senza grazie inclinato. Per modificare la loro formattazione occorre ridefinire il comando `\printchord`, il cui argomento è il nome dell'accordo. Per stampare gli accordi in neretto con un carattere con grazie si scrive

`\renewcommand{\printchord}[1]{\rmfamily\bf#1}`

`\sharpsymbol` **Simboli delle alterazioni.** I simboli di diesis e bemolli vengono stampati con i comandi `\#` (`#`) e `\flat` (`b`) di L^AT_EX. Per modificarli si deve ridefinire `\sharpsymbol` e `\flatsymbol`. Ad esempio, per usare `\sharp` (`#`) anziché `#`, bisogna ridefinire `\sharpsymbol` come segue.

```
\renewcommand{\sharpsymbol}{\ensuremath{\^{\sharp}}}
```

`\everyverse` **Intestazioni delle strofe e dei ritornelli.** Il comando `\everyverse` viene eseguito all’inizio di ogni strofa e `\everychorus` all’inizio di ogni ritornello. Quindi per iniziare ogni ritornello con la parola “Chorus:” si può scrivere:

```
\renewcommand{\everychorus}{\textnote{Chorus:}}
```

`\versesep` **Opzioni di spaziatura.** La spaziatura verticale fra strofe e ritornelli è definita dal registro `\versesep`. Ad esempio per inserire uno spazio di 12 punti strofe e ritornelli, con un’elasticità di più/meno 2 punti si scrive

```
\versesep=12pt plus 2pt minus 2pt
```

`\afterpreludeskip` La spaziatura verticale fra il corpo di una canzone e la sua intestazione e il
`\beforepostludeskip` materiale in fondo è controllata da `\afterpreludeskip` e `\beforepostludeskip`. La spaziatura può essere *stiracchiata* per ottenere un miglior centraggio del corpo della canzone. Ad esempio per ottenere che il corpo delle canzoni sia centrato nella pagina con una sola canzone per pagina si scrive:

```
\songcolumns{1}
\spenalty=-10000
\afterpreludeskip=2pt plus 1fil
\beforepostludeskip=2pt plus 1fil
```

`\baselineadj` La distanza verticale fra le linee di base di due righe di testo consecutive (*interlinea*) è gestita dal pacchetto **songs** considerando diversi fattori fra cui il corpo del testo, il corpo degli accordi (se ci si trova in modalità **chord**) e se ci si trova in modalità **slides**. L’interlinea può essere modificata agendo sul registro `\baselineadj`. Ad esempio, per diminuire l’interlinea di un punto ma permettere una espansione di un punto quando L^AT_EX prova a bilanciare le colonne si può scrivere

```
\baselineadj=-1pt plus 1pt minus 0pt
```

`\clineparams` Per modificare la distanza verticale fra gli accordi e il testo sotto di loro si ridefinisce il comando `\clineparams` con una definizione che modifica i parametri di L^AT_EX `\baselineskip`, `\lineskiplimit` e `\lineskip`. Ad esempio per distanziare testo e accordi di 12 punti con almeno 1 punto di distanza fra la parte inferiore dell’accordo e quella superiore del testo si scrive:

```
\renewcommand{\clineparams}{
\baselineskip=12pt
\lineskiplimit=1pt
\lineskip=1pt
}
```

`\cbarwidth` Lo spessore del filetto verticale che viene stampato alla sinistra dei ritornelli è controllato dalla dimensione `\cbarwidth`. Per eliminare i filetti e lo spazio attorno a loro si può impostare `\cbarwidth` a `0pt`:

```
\setlength{\cbarwidth}{0pt}
```

`\sbarheight` Lo spessore del filetto orizzontale che viene stampato fra due canzoni è controllato dalla dimensione `\sbarheight`. Per eliminare i filetti e lo spazio attorno a loro si può impostare `\sbarheight` a `0pt`:

```
\setlength{\sbarheight}{0pt}
```

Intestazione e conclusione di una canzone. I contenuti dell'intestazione di una canzone e del materiale che si trova in fondo a essa possono essere modificati ridefinendo `\extendprelude` e `\extendpostlude`.

`\extendprelude` Di default `\extendprelude` mostra gli autori della canzone e i riferimenti biblici usando i comandi `\showauthors` e `\showrefs`. La seguente definizione modifica l'intestazione in modo che compaiano anche le informazioni sul copyright.

```
\showauthors
\showrefs
\renewcommand{\extendprelude}{
  \showrefs\showauthors
  {\bfseries\songcopyright\par}
}
```

`\extendpostlude`

`\extendpostlude` mostra di default le informazioni su copyright e licenza d'uso con i comandi `\songcopyright` e `\songlicense`. La seguente definizione permette di inserire le parole "Used with permission" alla fine di ogni canzone:

```
\renewcommand{\extendpostlude}{
  \songcopyright\ \songlicense\unskip
  \ Used with permission.
}
```

Ogni comando descritto nella sezione §12 può essere usato in `\extendprelude` e `\extendpostlude` per stampare informazioni sulla canzone come `\songauthors`, `\songrefs`, `\songcopyright` e `\songlicense`. I comandi `\showauthors` e `\showrefs` mostrano gli autori e i riferimenti biblici come capoversi pre-formatati.

Si veda §11.8 per capire come definire nuove chiavi per `\beginsong` e usarle in `\extendprelude`.

`\makeprelude` Per un controllo completo delle intestazioni e del materiale che si trova in
`\makepostlude` fondo alle canzoni si devono ridefinire i comandi `\makeprelude` e `\makepostlude`. Quando compone una canzone il pacchetto **songs** richiama entrambi questi comandi una volta (dopo aver processato tutto il materiale fra `\beginsong` e `\endsong`), posizionando il risultato all'interno di vbox. Le vbox risultanti sono posizionate in cima e in fondo al testo della canzone. Di default `\makeprelude` mostra i titoli, gli autori e i riferimenti biblici alla destra di una cella colorata in grigio contenente il numero della canzone.; e `\makepostlude` mostra le informazioni sul copyright e sulla licenza con un corpo piccolo.

`\vvpenalty` **Interruzioni di pagina e di colonna.** Il posizionamento delle interruzioni di colonna e di pagina all'interno di canzoni che sono troppo lunghe per strare all'interno di una sola colonna/pagina è influenzato dai valori di diverse penalità. Fra le righe di ogni strofa e di ogni ritornello viene inserita una penalità pari a `\interlinepenalty`; le penalità `\vvpenalty`, `\ccpenalty`, `\vcpenalty` e `\cvpenalty` vengono inserite rispettivamente fra strofe consecutive, fra ritornelli consecutivi, fra una strofa e un ritornello e fra un ritornello e una strofa. La penalità `\brkpenalty` viene inserita ovunque venga usato `\brk` su una riga a sé stante. Più alto è il valore della penalità, meno facilmente T_EX inserirà un'interruzione di colonna o di pagina in quel punto. Se una penalità è impostata a -10000 o meno là verrà forzata un'interruzione. Di default `\interlinepenalty` è impostata a 1000 e le altre a 200, così le interruzioni tra strofa e ritornello sono preferite alle interruzioni fra ritornello e strofa, ma non sono forzate.

`\sepverses` Scrivendo `\sepverses` si impostano tutte le penalità a -10000 eccetto `\ccpenalty` che viene impostata a 100. Questo può essere comodo in modalità `slides` perché forza ogni strofa e ritornello a essere stampati su slide separate, tranne che per i ritornelli consecutivi, che restano uniti, quando possibile. (Questo perché si suppone che due ritornelli di seguito siano in realtà un pre-ritornello e un ritornello che devono essere cantati assieme.)

Questo comportamento può essere modificato intervenendo direttamente sui valori delle penalità. Ad esempio per forzare un'interruzione di colonna o di pagina fra ritornelli consecutivi si scrive:

```
\ccpenalty=-10000
```

`\versejustify` **Giustificazione del testo.** Per giustificare le strofe e i ritornelli a destra oppure al centro si impostano rispettivamente `\versejustify` o `\chorusjustify` a `\justifyleft` o `\justifycenter`. Ad esempio per centrare il testo dei ritornelli si scrive:

```
\renewcommand{\chorusjustify}{\justifycenter}
```

`\notejustify` La giustificazione delle note testuali che sono troppo lunghe per stare in una riga di testo è controllata dal comando `\notejustify`. Di default imposta un ambiente che giustifica completamente le note (cioè tutte le righe di testo tranne l'ultima si estendono dal margine sinistro al margine destro).

Chi fosse interessato a modificare queste impostazioni legga §16.2 nella sezione "implementazione".

`\placernote` Un'annotazione testuale più corta di una riga di testo è posizionata allineata a sinistra o viene centrata se si lavora in modalità `slide`. Il posizionamento delle note è controllato da `\placernote`. Chi fosse interessato a modificare queste impostazioni legga §16.2 nella sezione "implementazione".

11.3 Formattazione delle citazioni bibliche

`\scripturefont` Le citazioni bibliche vengono stampate con il font Zaph Chancery utilizzando una dimensione pari a quella usata nel resto del documento (`\normalsize`). Queste

Type	Processed only if...
<code>chorded</code>	the <code>chorded</code> option is active
<code>lyric</code>	the <code>chorded</code> option is not active
<code>slides</code>	the <code>slides</code> option is active
<code>partiallist</code>	the <code>\includeonlysongs</code> macro is being used to extract a partial list of songs
<code>songindexes</code>	the <code>noindexes</code> option is not active
<code>measures</code>	the <code>nomeasures</code> option is not active
<code>pdfindex</code>	the <code>nopdfindex</code> option is not active
<code>rawtext</code>	the <code>rawtext</code> option is active
<code>transcapos</code>	the <code>transposecapos</code> option is active
<code>nolyrics</code>	the <code>\nolyrics</code> macro is in effect
<code>vnumbered</code>	the current verse is numbered (i.e., it was started with <code>\beginverse</code> instead of <code>\beginverse*</code>)

Tabella 1: Comandi condizionali

impostazioni possono essere modificate agendo su `\scripturefont`. Ad esempio per stampare le citazioni con un carattere senza grazie e in corsivo si scrive:

```
\renewcommand{\scripturefont}{\sffamily\it}

\printscrcite    I riferimenti biblici alla fine di una citazione vengono stampati con un carat-
```

tere senza grazie di dimensione `\normalsize`. L'aspetto dei riferimenti può essere modificato agendo su `\printscrcite`, che accetta i riferimenti come argomento. Ad esempio per stampare i riferimenti con un carattere con grazie e corsivo si scrive:

```
\renewcommand{\printscrcite}[1]{\rmfamily\it#1}
```

11.4 Blocchi condizionali

I comandi condizionali permettono di includere dei materiali in alcuni tipi di canzoniere e non in altri. Ad esempio un canzoniere con accordi può contenere strofe aggiuntive con accordi diversi.

```
\if...    Un blocco condizionale comincia con un comando chiamato \if<type>, dove <type> uno degli elementi della prima colonna della tabella 1. Un blocco condizionale termina con il comando \fi. Fra \if<type> e \fi può anche esserci un \else. Ad esempio nel codice
```

```
\ifchorded
  <A>
\else
  <B>
\fi
```


il materiale $\langle A \rangle$ viene incluso solo se l'opzione `chorded` è attiva e il materiale $\langle B \rangle$ viene incluso solo se l'opzione `chorded` non è attiva.

11.5 Layout di pagina

`\songcolumns` Il numero di colonne per pagina può essere impostato col comando `\songcolumns`. Ad esempio per ottenere 3 colonne per pagine si scrive

```
\songcolumns{3}
```

Il numero delle colonne può essere modificato soltanto al di fuori dell'ambiente `songs`.

Impostando a zero il numero delle colonne si disattiva completamente l'algoritmo di composizione delle pagine. Questo può essere utile qualora si voglia usare un altro pacchetto come `multicol` o la macro `\twocolumn` di \LaTeX . Ad esempio questo codice permette di creare un canzoniere utilizzando `\twocolumn`:

```
\songcolumns{0}
\flushbottom
\twocolumn[\LARGE\centering My Songs]
\begin{songs}{}
:
:
\end{songs}
```

Si faccia attenzione perché disabilitando l'algoritmo di composizione possono manifestarsi alcuni problemi:

- Il comando `\repchoruses` non funziona.
- I pacchetti esterni permettono interruzioni di pagina e di colonna anche all'interno di una canzone perché non hanno nessun meccanismo per spostare un'intera canzone in una pagina o una colonna vuota (si veda `\songpos` più sotto).
- Gli indici generati con `\showindex` vengono stampati alla larghezza dell'ambiente che li racchiude. Per questo occorre riportare \LaTeX a una colonna (tramite il comando `\onecolumn`) prima di eseguire `\showindex`.

`\columnsep` La distanza orizzontale fra due colonne consecutive è controllata dalla dimensione `\columnsep`. Per separare due colonne di 1 centimetro si scrive

```
\columnsep=1cm
```

`\colbotglue` Quando \LaTeX termina una colonna inserisce una "colla" pari a `\colbotglue`. Nei canzonieri senza accordi questo valore è impostato a `0pt` in modo che ogni colonna sia riempita fino in fondo. Negli altri formati il parametro è impostato in modo che la colonna possa terminare a qualsiasi altezza, lasciando dello spazio vuoto in fondo. Le impostazioni raccomandate in questo caso sono:

```
\renewcommand{\colbotglue}{0pt plus .5\textheight minus 0pt}
```

`\lastcolglue` L'ultima colonna di un ambiente `songs` è gestita con il comando `\lastcolglue`. di default questo ha un'elasticità infinita in modo che la colonna possa terminare alla sua altezza naturale. Impostandolo a `0pt`, si forza la colonna a raggiungere il fondo della pagina:

`\renewcommand{\lastcolglue}{0pt}`

`\songpos` Il pacchetto `songs` utilizza un algoritmo di posizionamento delle canzoni che sposta le canzoni in colonne o pagine vuote per impedire interruzioni di colonna o di pagina al loro interno. L'algoritmo ha quattro livelli di aggressività, da 0 a 3, modificabili scrivendo:

`\songpos{<level>}`

Il livello di default è 3, che impedisce, quando possibile, interruzioni di colonna e di pagina, oltre a impedire di dover girare pagina (cioè che una canzone sfiori su una pagina pari in un documento fronte retro o su una pagina nuova in un documento solo fronte). Il livello 2 impedisce le interruzioni di pagina e di dover girare pagina ma permette le interruzioni di colonna all'interno di una canzone. Il livello 1 impedisce solo di dover girare pagina. Il livello 0 disattiva completamente l'algoritmo di posizionamento delle canzoni. Questo comporta che le canzoni vengano posizionate dove \TeX crede sia meglio in base ai valori di penalità (si veda `\vvpenalty` e `\spenalty`).

`\spenalty` Il valore di `\spenalty` controlla la possibilità di avere interruzioni di colonna fra le canzoni. Solitamente deve essere impostato a un valore compreso fra 0 e `\vvpenalty` in modo che le interruzioni fra le canzoni siano preferibili a quelle tra le strofe. Di default è impostato a 100. Quando è -10000 o meno, si forzano le interruzioni fra le canzoni e ogni canzone inizia su una nuova colonna.

11.6 Indici

`\indexsongsas` **Aspetto degli indici.** Di default i titoli delle canzoni vengono inseriti nell'indice usando come riferimento il numero delle canzoni stesse. Per utilizzare i numeri di pagina si usa il comando `\indexsongsas`:

`\indexsongsas{<id>}{\thepage}`

dove `<id>` è l'identificatore usato in `\newindex`, `\newauthorindex` o `\newscripindex`. Il secondo argomento invece deve essere qualcosa che si espanda come testo semplice senza formattazione, poiché questo testo viene utilizzato dal programma di composizione degli indici che accetta solo questo tipo di dati. Per creare gli indici utilizzando i numeri delle canzoni si usa `\thesongnum` al posto di `\thepage` nell'esempio sopra.

`\sepindexestruue` Gli indici vengono stampati su pagine diverse e quando un indice è sufficientemente piccolo viene centrato nella pagina a una colonna sola. Per disabilitare questo comportamento si scrive `\sepindexesfalse`. Questo evita che gli indici usino spazio verticale non necessario o che comincino inutilmente nuove pagine. Per ripristinare il comportamento di default si usa `\sepindexestruue`.

`\idxrefsfont` Per controllare la formattazione della lista dei riferimenti biblici nella parte

destra dell'indice si ridefinisce `\idxrefsfont`. Ad esempio per stampare ogni lista in neretto si scrive:

```
\renewcommand{\idxrefsfont}{\bfseries}
```

`\idxtitlefont` Gli indici dei titoli possono contenere, oltre ai titoli, anche dei pezzi di canzone
`\idxlyricfont` che facilitino il riconoscimento delle canzoni. I font di queste voci sono controllati
rispettivamente da `\idxtitlefont` e `\idxlyricfont`. Ad esempio per visualizzare
i titoli con un carattere senza grazie e neretto e invece i pezzi di canzone in carattere
con grazie si scrive:

```
\renewcommand{\idxtitlefont}{\sffamily\bfseries}
\renewcommand{\idxlyricfont}{\rmfamily\mdseries}
```

`\idxheadfont` Per modificare il font usato per stampare le lettere maiuscole all'inizio di
ogni sezione alfabetica in un indice si ridefinisce `\idxheadfont`. Ad esempio per
stampare le lettere in corsivo anziché in neretto si scrive

```
\renewcommand{\idxheadfont}{\sffamily\it\LARGE}
```

`\idxbgcolor` Per modificare il colore di sfondo delle celle colorate che contengono le let-
tere maiuscole all'inizio di ogni gruppo alfabetico in un indice si ridefinisce
`\idxbgcolor`. Ad esempio:

```
\renewcommand{\idxbgcolor}{red}
```

`\idxheadwidth` La dimensione `\idxheadwidth` definisce la larghezza delle celle colorate che
iniziano ogni blocco alfabetico in un indice. Per impostare a una larghezza di un
centimetro si scrive:

```
\setlength{\idxheadwidth}{1cm}
```

`\idxauthfont` Il font usato per stampare i nomi degli autori nell'indice degli autori è gestito
dal comando `\idxauthfont`. Per stamparli in corsivo anziché in neretto si scrive:

```
\renewcommand{\idxauthfont}{\small\it}
```

`\idxscripfont` Il font usato per stampare le voci dell'indice dei riferimenti biblici è controllato
da `\idxscripfont`. Per stamparle in neretto anziché in corsivo si scrive:

```
\renewcommand{\idxscripfont}{\sffamily\small\bfseries}
```

`\idxbook` Per controllare il formato dei filetti che indicano l'inizio di ogni nuovo libro
della Bibbia all'interno dell'indice dei riferimenti si ridefinisce il comando `\idxbook`,
che ha come argomento i nomi dei libri della Bibbia. Ad esempio per stampare i
nomi all'interno di una cella si può scrivere

```
\renewcommand{\idxbook}[1]{\framebox{\small\bfseries#1}}
```

`\idxcont` Nell'indice dei riferimenti biblici, se succede che un'interruzione di colonna
divide delle voci riferite allo stesso libro, all'inizio della nuova colonna compare
il titolo “*<bookname>* (continued)”. Questo comportamentno può essere modificato
ridefinendo il comando `\idxcont`, che ha come unico argomento *<bookname>*. Ad
esempio per scrivere in italiano:

```
\renewcommand{\idxcont}[1]{\small\textbf{#1} (segue)}
```

`\titleprefixword` **Opzioni per l'ordine alfabetico.** In inglese, quando un titolo inizia con “The” o “A”, si è soliti spostare queste parole alla fine del titolo e ordinare le voci dell'indice considerando la parola successiva. Ad esempio, “The Song Title” viene indicizzato come “Song Title, The”. Per modificare questo comportamento si può usare il comando `\titleprefixword` nel preambolo del documento per identificare quali parole devono essere spostate alla fine quando appaiono all'inizio di un titolo. Ad esempio per fare in modo che la parola “La” sia spostata in fondo al titolo si può scrivere

`\titleprefixword{La}`

L'uso di `\titleprefixword` azzerava la lista di parole che vengono spostate in fondo al titolo, quindi se si vuole ancora spostare “The” e “A” alla fine del titolo si deve scrivere esplicitamente `\titleprefixword{The}` e `\titleprefixword{A}`. Questo comando può essere usato solo nel preambolo del documento e può essere usato quante volte si desidera.

`\authseppword` **Parole speciali nelle informazioni sulla canzone.** Quando il programma di composizione degli indici (`songidx`) legge le stringhe di testo riferite ai nomi degli autori considera la parola “and” come una congiunzione usata per separare i nomi degli autori. Per modificare questa impostazione o per aggiungere altre congiunzioni si usa il comando `\authseppword`. Esso può essere usato quante volte si desidera per specificare più congiunzioni. Ad esempio per usare “e” come congiunzione si può scrivere

`\authseppword{e}`

L'uso di `\authseppword` sovrascrive le impostazioni di default. Quindi se si vuole utilizzare “and” come congiunzione occorre scriverlo esplicitamente (`\authseppword{and}`). Il comando `\authseppword` può essere usato solo nel preambolo e può essere usato quante volte si vuole.

`\authbyword` La parola “by” viene riconosciuta come una parola chiave che indica che soltanto il materiale che si trova dopo di essa dovrà essere inserito nell'indice. Così “Music by J.S. Bach” viene indicizzato come “Bach, J.S.” anziché come “Bach, Music by J.S.” Per utilizzare altre parole al posto di “by”, si usa il comando `\authbyword` nel preambolo del documento. Ad esempio per utilizzare “di” si scrive

`\authbyword{di}`

`\authignoreword` Se una lista contiene la parola “unknown”, quella voce non viene inserita nell'indice. Questo fa sì che voci come “Composer unknown” non vengano messe nell'indice. Per dire al programma di ignorare anche altre parole si usa il comando `\authignoreword` nel preambolo del documento. Ad esempio per ignorare le voci che contengono la parola “sconosciuto”, si scrive,

`\authignoreword{sconosciuto}`

11.7 Intestazione e piè di pagina

In \LaTeX le intestazioni e i piè di pagina vengono costruiti utilizzando dei *mark* invisibili che vengono inseriti all'inizio di ogni unità logica del documento (sezioni, canzoni, strofe, ritornelli). Le intestazioni e i piè di pagina vengono quindi definiti in modo da fare riferimento al primo o all'ultimo mark invisibile che compare nelle pagine del documento. Questa sezione descrive quali mark vengono abilitati da **songs**. Per maggiori informazioni sui mark creati da \LaTeX e su come utilizzarli si può leggere qualsiasi manuale di \LaTeX .

`\songmark`
`\versemark`
`\chorusmark`

Per inserire nell'intestazione o nel piè di pagina delle informazioni riguardanti la canzone bisogna ridefinire `\songmark`, `\versemark` o `\chorusmark` per aggiungere i necessari mark qualora si abbia l'inizio di una nuova canzone, di un nuovo ritornello o di una nuova strofa. Questi comandi non hanno argomenti. Per avere informazioni riguardo alla canzone corrente, titolo compreso, si possono usare i comandi descritti nella sezione §12. Per stampare il numero della canzone corrente o il numero della strofa corrente si usano i comandi `\thesongnum` e `\theversenum` (si veda §11.1). Ad esempio per inserire il numero della canzone nell'intestazione generata col comando di \LaTeX `\pagestyle{myheadings}`, si può ridefinire `\songmark` come segue:

```
\renewcommand{\songmark}{\markboth{\thesongnum}{\thesongnum}}
```

11.8 Definire nuove chiavi per beginsong

`\newsongkey`

Il comando `\beginsong` ha diverse chiavi opzionali che permettono di inserire informazioni riguardo alla canzone, come `by=`, `sr=` e `cr=`. Ogni utente può definire nuove chiavi. Per fare questo si usa il comando `\newsongkey`, che ha la sintassi

```
\newsongkey{⟨keyname⟩}{⟨initcode⟩}[⟨default⟩]{⟨setcode⟩}
```

Dove `⟨keyname⟩` è il nome della nuova chiave, `⟨initcode⟩` è il codice \LaTeX che viene eseguito all'inizio di ogni `\beginsong` prima che gli argomenti di `\beginsong` vengano processati, `⟨default⟩` (se specificato) è il valore di default della chiave quando `⟨keyname⟩` viene inserito in `\beginsong` senza un valore esplicito, `⟨setcode⟩` è il codice che viene eseguito ogni volta che `⟨key⟩` viene letta da \LaTeX . All'interno di `⟨setcode⟩` la stringa `#1` si riferisce al valore della chiave che è stato esplicitato dell'utente o al valore `⟨default⟩` se non è stato inserito alcun valore.

Ad esempio per definire una nuova chiave nominata **arr** che registra i propri valori in un comando chiamato `\arranger` si può scrivere:

```
\newcommand{\arranger}{}  
\newsongkey{arr}{\def\arranger{}}  
{\def\arranger{Arranged by #1\par}}
```

Poi si può ridefinire `\extendprelude` in modo che il nome dell'arrangiatore della canzone venga stampato sotto le altre informazioni nell'intestazione della canzone:

```

\renewcommand{\extendprelude}{
  \showrefs\showauthors
  {\bfseries\arranger}
}

```

Nella riga `\beginsong` si può specificare il nome dell'arrangiatore come segue:

```

\beginsong{The Title}[arr={R. Ranger}]
:
\endsong

```

Questo codice produce:



Per informazioni più precise sulle chiavi e sul loro funzionamento si consiglia di leggere la documentazione del pacchetto `keyval` di David Carlisle.

11.9 Correzione della crenatura dei font

Sovrapposizione degli accordi. Per risparmiare spazio e e per fare in modo che le canzoni risultino leggibili il pacchetto `songs` mette gli accordi molto vicini al testo a cui sono sovrapposti. Sfortunatamente può succedere che qualche carattere con dei discendenti molto lunghi si sovrapponga al testo della canzone. Ad esempio,

`\[(Gsus4/D)]Overstrike` *produce* *(Gsus4/D)*
`Overstrike`

Si noti che le parentesi e la barra sono sovrapposti al testo della canzone.

`\chordlocals` La soluzione migliore è quella di usare per gli accordi un font che abbia dei discendenti corti. tuttavia se non si ha a disposizione un font del genere si può usare questo trucco: Nel preambolo del documento si può scrivere:

```

\renewcommand{\chordlocals}{\catcode'\active
                             \catcode'\active
                             \catcode'\active}
\newcommand{\smraise}[1]{\raise2pt\hbox{\small#1}}
\newcommand{\myslash}{\smraise/}
\newcommand{\myopenparen}{\smraise(}
\newcommand{\mycloseparen}{\smraise)}
\chordlocals
\global\let\myopenparen
\global\let\mycloseparen
\global\let\myslash

```

Questo codice fa sì che `/`, `(`, e `)` diventino dei caratteri attivi quando si trovano all'interno del nome degli accordi. (Si veda §16.2 per la documentazione sul comando `\chordlocals`.) Ogni carattere attivo è definito in modo da produrre una versione più piccola e rialzata del simbolo originale. Il risultato è il seguente:

$\backslash[(Gsus4/D)]Overstrike$ (fixed) *produce* $\overset{(Gsus4/D)}{Overstrike}$ (fixed)

Come si vede i simboli sono stati sollevati in modo che siano allineati alla linea di base, risolvendo il problema della sovrapposizione.

`\shiftdblquotes` **Scripture Font Quotation Marks.** Il pacchetto `songs` risolve un problema di crenatura del font Zaph Chancery font (usato per comporre le citazioni bibliche) ridefinendo ‘ ‘ e ’ ’ in modo che diventino caratteri attivi spostati a sinistra rispettivamente di 1,1 pt e 2 pt rispetto alla loro posizione normale. Se si utilizza per le citazioni bibliche un corpo del testo differente da quello standard si può usare il comando `\shiftdblquotes` per ridefinire `\scripturefont` per cambiare la correzione della crenatura. Ad esempio,

```

\renewcommand{\scripturefont}{
  \usefont{OT1}{pzc}{mb}{it}
  \shiftdblquotes{-1pt}{-2pt}{-3pt}{-4pt}
}

```

toglie 1 punto di spazio alla sinistra e 2 punti alla destra delle virgolette aperte e 3 punti alla sinistra e 4 alla destra delle virgolette chiuse, sempre all’interno delle citazioni bibliche.

12 Informazioni sulle canzoni

I comandi descritti in questa sezione possono essere utilizzati per ottenere informazioni riguardo alla canzone corrente. Possono essere usati quando si ridefiniscono `\extendprelude`, `\extendpostlude`, `\makeprelude`, `\makepostlude`, `\songmark`, `\versemark` o `\chorusmark`, o qualsiasi altro comando che permetta di stampare queste informazioni.

`\songauthors` Per stampare la lista degli autori di una canzone (se ci sono) si usa `\songauthors`. Questo stampa il valore contenuto nella chiave `by=` usata nella riga di `\beginsong`.

`\songrefs` Per stampare la lista dei riferimenti biblici di una canzone si usa `\songrefs`. Questo stampa i valori contenuti nella chiave `sr=` usata nella riga di `\beginsong`, ma con alcune modifiche. I trattini diventano trattini doppi, e gli spazi all’interno delle liste dei numeri dei versetti diventano spazi sottili. Inoltre vengono inserite delle penalità per impedire delle interruzioni di linea in certi punti e favorirle in altri.

`\songcopyright` Per stampare le informazioni relative al copyright di una canzone si usa `\songcopyright`. Questo comando stampa il contenuto della chiave `cr=` usata nella riga di `\beginsong`.

`\songlicense` Per ottenere le informazioni sulla licenza d’uso di una canzone si usa `\songlicense`. Questo comando stampa il valore contenuto nella chiave `li=` usata nella riga di `\beginsong` o qualunque testo dichiarato con `\setlicense`.

`\songtitle` Il comando `\songtitle` restituisce il titolo della canzone. Di default dà il

	primo titolo inserito con <code>\beginsong</code> . I comandi <code>\nexttitle</code> e <code>\foreachtitle</code> (si veda sotto) fanno sì che <code>\songtitle</code> restituisca il titolo alternativo, se c'è.
<code>\resettitles</code>	Per ottenere il titolo principale della canzone (cioè il primo specificato con <code>\beginsong</code>), si usa il comando <code>\resettitles</code> . Questo imposta il comando <code>\songtitle</code> al titolo principale.
<code>\nexttitle</code>	Per ottenere il titolo <i>successivo</i> si usa <code>\nexttitle</code> , che imposta <code>\songtitle</code> al titolo successivo nella lista dei titoli. (o imposta <code>\songtitle</code> a <code>\relax</code> se non ci sono altri titoli).
<code>\foreachtitle</code>	Il comando <code>\foreachtitle</code> accetta solo codice L ^A T _E X come argomento e lo esegue una volta per ogni titolo (rimanente). All'interno del codice si usa sempre <code>\songtitle</code> per ottenere il titolo corrente. Ad esempio il codice seguente genera una lista separata da virgole di tutti i titoli di una canzone.
	<pre> \resettitles \songtitle \nexttitle \foreachtitle{, \songtitle} </pre>
<code>\songlist</code>	Quando si usa <code>\includeonlysongs</code> per stampare una lista parziale di canzoni il comando <code>\songlist</code> espande la lista dei titoli delle canzoni che vengono estratte. Ridefinendo <code>\songlist</code> all'interno del preambolo si modifica la lista delle canzoni che vengono estratte. Ridefinendolo dopo il preambolo si possono avere risultati imprevedibili.

13 Creazione degli indici

In questa sezione vengono descritti comandi che **songs** utilizza durante la compilazione degli indici. Dal momento che la compilazione è automatica non si dovrebbe mai avere bisogno di usare direttamente questi comandi. Pertanto questa parte di documentazione viene fornita soltanto per completezza e a titolo informativo. Per le istruzioni su come generare automaticamente gli indici si veda §6. Per informazioni su come personalizzare l'aspetto degli indici si veda §11.6.

La generazione automatica degli indici è un processo diviso in tre parti:

1. Ogniqualvolta si compila un file di un canzoniere si crea un file denominato `<filename>.sxd` per ogni `<filename>` definito usando `\newindex`, `\newauthorindex` o `\newscripindex`. Questi file `.sxd` sono semplici file di testo che possono essere aperti con qualsiasi editor di testo. Essi cominciano con una riga in cui viene identificato il tipo di indice (titoli, autori o riferimenti biblici) e poi contengono terne di righe, una terna per ogni canzone che appare nell'indice. Nella prima riga della terna viene descritto il criterio in base a cui la canzone deve essere indicizzata (per titolo, autore o riferimento biblico). La seconda riga riporta il numero della canzone (dato da `\thesongnum`). La terza riga è un'etichetta identificativa usata nei collegamenti ipertestuali.

2. Si usa poi un programma esterno per trasformare i file `.sxd` in file `.sbx`. dal momento che il programma `makeindex` fornito con \LaTeX non è in grado di ordinare i riferimenti biblici, il pacchetto `songs` viene fornito con il programma `songidx`, che è in grado di farlo.
3. I file `.sbx` generati da `songidx` vengono letti nel comando `\showindex` durante la necessaria successiva compilazione con \LaTeX . I file `.sbx` contengono i comandi e gli ambienti descritti di seguito.

`idxblock` Per gli indici che vengono divisi in blocchi alfabetici si avrà che i file `\langle filename \rangle.sbx` generati per quegli indici conterranno una serie di ambienti `idxblock`, uno per ogni blocco alfabetico. Un ambiente `idxblock` inizia e finisce con

```
\begin{idxblock}{\langle letter \rangle}
:
\end{idxblock}
```

dove `\langle letter \rangle` è la lettera dell'alfabeto per quel blocco.

`\idxentry` Le voci degli indici sono generate con righe del tipo:
`\idxaltentry` `\idxentry{\langle leftside \rangle}{\langle rightside \rangle}`
`\indexaltentry{\langle leftside \rangle}{\langle rightside \rangle}`

ognuna delle quali crea una voce dell'indice con `\langle leftside \rangle` a sinistra, seguita da una serie di punti, seguiti da `\langle rightside \rangle` osulla destra. Il comando `\indexentry` viene utilizzato per le voci “normali” (come i titoli nell'indice dei titoli) e `\indexaltentry` viene usato per le voci “alternative” (come pezzi di canzone nell'indice dei titoli).

All'interno di `\langle rightside \rangle` gli oggetti vengono separati con `\\` anziché con le virgole. Se utilizzato all'interno di un file `.sbx` il comando `\\` genera una virgola seguita da una spaziatura che permette alle righe dell'indice di essere adeguatamente spezzate se non stanno in una sola riga di testo.

14 Altri pacchetti

Esistono altri pacchetti per \LaTeX che permettono di scrivere canzoni, tablature per chitarra e canzonieri con accordi. Probabilmente il migliore fra questi è `Songbook` di Christopher Rath (<http://rath.ca/Misc/Songbook/>). La maggior parte delle differenze fra questi due pacchetti è voluta; di seguito si riporta un riassunto delle decisioni prese e delle motivazioni che hanno portato alle scelte fatte.

Facilità di scrittura delle canzoni. I maggiori sforzi nella scrittura del pacchetto `songs` sono stati dedicati a rendere l'inserimento degli accordi il più facile possibile. Con la maggior parte degli altri pacchetti per \LaTeX per inserire un accordo si usa la sintassi `\chord{\langle chord \rangle}{\langle lyric \rangle}`. Il pacchetto `songs` usa invece la sintassi meno convenzionale `\[\langle chord \rangle]\langle lyric \rangle`. Le ragioni sono molteplici e sono descritte di seguito.

Innanzitutto la sintassi tradizionale richiede un numero maggiore di caratteri rispetto a quella di **songs**. Se si scrive un canzoniere molto lungo questo aspetto può far risparmiare molto tempo. E in questo modo il codice risulta anche più leggibile.

Poi, i comandi standard di L^AT_EX richiedono che l'utente stimi quale porzione della canzone debba stare sotto a un accordo (perché la parte *lyric* deve essere racchiusa all'interno di parentesi graffe) mentre il pacchetto **songs** non lo richiede. Non è facile stimare correttamente la lunghezza della parte *lyric* ed essa deve talvolta contenere spaziature, punteggiatura particolare o più parola al fine di ottenere il risultato corretto. Il pacchetto **songs** risolve questo problema scegliendo al posto dell'utente.

Terza cosa, il pacchetto **songs** gestisce automaticamente la sillabazione dei pezzi di canzone che giacciono sotto agli accordi. Se una sillaba è più corta dell'accordo che sta sopra di essa, **songs** provvede automaticamente a separarla dalla sillaba successiva e a inserire un trattino di sillabazione.

E ultima cosa, alcuni pacchetti utilizzano la lettera “b” in un *chord* per generare il simbolo di bemolle, mentre **songs** usa “&”. Utilizzare “b” è certamente più intuitivo ma impedisce di utilizzare la “b” all'interno del campo *chord*, ad esempio per stampare proprio la lettera “b” o per scrivere un comando come `\hbox` che contiene una “b”.

Struttura delle canzoni. Il pacchetto **songs** fornisce un numero di comandi relativamente ristretto per la gestione delle strutture logiche di una canzone (strofe, ritornelli, commenti e comandi condizionali). Questi comandi possono essere combinati fra loro per creare strutture più complicate come introduzioni, intermezzi, finali e cose del genere. Questo è stato fatto nella convinzione che avere a disposizione pochi comandi renda il pacchetto più flessibile e ne faciliti l'apprendimento.

Colonne multiple. Il pacchetto **songs** è stato progettato da zero per produrre canzonieri con più canzoni per pagina, disposte su più colonne. Per ottenere questo risultato esso contiene delle caratteristiche che non si trovano negli altri pacchetti, come ad esempio il bilanciamento automatico delle colonne, la possibilità di personalizzare le intestazioni delle canzoni e la possibilità di inserire delle citazioni bibliche per riempire gli spazi fra le canzoni.

Indici. un'altra caratteristica importante del pacchetto **songs** è la possibilità di creare diversi tipi di indice (dei titoli, degli autori e dei riferimenti biblici). L'indice dei riferimenti biblici può essere molto comodo per trovare le canzoni adatte a certe cerimonie. Il pacchetto **songs** permette di specificare i nomi dei libri della bibbia e di disporli nell'ordine desiderato.

Trasposizione automatica. Il pacchetto **songs** permette di trasporre automaticamente gli accordi di una canzone. Permette inoltre di stampare canzonieri con

gli accordi scritti in due tonalità diverse, (così che un chitarrista e un pianista possano leggere lo stesso canzoniere).

Il pacchetto **songs** è stato sviluppato indipendentemente dagli altri pacchetti **L^AT_EX** che si possono usare per scrivere canzoni. inizialmente è stato sviluppato un gruppo di comandi per **L^AT_EX** che in seguito è diventato il pacchetto **songs**. Tutto è nato per scrivere il canzoniere della Graduate Christian Fellowship (GCF) alla Cornell University e della Cornell International Christian Fellowship (CICF). Dopo diversi affinamenti, quando il pacchetto si è rivelato sufficientemente versatile, è stato pubblicato per il pubblico utilizzo.

Per avere informazioni più dettagliate sulle altre risorse per scrivere canzoni si raccomanda la lettura della documentazione del pacchetto **Songbook**. Essa contiene molte informazioni che possono interessare chi si cimenta nella creazione di canzonieri.

15 GNU General Public License

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

- (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However,

as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

16 Implementation

The following provides the verbatim implementation of the **songs** L^AT_EX package, along with commentary on how it works. In general, macro names that contain a @ symbol are not intended to be directly accessible by the outside world; they are for purely internal use. All other macros are intended to be used or redefined by document authors.

Most of the macros likely to be of real interest to song book authors can be found in §16.2. To find the implementation of any particular macro, the index at the end of this document should prove helpful.

The unwary T_EXer may wonder at the rather large size of the implementation. The volume and complexity of the code stems mainly from the following challenging features:

- Putting chords above lyrics fully automatically requires building an entire lyric-parser in L^AT_EX (see §16.10).
- Avoiding page-turns within songs without prohibiting column-breaks requires building a completely new page-breaking algorithm (see §16.4).
- The package must be able to generate an astonishing number of document variants from a common source: lyric-only books, chorded books, digital slides, transparency slides, selected song subsets, transposed songs, and combinations of the above. This is like putting six or more packages into one.
- Song book indexes are far more complex than those for a prose book. See §16.15 for some of the difficulties involved.

16.1 Initialization

The code in this section detects any T_EX versioning or configuration settings that are relevant to the rest of the song book code.

`\ifSB@etex` Numerous enhancements are possible when using an ε -T_EX compatible version of L^AT_EX. We start by checking to see if ε -T_EX primitives are available.

```
1 \newif\ifSB@etex
2 \ifx\TeXversion\undefined\else
3   \ifx\TeXversion\relax\else
4     \SB@etextrue
5     \IfFileExists{etex.sty}{\RequirePackage{etex}}{}
6   \fi
7 \fi
```

`\ifSB@pdf` Detect whether we're generating a pdf file, since this affects the treatment of hyperlinks and bookmark indexes.

```

8 \newif\ifSB@pdf\SB@pdffalse
9 \ifx\pdfoutput\undefined\else
10 \ifx\pdfoutput\relax\else
11 \ifnum\pdfoutput<\@ne\else
12 \SB@pdftrue
13 \fi
14 \fi
15 \fi

```

`\ifSB@preamble` Some macros have different effects depending on when they're used in the preamble or in the document body, so we need a conditional that remembers whether we're still in the preamble. It gets initialized to true and later changed to false once the body begins.

```

16 \newif\ifSB@preamble
17 \SB@preambletrue

```

`\ifSB@test` Reserve some control sequence names for scratch use.

```

\SB@temp 18 \newif\ifSB@test
\SB@tempii 19 \newcommand\SB@temp{}
\SB@tempiii 20 \newcommand\SB@tempii{}
\SB@tempiv 21 \newcommand\SB@tempiii{}
\SB@tempv 22 \newcommand\SB@tempiv{}
23 \newcommand\SB@tempv{}

```

`\SB@dimen` Reserve some temp registers for various purposes.

```

\SB@dimenii 24 \newdimen\SB@dimen
\SB@dimeniii 25 \newdimen\SB@dimenii
\SB@dimeniv 26 \newdimen\SB@dimeniii
\SB@box 27 \newdimen\SB@dimeniv
\SB@boxii 28 \newbox\SB@box
\SB@boxiii 29 \newbox\SB@boxii
\SB@toks 30 \newbox\SB@boxiii
\SB@cnt 31 \newtoks\SB@toks
\SB@cntii 32 \newcount\SB@cnt
\SB@cntiii 33 \newcount\SB@cntii
\SB@skip 34 \newskip\SB@skip

```

Load David Carlisle's `keyval` package for processing `\key=value` style macro arguments.

```

35 \RequirePackage{keyval}

```

16.2 Default Parameters

This section defines macros and lengths that will typically be executed or redefined by the user in the document preamble to initialize the document. (Not all of these are restricted to preamble usage, however. Many can be used throughout the document to switch styles for different sections or different songs.)

`\lyricfont` Define the font style to use for formatting song lyrics.
36 `\newcommand\lyricfont{\normalfont\normalsize}`

`\stitlefont` Define the font style to use for formatting song titles.
37 `\newcommand\stitlefont{%`
38 `\ifslides\sffamily\Huge\else\sffamily\slshape\Large\fi%`
39 `}`

`\versefont` By default, verses, choruses, and textual notes just allow the `\lyricfont` style to
`\chorusfont` continue.
`\notefont` 40 `\newcommand\versefont{}`
41 `\newcommand\chorusfont{}`
42 `\newcommand\notefont{}`

`\scripturefont` Define the font style to use for formatting scripture quotations (defaults to Zapf
Chancery).
43 `\newcommand\scripturefont{%`
44 `\usefont{OT1}{pzc}{mb}{it}%`
45 `\shiftdblquotes{-1.1\p@}\z@{-2\p@}\z@%`
46 `}`

`\printscrcite` Define the printing style for the citation at the end of a scripture quotation.
47 `\newcommand\printscrcite[1]{\sffamily\small#1}`

`\snumbgcolor` Define the background color used for shaded boxes containing song numbers, tex-
`\notebgcolor` tual notes, and index section headers, respectively. To turn off all shading for a
`\idxbgcolor` box type, use `\def<macroname>{}`.
48 `\newcommand\snumbgcolor{SongbookShade}`
49 `\newcommand\notebgcolor{SongbookShade}`
50 `\newcommand\idxbgcolor{SongbookShade}`

`\versejustify` Verses and choruses are both left-justified with hanging indentation equal to
`\chorusjustify` `\parindent`,
51 `\newcommand\versejustify{\justifyleft}`
52 `\newcommand\chorusjustify{\justifyleft}`

`\notejustify` Textual notes are fully justified when they are too long to fit in a single line.
53 `\newcommand\notejustify{%`
54 `\advance\baselineskip\p@relax%`
55 `\leftskip\z@skip\rightskip\z@skip%`
56 `\parfillskip\@flushglue\parindent\z@%`
57 `}`

`\placernote` Textual notes are placed flush-left. The single argument to this macro is horizontal
material that comprises the note. Usually it will consist of various hboxes and
specials that were produced by `\colorbox`.
58 `\newcommand\placernote[1]{%`
59 `\leftskip\z@skip\rightskip\@flushglue\SB@cbarshift%`
60 `\noindent#1\par%`
61 `}`

These counters define the current song number and verse number. They can be redefined by the user at any time.

```

62 \newcounter{songnum}
63 \newcounter{versenum}

\thesongnum By default, the song numbering style will simply be an arabic number. Redefine
\songnumstyle \thesongnum to change it. (The \songnumstyle macro is obsolete and exists only
for backward compatibility.)
64 \renewcommand\thesongnum{\songnumstyle{songnum}}
65 \newcommand\songnumstyle{}
66 \let\songnumstyle\arabic

\theversenum By default, the verse numbering style will simply be an arabic number. Redefine
\versenumstyle \theversenum to change it. (The \versenumstyle macro is obsolete and exists
only for backward compatibility.)
67 \renewcommand\theversenum{\versenumstyle{versenum}}
68 \newcommand\versenumstyle{}
69 \let\versenumstyle\arabic

\printsongnum Define the printing style for the large, boxed song numbers starting each song.
70 \newcommand\printsongnum[1]{\sffamily\bfseries\LARGE#1}

\printversenum Define the printing style for the verse numbers to the left of each verse.
71 \newcommand\printversenum[1]{\lyricfont#1.\ }

\placeversenum Verse numbers are placed flush-left. This is achieved by inserting horizontal glue
that reverses both the \leftskip and the \parindent. The single argument to
this macro is an hbox containing the verse number.
72 \newcommand\placeversenum[1]{%
73 \hskip-\leftskip\hskip-\parindent\relax%
74 \box#1%
75 }

\everyverse The following hooks allow users to insert material at the head of each verse or
\everychorus chorus.
76 \newcommand\everyverse{}
77 \newcommand\everychorus{}

\printchord Define the printing style for chords.
78 \newcommand\printchord[1]{\sffamily\slshape\large#1}

\chordlocals This hook is expanded at the start of the scoping group that surrounds every
chord name. Thus, it can be used to set any catcodes or definitions that should
be local to chord names.
79 \newcommand\chordlocals{}

```

<code>\versesep</code>	Specify the vertical distance between song verses. This gets set to a sentinel value by default; if the user doesn't redefine it by the end of the document preamble, it gets redefined to something sensible based on other settings. 80 <code>\newskip\versesep</code> 81 <code>\versesep123456789sp\relax</code>
<code>\afterpreludeskip</code> <code>\beforepostludeskip</code>	Users can specify the amount of vertical space that separates song prelude and postlude material from the body of the song by adjusting the following two macros. 82 <code>\newskip\afterpreludeskip</code> 83 <code>\afterpreludeskip=2\p@\@plus4\p@</code> 84 <code>\newskip\beforepostludeskip</code> 85 <code>\beforepostludeskip=2\p@\@plus4\p@</code>
<code>\baselineadj</code>	Define an adjustment factor for the vertical distance between consecutive lyric baselines. Setting this to zero accepts the default baseline distance computed by the songs package. 86 <code>\newskip\baselineadj</code> 87 <code>\baselineadj\z@skip</code>
<code>\clineparams</code>	The spacing between chords and the lyrics below them can be adjusted by changing the values of <code>\baselineskip</code> , <code>\lineskiplimit</code> , and <code>\lineskip</code> within the following macro. By default, <code>\baselineskip</code> is set to 2 points smaller than the height of the current (lyric) font, and <code>\lineskiplimit</code> and <code>\lineskip</code> are set so that chords intrude at most 2 points into the lyric below them. This helps to keep chords tight with lyrics. 88 <code>\newcommand\clineparams{%</code> 89 <code>\baselineskip\fontsize\p@%</code> 90 <code>\advance\baselineskip-2\p@%</code> 91 <code>\lineskiplimit-2\p@%</code> 92 <code>\lineskip-2\p@%</code> 93 <code>}</code>
<code>\parindent</code>	The <code>\parindent</code> length controls how far broken lyric lines are indented from the left margin. 94 <code>\parindent.25in</code>
<code>\idxheadwidth</code>	Specify the width of the head-boxes in a large index. 95 <code>\newlength\idxheadwidth</code> 96 <code>\setlength\idxheadwidth{1.5cm}</code>
<code>\songnumwidth</code>	Set the width of the song number boxes that begin each song. We guess a suitable width by typesetting the text "999." 97 <code>\newlength\songnumwidth</code> 98 <code>\settowidth\songnumwidth{\printsongnum{999.}}</code>
<code>\versenumwidth</code>	Set the width that is reserved for normal-sized verse numbers. (Verse numbers wider than this will indent the first line of lyrics.) 99 <code>\newlength\versenumwidth</code> 100 <code>\settowidth\versenumwidth{\printversenum{9\kern1em}}</code>

`\cbarwidth` This dictates the width of the vertical line placed to the left of choruses. Setting it to `0pt` eliminates the line entirely.

```

101 \newlength\cbarwidth
102 \setlength\cbarwidth\p@

```

`\sbarheight` This dictates the height of the horizontal line placed between each pair of songs. Setting it to `0pt` eliminates the line entirely.

```

103 \newlength\sbarheight
104 \setlength\sbarheight\p@

```

Column- and page-breaks should typically not occur within a verse or chorus unless they are unavoidable. Thus, we set the `\interlinepenalty` to a high number (1000).

```

105 \interlinepenalty\@m

```

`\vvpenalty` The following count registers define the line-breaking penalties inserted between verses, between choruses, after a verse followed by a chorus, after a chorus followed by a verse, and at `\brk` macros, respectively.

`\ccpenalty` The default value of 200 was chosen based on the following logic: Chord books should not yield underfull vbox warnings no matter how short their columns are. However, we still want to put as much material in each column as possible while avoiding intra-song column-breaks when they can be avoided. Chorded mode therefore sets `\colbotglue` with glue whose stretchability is half of the `\textheight`. Such glue will stretch at most twice its stretchability, yielding a badness of 800 in the worst case. The default `\vbadness` setting starts issuing warnings at badness 1000, so we set the penalties below to $1000 - 800 = 200$.

```

106 \newcount\vvpenalty\vvpenalty200
107 \newcount\ccpenalty\ccpenalty200
108 \newcount\vcpenalty\vcpenalty200
109 \newcount\cvpenalty\cvpenalty200
110 \newcount\brkpenalty\brkpenalty200

```

`\spenalty` The following penalty gets inserted between songs. Setting it to a proper value is a somewhat delicate balancing act. It should typically be something between 0 and the default penalties above, so for now it defaults to 100. To start each song on a fresh column/page, set it to -10000 or below.

```

111 \newcount\spenalty\spenalty100

```

`\songmark` The user can redefine the following macros to add \TeX marks for each song, each verse, or each chorus. Such marks are used by \LaTeX to define page headers and footers.

```

112 \newcommand\songmark{}
113 \newcommand\versemark{}
114 \newcommand\chorusmark{}

```

`\extendprelude` To just add some fields to the existing `\makeprelude` or `\makepostlude` without having to redefine them entirely, users can redefine `\extendprelude` or `\extendpostlude`. By default, the prelude has the scripture references followed by the authors, and the postlude has the copyright info followed by the licensing info.

```

115 \newcommand\extendprelude{\showrefs\showauthors}
116 \newcommand\extendpostlude{\songcopyright\ \songlicense\unskip}

```

`\idxheadfont` Users can redefine `\idxheadfont` to affect the font in which each capital letter that heads a section of a title index is rendered.

```

117 \newcommand\idxheadfont{\sffamily\bfseries\LARGE}

```

`\idxtitlefont` Users can redefine `\idxtitlefont` to affect the font in which song title index entries are rendered.

```

118 \newcommand\idxtitlefont{\sffamily\slshape}

```

`\idxlyricfont` Users can redefine `\idxlyricfont` to affect the font in which notable lines of lyrics are rendered in a title index.

```

119 \newcommand\idxlyricfont{\rmfamily}

```

`\idxscripfont` Users can redefine `\idxscripfont` to affect the font in which scripture references are rendered in a scripture index.

```

120 \newcommand\idxscripfont{\sffamily\small\slshape}

```

`\idxauthfont` Users can redefine `\idxauthfont` to affect the font in which contributor names are rendered in an author index.

```

121 \newcommand\idxauthfont{\small\bfseries}

```

`\idxrefsfont` Users can redefine `\idxrefsfont` to affect the font in which the list of song references on the right-hand-side of an index entry is typeset.

```

122 \newcommand\idxrefsfont{\normalfont\normalsize}

```

`\idxbook` Users can redefine `\idxbook` to dictate the book name header in a scripture index that begins each book of the bible.

```

123 \newcommand\idxbook[1]{\small\bfseries#1}

```

`\idxcont` Users can redefine `\idxcont` to dictate the column header in a scripture index after a column break falls within a book of the bible.

```

124 \newcommand\idxcont[1]{\small\textbf{#1} (continued)}

```

`\colbotglue` Glue of size `\colbotglue` is inserted at the bottom of each column. We use a macro instead of a glue register so that this can be redefined in terms of variable quantities such as `\textheight`.

```

125 \newcommand\colbotglue{}
126 \let\colbotglue\z@skip

```

`\lastcolglue` Glue of size `\lastcolglue` is inserted at the bottom of the last column.

```

127 \newcommand\lastcolglue{}
128 \let\lastcolglue\@flushglue

\minfrets Define the minimum number of fret rows that should appear in tablature diagrams.
129 \newcount\minfrets\minfrets4

\SB@colwidth Define a length to store the computed width of each column in a multi-column
song page. The user shouldn't set this one directly, but some users might want to
refer to it in calculations.
130 \newdimen\SB@colwidth

```

16.3 Package Options

This section defines code associated with the various option settings that can be specified on the `\usepackage` line. Many of these options can also be turned on or off subsequent to the `\usepackage` line, so macros for doing that are also located here. The options are not actually processed until §16.17 because some of the macros defined here refer to macros that have not yet been defined.

`slides` (*Default: off*) Turning this option on generates a book of overhead slides—one for each song. It really just amounts to changing various parameter settings. Elsewhere in the code we also consult `\ifslides` to determine a few default parameter settings and to use a different song preamble structure. All the parameter changes below are local to the current scope; so to undo slides mode, just put `\slides` within a group and end the group wherever you want the slides settings to end.

```

131 \DeclareOption{slides}{\slides}
132 \newcommand\slides{%
133   \slidestrue%
134   \def\lyricfont{\normalfont\huge}%
135   \def\chorusfont{\slshape}%
136   \def\versejustify{\justifycenter}%
137   \let\chorusjustify\versejustify
138   \def\placenote##1{\justifycenter\noindent##1\par}%
139   \scriptureoff%
140   \onesongcolumn%
141   \ifSB@preamble\ifSB@chordedspec\else\SB@chordsoff\fi\fi%
142   \spenalty-\@M%
143   \let\colbotglue\@flushglue%
144   \setlength\cbarwidth\z@%
145   \setlength\sbarheight\z@%
146 }

```

`\justifyleft` The `\justifyleft` macro sets up an environment in which lyrics are left-justified with hanging indentation equal to `\parindent`. It reserves spaces for verse numbers if used in a verse, and reserves space for the vertical bar left of choruses if used in a chorus.

```

147 \newcommand\justifyleft{%
148   \leftskip\parindent%
149   \ifSB@inverse\advance\leftskip\versenumwidth\fi%
150   \SB@cbarshift%
151   \parindent-\parindent%
152 }

```

`\justifycenter` The `\justifycenter` macro sets up an environment in which lyrics are centered on each line. Verse numbers continue to be placed flush-left, but `\placeversenum` is temporarily redefined to keep the rest of the line containing a verse number centered.

```

153 \newcommand\justifycenter{%
154   \centering\SB@cbarshift\rightskip\leftskip%
155   \def\placeversenum##1{%
156     \hskip-\leftskip\hskip-\parindent\relax%
157     \hangindent-\wd##1\hangafter\m@ne%
158     \box##1\hfil%
159   }%
160 }

```

`unouter` (*Default: off*) Several macros provided by the `songs` package are, by default, declared `\outer` to aid in debugging. However, unusual documents may need to use these macros within larger constructs. To do so, use the `unouter` option to prevent any of the macros supplied by this package from being declared `\outer`.

```

161 \newcommand\SB@outer{\outer}
162 \DeclareOption{unouter}{\let\SB@outer\relax}

```

`rawtext` (*Default: off*) Instead of generating a document, this dumps a text version of the song book to a file. This option can only be set in the `\usepackage` line because it dictates many top-level macro definitions. Turning `rawtext` on turns off the indexes by default, but this can be overridden by explicitly setting index options. (Note: Using `rawtext` with indexes turned on doesn't actually work yet, but might be added in a future revision.)

```

163 \DeclareOption{rawtext}{\rawtexttrue\indexsoff}

```

`nopdfindex` (*Default: off*) Inhibit the creation of the bookmark index in pdf files. This option can only be set in the `\usepackage` line because initializing the `pdfbookmark` library at all causes a (possibly empty) bookmark index to be created.

```

164 \DeclareOption{nopdfindex}{\pdfindexfalse}

```

`noshading` (*Default: off*) Inhibit all shaded boxes (e.g., if the `color` package is unavailable). This option can only be set in the `\usepackage` line because the `color` package must be loaded in the preamble if at all. (Note: In a future release this might be extended to be modifiable throughout the preamble.)

```

165 \DeclareOption{noshading}{\SB@colorboxesfalse}

```

`noindexes` (*Default: off*) Suppress generation of index files and displaying of in-document indexes. The `\indexeson` and `\indexesoff` macros can be used elsewhere to toggle display of indexes. Index-regeneration will occur if indexes are turned on by the end of the document.

```

166 \DeclareOption{noindexes}{\indexesoff}
167 \newcommand\indexeson{\songindexestruer}
168 \newcommand\indexesoff{\songindexesfalse}

```

`\ifSB@measurespec` The `showmeasures` and `chorded` options interact in the sense that by default, switching one of them on or off switches the other on or off as well. However, if the user explicitly says that one should be on or off, then switching the other shouldn't affect it. To produce this behavior, we need two extra conditionals to remember if each of these options has been explicitly specified by the user or if it is still in a default state.

```

169 \newif\ifSB@measurespec
170 \newif\ifSB@chordedspec

```

`chorded` (*Default: chorded*) Determines whether chords should be shown. This option can be set in the `\usepackage` line or toggled elsewhere with the `\chordson` and `\chordsoff` macros. Chords cannot be turned on in conjunction with the `rawtext` option. If chords are turned on by the end of the preamble, no attempt will be made to balance columns on each page.

`\SB@chordson`

`\SB@chordsoff`

```

171 \DeclareOption{chorded}{\chordson}
172 \DeclareOption{lyric}{\chordsoff}
173 \newcommand\chordson{\SB@chordedspectruer\SB@chordson}
174 \newcommand\chordsoff{\SB@chordedspectruer\SB@chordsoff}
175 \newcommand\SB@chordson{%
176   \ifrawtext%
177     \SB@errrtopt%
178   \else%
179     \chordedtrue\lyricfalse%
180     \let\SB@bracket\SB@chord%
181     \let\SB@rechord\SB@@rechord%
182     \let\SB@ch\SB@ch@on%
183     \ifSB@measurespec%
184       \ifmeasures\SB@measureson\else\SB@measuresoff\fi%
185     \else%
186       \SB@measureson%
187     \fi%
188     \ifSB@preamble\def\colbotglue{\z@\@plus.5\textheight}\fi%
189     \SB@setbaselineskip%
190   \fi%
191 }
192 \newcommand\SB@chordsoff{%
193   \chordedfalse\lyrictrue%
194   \def\SB@bracket##1{\ignorespaces}%
195   \let\SB@rechord\relax%
196   \let\SB@ch\SB@ch@off%

```



```

197 \ifSB@measurespec%
198 \ifmeasures\SB@measureson\else\SB@measuresoff\fi%
199 \else%
200 \SB@measuresoff%
201 \fi%
202 \ifSB@preamble\let\colbotglue\z@skip\fi%
203 \SB@setbaselineskip%
204 }

showmeasures (Default: showmeasures if chorded, nomeasures otherwise) Determines whe-
nomeasures ther measure bars and meter notes should be shown. Option can be set in the
\measureson \usepackage line or toggled elsewhere with the \measureson and \measuresoff
\measuresoff macros.
\SB@measureson 205 \DeclareOption{showmeasures}{\measureson}
\SB@measuresoff 206 \DeclareOption{nomeasures}{\measuresoff}
207 \newcommand\measureson{\SB@measurespectrue\SB@measureson}
208 \newcommand\measuresoff{\SB@measurespectrue\SB@measuresoff}
209 \newcommand\SB@measureson{%
210 \measurestrue%
211 \let\SB@mbar\SB@makembar%
212 \ifchorded%
213 \let\SB@mch\SB@mch@on%
214 \else%
215 \let\SB@mch\SB@mch@m%
216 \fi%
217 \ifSB@inverse\SB@loadactives\fi%
218 \ifSB@inchorus\SB@loadactives\fi%
219 }
220 \newcommand\SB@measuresoff{%
221 \measuresfalse%
222 \let\SB@mbar\@gobbletwo%
223 \ifchorded%
224 \let\SB@mch\SB@ch@on%
225 \else%
226 \let\SB@mch\SB@ch@off%
227 \fi%
228 \ifSB@inverse\SB@loadactives\fi%
229 \ifSB@inchorus\SB@loadactives\fi%
230 }

transposecapos (Default: off) If set, the \capo macro transposes the song instead of printing
a note to use a capo. Use this option to generate a chord book for pianists who
have trouble transposing or guitarists who don't have capos.
231 \DeclareOption{transposecapos}{\transcapostrue}

noscripture (Default: off) Inhibits the display of scripture quotes. This option can also be
\scriptureon toggled on and off anywhere with the \scriptureon and \scriptureoff macros.
\scriptureoff 232 \DeclareOption{noscripture}{\SB@omitscriptrue}
233 \newcommand\scriptureon{\SB@omitscripfalse}
234 \newcommand\scriptureoff{\SB@omitscriptrue}

```

`onesongcolumn` (Default: *onesongcolumn* is the default if generating slides or rawtext, *twosongcolumns* otherwise) The number of columns per page is specified using the following package options and macros. In `rawtext` mode it must remain set to one column per page. The entire page-making system can be turned off by setting the number of columns to zero. This will cause each song to be contributed to the current vertical list without any attempt to form columns; the enclosing environment must handle the page layout. Probably this means that `\repchoruses` will not work, since an external package won't know to insert repeated choruses when building pages.

```

235 \DeclareOption{twosongcolumns}{\SB@numcols\tw@}
236 \DeclareOption{onesongcolumn}{\SB@numcols\@ne}
237 \newcommand\songcolumns[1]{%
238   \SB@cnt#1\relax%
239   \ifnum\SB@cnt=\SB@numcols\else%
240     \ifSB@preamble\else{\SB@clearpage}\fi%
241   \fi%
242   \SB@numcols\SB@cnt%
243   \ifnum\SB@numcols>\z@%
244     \SB@colwidth-\columnsep%
245     \multiply\SB@colwidth\SB@numcols%
246     \advance\SB@colwidth\columnsep%
247     \advance\SB@colwidth\textwidth%
248     \divide\SB@colwidth\SB@numcols%
249   \else%
250     \ifrepchorus\SB@warnrc\fi%
251   \fi%
252 }
253 \newcommand\onesongcolumn{\songcolumns\@ne}
254 \newcommand\twosongcolumns{\songcolumns\tw@}

```

`\includeonlysongs` Display only a select list of songs and ignore the rest.

```

\songlist 255 \newcommand\songlist{}
256 \newcommand\includeonlysongs[1]{%
257   \ifSB@songsenv\SB@errpl\else%
258     \partiallisttrue%
259     \renewcommand\songlist{#1}%
260   \fi%
261 }

```

`\nosongnumbers` The user can turn off song numbering with the following macro.

```

262 \newcommand\nosongnumbers{\setlength\songnumwidth\z@}

```

`\noversenumbers` The user can turn off verse numbering with the following macro.

```

263 \newcommand\noversenumbers{%
264   \renewcommand\printversenum[1]{}%
265   \setlength\versenumwidth\z@%
266 }

```

`\repchoruses` Using `\repchoruses` causes choruses to be automatically repeated on subsequent pages of the song. The feature requires ε -TeX because the supporting code needs an extended mark register class.

```

267 \ifSB@etex
268   \newcommand\repchoruses{%
269     \ifnum\SB@numcols<\@ne\SB@warnrc\fi%
270     \repchorustrue%
271   }
272 \else
273   \newcommand\repchoruses{\SB@erretex}
274 \fi
275 \newcommand\norepchoruses{\repchorusfalse}

```

`\sepverses` The following penalty settings cause verses and choruses to be separated onto different slides when in slides mode, except that consecutive choruses remain together when they fit.

```

276 \newcommand\sepverses{%
277   \vvpentalty-\@M%
278   \ccpenalty100 %
279   \vcpenalty\vvpentalty%
280   \cvpenalty\vvpentalty%
281   \let\colbotglue\@flushglue%
282 }

```

Some option settings, margins, and other lengths are finalized at the end of the preamble. That code is below.

```

283 \AtBeginDocument{
    If the user hasn't set the \versesep, set it to the default.
284   \SB@setversesep
    Initialize page layout algorithm.
285   \songcolumns\SB@numcols
    Macros used after this point occur outside the preamble.
286   \SB@preamblefalse
287 }

```

16.4 Page-builder

The following macros handle the building of pages that contain songs. They compute where best to place each song (e.g., whether to place it in the current column or move to the next column or page). The output routines for generating a partial list of songs in a specified order also can be found here.

`\SB@songbox` The most recently processed song (or scripture quotation) is stored in this box.

```

288 \newbox\SB@songbox

```

`\SB@numcols` Reserve two count registers to hold the total number of columns and the current column number, respectively.

```

289 \newcount\SB@numcols\SB@numcols\tw@
290 \newcount\SB@colnum

```

`\SB@colbox` Reserve a box register to hold the current column in progress.

```

291 \newbox\SB@colbox

```

`\SB@colbox` Reserve a box register to hold the current page in progress.

```

292 \newbox\SB@pgbox

```

`\SB@mrkbox` Reserve a box register to hold marks that migrate out of songs as they get split into columns and pages.

```

293 \newbox\SB@mrkbox

```

`\SB@maxmin` The following helper macro takes the max or min of two dimensions. If $\langle arg2 \rangle = "<"$, it sets $\langle arg1 \rangle$ to the maximum of $\langle arg1 \rangle$ and $\langle arg3 \rangle$. If $\langle arg2 \rangle = ">"$, it sets $\langle arg1 \rangle$ to the minimum of $\langle arg1 \rangle$ and $\langle arg3 \rangle$.

```

294 \newcommand\SB@maxmin[3]{\ifdim#1#2#3#1#3\fi}

```

`\SB@mkpage` The following macro is the heart of the page-building engine. It splits the contents of a box into a page of columns. If `\repchoruses` is active, the contents of `\SB@chorusbox` are additionally inserted into fresh columns created during the spitting process. The macro arguments are:

1. the box b to split (must not be `\SB@box`, which is used as a temp register),
2. a count register i equaling column index (zero or greater) where the content of b is to begin, and
3. the desired column height.

Box b is split and i is incremented until i reaches `\SB@numcols` or b is emptied, whichever occurs first. If b is emptied first, the final column is *not* contributed; instead it is left in b and i is left equal to the index of the column that would have been added if b had been emptied. This allows the next call to reconsider whether to end the current column here or add some or all of the next contribution to it. Box b and count register i are globally modified. If `\SB@updatepage` is not redefined, boxes `\SB@pgbox` and `\SB@mrkbox` are also globally modified based on the results of the split.

```

295 \newcommand\SB@mkpage[3]{%
296   \begingroup%
297   \splitmaxdepth\maxdepth\splittopskip\z@skip%
298   \global\setbox#1\vbox{%
299     \unvbox#1%
300     \nointerlineskip%
301     \null%
302     \vfil%

```

```

303     }%
304     \loop\ifnum#2<\SB@numcols%
305         \setbox\SB@box\vsplit#1to#3\relax%
306         \ifvoid#1%
307             #2\SB@numcols%
308         \else%
309             \SB@updatepage%
310             \global\advance#2\@ne%
311             \ifrepchorus\ifvoid\SB@chorusbox\else%
312                 \SB@insertchorus#1%
313             \fi\fi%
314         \fi%
315     \repeat%
316     \global\setbox#1\vbox{%
317         \unvbox\SB@box%
318         \unvbox#1%
319         \unskip%
320         \setbox\SB@box\lastbox%
321     }%
322 \endgroup%
323 }

```

\SB@migrate Migrate a mark out of a recently split vertical list, but do not insert superfluous empty marks that may override previous marks.

```

324 \newcommand\SB@migrate[1]{%
325     \SB@toks\expandafter{#1}%
326     \edef\SB@temp{\the\SB@toks}%
327     \ifx\SB@temp\@empty\else\mark{\the\SB@toks}\fi%
328 }

```

\SB@updatepage Update boxes \SB@pgbox and \SB@mrkbox immediately after splitting the contents of \SB@colbox.

```

329 \newcommand\SB@updatepage{%
330     \global\setbox\SB@mrkbox\vbox{%
331         \unvbox\SB@mrkbox%
332         \SB@migrate\splitfirstmark%
333         \SB@migrate\splitbotmark%
334     }%
335     \global\setbox\SB@pgbox\hbox{%
336         \SB@dimen\SB@colwidth%
337         \advance\SB@dimen\columnsep%
338         \multiply\SB@dimen\SB@colnum%
339         \advance\SB@dimen-\wd\SB@pgbox%
340         \unhbox\SB@pgbox%
341         \ifdim\SB@dimen=z@\else\hskip\SB@dimen\relax\fi%
342         \box\SB@box%
343     }%
344 }

```

- `\SB@droppage` This alternate definition of `\SB@updatepage` drops the just-created page instead of contributing it. This allows `\SB@mkpage` to be called by the song-positioning algorithm as a trial run without outputting anything.
- ```

345 \newcommand\SB@droppage{\setbox\SB@box\box\voidb@x}

```
- `\SB@output` This is the main output routine for the page-builder. It repeatedly calls `\SB@mkpage`, emitting pages as they are completed, until the remaining content of box `\SB@colbox` is not enough to fill a column. This final, in-progress column is left unfinished, pending future contributions.
- ```

346 \newcommand\SB@output{%
347   \ifnum\SB@numcols>\z@\begingroup%
348     \loop%
349       \SB@dimen\textheight%
350       \ifinner\else\advance\SB@dimen-\pagetotal\fi%
351       \SB@mkpage\SB@colbox\SB@colnum\SB@dimen%
352       \ifnum\SB@colnum<\SB@numcols\else%
353         \unvbox\SB@mrkbox%
354         \ifinner\else\kern\z@\fi%
355         \box\SB@pgbox%
356         \ifinner\else\vfil\break\vskip\vsize\relax\fi%
357         \global\SB@colnum\z@%
358       \repeat%
359   \endgroup\else%
360     \unvbox\SB@colbox\unskip%
361   \fi%
362 }

```
- `\SB@putboxes` Create a vertical list consisting of the already committed contents of the current column plus the most recently submitted song box. The \LaTeX primitive that should be used to contribute each box is specified in the first argument.
- ```

363 \newcommand\SB@putboxes[1]{%
364 \SB@dimen\ifnum\SB@numcols>\z@\ht\SB@colbox\else\p@\fi%
365 #1\SB@colbox%
366 \ifdim\SB@dimen>\z@%
367 \SB@breakpoint\spenalty%
368 \ifdim\sbarheight>\z@%
369 \vskip-\sbarheight\relax%
370 \fi%
371 \fi%
372 #1\SB@songbox%
373 }

```
- `\SB@nextcol` Force  $n$  column breaks, where  $n$  is given by the first argument. The first created column is finished with the glue specified in the second argument. When the second argument is `\@flushglue`, this forces a break that leaves whitespace at the bottom of the column. When it's `\colbotglue`, it acts like a natural column break chosen by the page-breaker. However, if the current column is empty, `\@flushglue` is always used so that an empty column will result.

```

374 \newcommand\SB@nextcol[2]{%
375 \ifnum#1>\z@%
376 \ifnum\SB@numcols>\z@%
377 \global\setbox\SB@colbox\vbox{%
378 \SB@cnt#1\relax%
379 \SB@dimen\ht\SB@colbox%
380 \unvbox\SB@colbox%
381 \unskip%
382 \ifdim\SB@dimen>\z@%
383 \vskip#2\relax%
384 \break%
385 \advance\SB@cnt\m@ne%
386 \fi%
387 \loop\ifnum\SB@cnt>\z@%
388 \nointerlineskip%
389 \null%
390 \vfil%
391 \break%
392 \advance\SB@cnt\m@ne%
393 \repeat%
394 }%
395 \SB@output%
396 \else%
397 \ifnum\lastpenalty=-\@M\null\fi%
398 \break%
399 \fi%
400 \fi%
401 }

```

`\SB@selectcol` This is the entrypoint to the song-positioning algorithm. It gets defined by `\songpos` to either `\SB@@selectcol` (below) or `\relax` (when song-positioning is turned off).

```
402 \newcommand\SB@selectcol{}
```

`\SB@@selectcol` Songs should be squeezed in wherever they fit, but breaking a column or page within a song should be avoided. The following macro outputs zero or more column breaks to select a good place for `\SB@songbox` to be contributed to the current (or the next) page. The number of column breaks is determined by temporarily setting `\SB@updatepage` to `\SB@droppage` and then calling the `\SB@mkpage` algorithm under various conditions to see how many columns it would contribute if we start the current song at various positions.

```

403 \newcommand\SB@@selectcol{%
404 \begingroup%
405 \SB@cnt\z@%
406 \vbadness\@M\vfuze\maxdimen%
407 \let\SB@updatepage\SB@droppage%
408 \SB@dimen\textheight%
409 \ifinner\else\advance\SB@dimen-\pagetotal\fi%
410 \setbox\SB@boxii\vbox{\SB@putboxes\unvcopy}%

```

```

411 \SB@cntii\SB@colnum%
412 \SB@mkpage\SB@boxii\SB@cntii\SB@dimen%
413 \SB@spos%
414 \global\SB@cnt\SB@cnt%
415 \endgroup%
416 \SB@nextcol\SB@cnt\colbotglue%
417 }

\SB@spbegnew Begin a trial typesetting of the current song on a fresh page to see if it fits within
a page.
418 \newcommand\SB@spbegnew{%
419 \setbox\SB@boxiii\copy\SB@songbox%
420 \SB@cntii\z%
421 \SB@mkpage\SB@boxiii\SB@cntii\textheight%
422 }

\SB@spextold Tentatively extend the song previously typeset on the current even page to the next
odd page to see if it fits on a double-page. If the current page is odd-numbered, do
nothing since extending the song to the next page would introduce a page-turn.
423 \newcommand\SB@spextold{%
424 \ifodd\c@page\else%
425 \SB@cntii\z%
426 \SB@mkpage\SB@boxii\SB@cntii\textheight%
427 \fi%
428 }

\SB@spextnew Extend the trial typesetting started with \SB@spbegnew to a second page to see if
the song fits on a fresh double-page.
429 \newcommand\SB@spextnew{%
430 \SB@cntii\z%
431 \SB@mkpage\SB@boxiii\SB@cntii\textheight%
432 }

\SB@spdblp Compute the number of column breaks required to shift the current song to the
next double-page if the result of the last test run fits within its page (as indicated
by counter \SB@cntii). Otherwise leave the requested number of column breaks
set to zero.
433 \newcommand\SB@spdblp{%
434 \ifnum\SB@cntii<\SB@numcols%
435 \SB@cnt\SB@numcols%
436 \advance\SB@cnt-\SB@colnum%
437 \if@twoside\ifodd\c@page\else%
438 \advance\SB@cnt\SB@numcols%
439 \fi\fi%
440 \fi%
441 }

```



`\SB@sposi` This is the level-1 song positioning algorithm. It moves songs to the next double-page only if doing so would avoid a page-turn that would otherwise appear within the song.

```

442 \newcommand\SB@sposi{%
443 \ifnum\SB@cntii<\SB@numcols\else\if@twoside%
444 \SB@spextold%
445 \fi\fi%
446 \ifnum\SB@cntii<\SB@numcols\else%
447 \SB@spbegnew%
448 \ifnum\SB@cntii<\SB@numcols\else\if@twoside%
449 \SB@spextnew%
450 \fi\fi%
451 \SB@spdblp%
452 \fi%
453 }

```

`\SB@sposii` This is the level-2 song-positioning algorithm. It moves songs to the next page or double-page if doing so avoids a page-break or page-turn that would otherwise appear within the song.

```

454 \newcommand\SB@sposii{%
455 \ifnum\SB@cntii<\SB@numcols\else%
456 \SB@spbegnew%
457 \ifnum\SB@cntii<\SB@numcols%
458 \SB@cnt\SB@numcols%
459 \advance\SB@cnt-\SB@colnum%
460 \else%
461 \if@twoside%
462 \SB@spextold%
463 \ifnum\SB@cntii<\SB@numcols\else%
464 \SB@spextnew%
465 \SB@spdblp%
466 \fi%
467 \fi%
468 \fi%
469 \fi%
470 }

```

`\SB@sposiii` This is the level-3 song-positioning algorithm. It moves songs to the next column, the next page, or the next double-page if doing so avoids a column-break, page-break, or page-turn that would otherwise appear within the song.

```

471 \newcommand\SB@sposiii{%
472 \ifnum\SB@cntii>\SB@colnum%
473 \SB@cnt\SB@colnum%
474 \advance\SB@cnt\@ne%
475 \ifnum\SB@cnt<\SB@numcols%
476 \setbox\SB@boxiii\copy\SB@songbox%
477 \SB@mkpage\SB@boxiii\SB@cnt\SB@dimen%
478 \advance\SB@cnt\m@ne%
479 \fi%

```

```

480 \ifnum\SB@cnt>\SB@colnum%
481 \SB@cnt\z@%
482 \SB@sposii%
483 \else%
484 \SB@cnt\@ne%
485 \fi%
486 \fi%
487 }

```

**\songpos** This is the macro by which the user adjusts the aggressiveness level of the song-positioning algorithm. See the macros above for what each level does.

```

488 \newcommand\songpos[1]{%
489 \ifcase#1%
490 \let\SB@selectcol\relax%
491 \let\SB@spos\relax%
492 \or%
493 \let\SB@selectcol\SB@@selectcol%
494 \let\SB@spos\SB@sposi%
495 \or%
496 \let\SB@selectcol\SB@@selectcol%
497 \let\SB@spos\SB@sposii%
498 \or%
499 \let\SB@selectcol\SB@@selectcol%
500 \let\SB@spos\SB@sposiii%
501 \else%
502 \SB@errspos%
503 \fi%
504 }

```

**\SB@spos** The **\SB@spos** macro gets redefined by **\songpos** above depending on the current song-positioning aggressiveness level. By default it is set to level 3.

```

505 \newcommand\SB@spos{}
506 \songpos\thr@@

```

**\SB@clearpage** Output all contributed material as a new page unless there is no contributed material. In that case do nothing (i.e., don't produce a blank page).

```

507 \newcommand\SB@clearpage{%
508 \SB@testtrue%
509 \ifvoid\SB@pgbox\ifvoid\SB@colbox\SB@testfalse\fi\fi%
510 \ifSB@test%
511 \SB@cnt\SB@numcols%
512 \advance\SB@cnt-\SB@colnum%
513 \SB@nextcol\SB@cnt\lastcolglue%
514 \fi%
515 }

```

**\SB@cleardpage** Like **\SB@clearpage** but shift to a fresh *even-numbered* page in two-sided documents. Note that this differs from L<sup>A</sup>T<sub>E</sub>X's **\cleardoublepage**, which shifts to

odd-numbered pages. Song books prefer starting things on even-numbered pages because this maximizes the distance until the next page-turn.

```

516 \newcommand\SB@clearpage{%
517 \SB@clearpage%
518 \if@twoside\ifodd\c@page%
519 \SB@nextcol\SB@numcols\@flushglue%
520 \fi\fi%
521 }

```

**\SB@stype** There are two song content submission types: column- and page-submissions. Page-submissions are page-width and go atop fresh pages unless the current page has only page-width material so far. Column-submissions are column-width and start a new page only when the current page is full. This macro gets set to the desired type for the current submission. Mostly it stays set to the default column-submission type.

```

522 \newcommand\SB@stype{\SB@stypcol}

```

**\SB@stypcol** Column-submissions contribute the contents of **\SB@songbox** to either the current column or the next column or page, depending on where it best fits.

```

523 \newcommand\SB@stypcol{%
524 \ifnum\SB@numcols>\z@%
525 \SB@selectcol%
526 \global\setbox\SB@colbox\vbox{\SB@putboxes\unvbox}%
527 \SB@output%
528 \else%
529 \unvbox\voidb@x%
530 \SB@breakpoint\spenalty%
531 \ifdim\sbarheight>\z@%
532 \vskip-\sbarheight\relax%
533 \fi%
534 \unvbox\SB@songbox%
535 \fi%
536 }

```

**\SB@styppage** Page-submissions go directly to the top of the nearest fresh page unless the current page has all page-width material so far.

```

537 \newcommand\SB@styppage{%
538 \ifnum\SB@numcols>\z@%
539 \SB@clearpage%
540 \unvbox\SB@songbox%
541 \null\nointerlineskip%
542 \else%
543 \unvbox\SB@songbox%
544 \fi%
545 }

```

`\SB@sgroup` This macro controls whether songs submitted to the page-builder are actually contributed to the final document when using `\includeonlysongs` to generate a partial list. If `\SB@sgroup` is empty, then the song is silently dropped. Otherwise it is contributed only if `\SB@sgroup` is a member of `\songlist`.

```

546 \newcommand\SB@sgroup{}
547 \let\SB@sgroup\@empty

```

`\SB@groupcnt` This counter assigns a unique integer to each item of a group. Environments that come before the group's song are numbered decreasingly from  $-1$ . The song itself has number 0. Environments that come after the song are numbered increasingly from 1.

```

548 \newcount\SB@groupcnt

```

`\SB@submitpart` When a song completes and we're generating a partial list, save the song in a box so that it can be submitted at the end of the section in the order specified by `\includeonlysongs`.

```

549 \newcommand\SB@submitpart{%
550 \ifx\SB@sgroup\@empty\else%
551 \@for\SB@temp:=\songlist\do{%
552 \ifx\SB@temp\SB@sgroup%
553 \edef\SB@tempii{\SB@sgroup @\the\SB@groupcnt}%
554 \expandafter\newbox\csname songbox@\SB@tempii\endcsname%
555 \global\expandafter\setbox
556 \csname songbox@\SB@tempii\endcsname\box\SB@songbox%
557 \global\expandafter\let%
558 \csname stype@\SB@tempii\endcsname\SB@stype%
559 \ifrepchorus\ifvoid\SB@chorusbox\else%
560 \expandafter\newbox\csname chbox@\SB@tempii\endcsname%
561 \global\expandafter\setbox%
562 \csname chbox@\SB@tempii\endcsname\box\SB@chorusbox%
563 \fi\fi%
564 \fi%
565 }%
566 \global\advance\SB@groupcnt%
567 \ifnum\SB@groupcnt<\z@\m@ne\else\@ne\fi%
568 \fi%
569 \setbox\SB@songbox\box\voidb@x%
570 \setbox\SB@chorusbox\box\voidb@x%
571 }

```

`\SB@submitsong` Submit the most recently finished song (or block of other vertical material) for output. If we're generating a partial list of songs, save it in a box instead of submitting it here. (The saved boxes will be submitted in the requested order at the end of the songs section.)

```

572 \newcommand\SB@submitsong{%
573 \ifpartiallist\SB@submitpart\else\SB@stype\fi%
574 }

```

`\SB@songlistbrk` These macros define the words that, when placed in a `\songlist`, force a column break at that point. Using `brk` produces a soft break (like `\brk`) that won't leave whitespace at the bottom of the broken column in lyric books. Using `nextcol` produces a hard break (like `\nextcol`) that may insert whitespace to finish the column. Using `sclearpage` moves to the next page if the current page is nonempty. Using `scleardpage` moves to the next double-page if the current double-page is nonempty.

```

575 \newcommand\SB@songlistbrk{}
576 \def\SB@songlistbrk{brk}
577 \newcommand\SB@songlistnc{}
578 \def\SB@songlistnc{nextcol}
579 \newcommand\SB@songlistcp{}
580 \def\SB@songlistcp{sclearpage}
581 \newcommand\SB@songlistcdp{}
582 \def\SB@songlistcdp{scleardpage}

```

`\commitsongs` If we're generating only a partial list, then wait until the end of the section and then output all the songs we saved in boxes in the order specified.

```

583 \newcommand\commitsongs{%
584 \ifpartiallist%
585 \ifnum\SB@numcols>\z%
586 \@for\SB@temp:=\songlist\do{%
587 \ifx\SB@temp\SB@songlistnc\SB@nextcol\@ne\@flushglue\else%
588 \ifx\SB@temp\SB@songlistbrk\SB@nextcol\@ne\colbotglue\else%
589 \ifx\SB@temp\SB@songlistcp\SB@clearpage\else%
590 \ifx\SB@temp\SB@songlistcdp\SB@cleardpage\else%
591 \SB@groupcnt\m@ne\SB@finloop%
592 \SB@groupcnt\z@\SB@finloop%
593 \fi\fi\fi\fi%
594 }%
595 \else%
596 \@for\SB@temp:=\songlist\do{%
597 \ifx\SB@temp\SB@songlistnc\vfil\break\else%
598 \ifx\SB@temp\SB@songlistbrk\break\else%
599 \ifx\SB@temp\SB@songlistcp\clearpage\else%
600 \ifx\SB@temp\SB@songlistcdp%
601 \clearpage%
602 \ifodd\c@page\null\newpage\fi%
603 \else%
604 \SB@groupcnt\m@ne\SB@finloop%
605 \SB@groupcnt\z@\SB@finloop%
606 \fi\fi\fi\fi%
607 }%
608 \fi%
609 \fi%
610 \SB@clearpage%
611 }

```

`\SB@finloop` While contributing saved material included by `\includeonlysongs`, this macro contributes each series of boxes grouped together as part of a `songgroup` environment.

```

612 \newcommand\SB@finloop{%
613 \loop\edef\SB@temp{ii\SB@temp @\the\SB@groupcnt}%
614 \expandafter\ifx%
615 \csname songbox@\SB@temp\endcsname\relax\else%
616 \setbox\SB@songbox\expandafter\box%
617 \csname songbox@\SB@temp\endcsname%
618 \expandafter\ifx\csname chbox@\SB@temp\endcsname\relax%
619 \repchorusfalse%
620 \else%
621 \repchorustrue%
622 \setbox\SB@chorusbox\expandafter\box%
623 \csname chbox@\SB@temp\endcsname%
624 \fi%
625 \csname stype@\SB@temp\endcsname%
626 \advance\SB@groupcnt\ifnum\SB@groupcnt<\z@\m@ne\else\@ne\fi%
627 \repeat%
628 }
```

`\SB@insertchorus` Insert a chorus into the first marked spot in the box given in the first argument. This is usually achieved by splitting the box at the first valid breakpoint after the first `\SB@cmark` in the box. The box is globally modified.

```

629 \newcommand\SB@insertchorus[i]{%
630 \vbadness\@M\vfuze\maxdimen%
631 \setbox\SB@box\copy#1%
632 \setbox\SB@box\vsplit\SB@box to\maxdimen%
633 \edef\SB@temp{\splitfirstmarks\SB@nocmarkclass}%
634 \ifx\SB@temp\SB@nocmark\else%
635 \edef\SB@temp{\splitfirstmarks\SB@cmarkclass}%
636 \ifx\SB@temp\SB@cmark%
637 \SB@dimen4096\p@%
638 \SB@dimenii\maxdimen%
639 \SB@dimeniii\SB@dimen%
640 \loop%
641 \SB@dimeniii.5\SB@dimeniii%
642 \setbox\SB@box\copy#1%
643 \setbox\SB@box\vsplit\SB@box to\SB@dimen%
644 \edef\SB@temp{\splitfirstmarks\SB@cmarkclass}%
645 \ifx\SB@temp\SB@cmark%
646 \SB@dimenii\SB@dimen%
647 \advance\SB@dimen-\SB@dimeniii%
648 \else%
649 \advance\SB@dimen\SB@dimeniii%
650 \fi%
651 \ifdim\SB@dimeniii>2\p@\repeat%
652 \setbox\SB@box\vsplit#1to\SB@dimenii%
653 \global\setbox#1\vbbox{%
```

```

654 \unvbox\SB@box\unskip%
655 \SB@inversefalse\SB@prevversettrue\SB@stanzabreak%
656 \SB@putbox\unvcopy\SB@chorusbox%
657 \SB@inversettrue\SB@prevversefalse\SB@stanzabreak%
658 \unvbox#1%
659 }%

```

However, if the first mark is a `\SB@lastcmark`, it means that this chorus should go after the last verse in the song. There is no valid breakpoint there, so to get a chorus into that spot, we have to do a rather ugly hack: We pull the bottom material off the box with `\unskip`, `\unpenalty`, and `\lastbox`, then insert the chorus, then put the bottom material back on. This works because the high-level structure of the bottom material should be static. Even if the user redefines `\makepostlude`, the new definition gets put in a single box that can be manipulated with `\lastbox`. However, if we ever change the high-level structure, we need to remember to change this code accordingly.

```

660 \else\ifx\SB@temp\SB@lastcmark%
661 \global\setbox#1\vbox{%
662 \unvbox#1%
663 \unskip%
664 \ifdim\sbarheight>\z@%
665 \setbox\SB@box\lastbox%
666 \unskip\unpenalty%
667 \fi%
668 \setbox\SB@box\lastbox%
669 \unskip\unskip%
670 \SB@inversefalse\SB@prevversettrue\SB@stanzabreak%
671 \marks\SB@nocmarkclass{\SB@nocmark}%
672 \unvcopy\SB@chorusbox%
673 \vskip\versesep\vskip\beforepostludeskip\relax%
674 \nointerlineskip\box\SB@box%
675 \ifdim\sbarheight>\z@%
676 \nobreak\vskip2\p@\@plus\p@%
677 \hrule\@height\sbarheight\@width\SB@colwidth%
678 \fi%
679 }%
680 \fi\fi%
681 \fi%
682 }}

```

`\nextcol` End the current column (inserting vertical space as needed). This differs from column breaks produced with `\brk`, which does not introduce any empty vertical space.

```

683 \newcommand\nextcol{%
684 \@ifstar{\SB@nextcol\@ne\@flushglue}%
685 {\ifpartiallist\else\SB@nextcol\@ne\@flushglue\fi}%
686 }

```

`\sclearpage` Move to the next page if the current page is nonempty.

```

687 \newcommand\sclearpage{%
688 \@ifstar\SB@clearpage{\ifpartiallist\else\SB@clearpage\fi}%
689 }

\sclearpage Move to the next even-numbered page if the current page is odd or nonempty.
690 \newcommand\scleardpage{%
691 \@ifstar\SB@cleardpage{\ifpartiallist\else\SB@cleardpage\fi}%
692 }

```

## 16.5 Songs

The following macros handle the parsing and formatting of the material that begins and ends each song.

\SB@lop The following macros were adapted from Donald Knuth's *The T<sub>E</sub>Xbook*, for manipulating lists of the form `||item1||item2||...||itemN||`.

```

\SB@emptylist 693 \newcommand\SB@lop[1]{\expandafter\SB@@lop\the#1\SB@@lop#1}
\SB@ifempty 694 \newcommand\SB@@lop{}
695 \def\SB@lop\#1\#2\SB@lop#3#4{\global#3{\#2}\global#4{#1}}
696 \newcommand\SB@emptylist{}
697 \def\SB@emptylist{\}
698 \newcommand\SB@ifempty[3]{%
699 \edef\SB@temp{\the#1}%
700 \ifx\SB@temp\SB@emptylist#2\else#3\fi%
701 }

```

\SB@titlelist These registers hold the full list of titles for the current song and the tail list of titles that has not yet been iterated over.

```

702 \newtoks\SB@titlelist
703 \newtoks\SB@titletail

```

\songtitle The \songtitle macro will initially hold the primary title of the current song. The user can iterate over titles using \nexttitle or \foreachtitle.

```

704 \newcommand\songtitle{}

```

\resettitles Initialize the title list iterator.

```

705 \newcommand\resettitles{%
706 \global\SB@titletail\SB@titlelist%
707 \nexttitle%
708 }

```

\nexttitle Advance the title list iterator to the next title.

```

709 \newcommand\nexttitle{%
710 \SB@ifempty\SB@titletail{%
711 \global\let\songtitle\relax%
712 }{%
713 \SB@lop\SB@titletail\SB@toks%
714 \edef\songtitle{\the\SB@toks}%
715 }%
716 }

```



`\foreachtitle` Execute a block of code for each remaining title in the title list.

```

717 \newcommand\foreachtitle[1]{%
718 \ifx\songtitle\relax\else%
719 \loop#1\nexttitle\ifx\songtitle\relax\else\repeat%
720 \fi%
721 }

```

`\ifSB@insong` To help the user locate errors, keep track of which environments we're inside and

`\ifSB@intersong` immediately signal an error if someone tries to use a song command inside a

`\ifSB@inverse` scripture quotation, etc.

`\ifSB@inchorus`

```

722 \newif\ifSB@songsenv\SB@songsenvfalse
723 \newif\ifSB@insong\SB@insongfalse
724 \newif\ifSB@intersong\SB@intersongfalse
725 \newif\ifSB@inverse\SB@inversefalse
726 \newif\ifSB@inchorus\SB@inchorusfalse

```

`\SB@closeall` If an error is detected using one of the above, the following macro will contain a macro sequence sufficient to end the unclosed environment, hopefully allowing processing to continue.

```

727 \newcommand\SB@closeall{}

```

`\SB@rawrefs` The current song's scripture references, authors, copyright info, and copyright

`\songauthors` license information are stored in these macros.

`\songcopyright`

```

728 \newcommand\SB@rawrefs{}

```

`\songlicense`

```

729 \newcommand\songauthors{}
730 \newcommand\songcopyright{}
731 \newcommand\songlicense{}

```

`\songrefs` When the user asks for the song's scripture references, rather than give them the raw token list that the author entered, we return a prettier version in which spaces, dashes, and penalties have been adjusted. The prettier version is stored in the following control sequence.

```

732 \newcommand\songrefs{}

```

`\setlicense` The user sets the licensing info for the current song with this command.

```

733 \newcommand\setlicense{\gdef\songlicense}

```

`\newsongkey` Defining a new key for `\beginsong` is just like the `keyval` package's `\define@key`

`\SB@clearbskeys` macro except that we must also define some initializer code for each key. This provides an opportunity to clear registers before each song. (Otherwise when a key wasn't specified, we'd inherit the old values from the previous song.)

```

734 \newcommand\SB@clearbskeys{}
735 \newcommand\newsongkey[2]{%
736 \expandafter\gdef\expandafter\SB@clearbskeys\expandafter%
737 {\SB@clearbskeys#2}%
738 \define@key{beginsong}{#1}%
739 }

```

Define keys `sr`, `by`, `cr`, `li`, `index`, and `ititle` for scripture references, authors, copyright info, licensing info, lyric index entries, and alternate title index entries, respectively.

```

740 \newsongkey{sr}{\def\SB@rawrefs{}\gdef\songrefs{}}
741 {\def\SB@rawrefs{#1}\SB@parsesrefs{#1}}
742 \newsongkey{by}{\def\songauthors{}}{\def\songauthors{#1}}
743 \newsongkey{cr}{\def\songcopyright{}}{\def\songcopyright{#1}}
744 \newsongkey{li}{\setlicense{}}{\setlicense{#1}}
745 \newsongkey{index}{\indexentry{#1}}
746 \newsongkey{ititle}{\indextitleentry{#1}}

```

`song` Parse the arguments of a `\beginsong` macro. The `\beginsong` macro supports two syntaxes. The preferred syntax takes the song title(s) as its first argument and an optional keyval list in brackets as its second argument. A legacy syntax `\SB@bsoldfmt` supports four arguments, all enclosed in braces, which are: the title(s), scripture references, authors, and copyright info.

```

747 \newenvironment{song}{\beginsong}{\SB@endsong}
748 \newcommand\beginsong[1]{%
749 \ifSB@insong\SB@errboo\SB@closeall\fi%
750 \ifSB@intersong\SB@errbor\SB@closeall\fi%
751 \SB@insongtrue%
752 \def\SB@closeall{\endsong}%
753 \SB@parsetitles{#1}%
754 \global\setbox\SB@songwrites\box\voidb@x%
755 \SB@clearbskeys%
756 \@ifnextchar[\SB@bskvfmt\SB@@beginsong%
757 }
758 \newcommand\SB@@beginsong{%
759 \@ifnextchar\bgroup\SB@bsoldfmt\SB@@@beginsong%
760 }
761 \newcommand\SB@bsoldfmt[3]{%
762 \SB@bskvfmt[sr={#1},by={#2},cr={#3}]%
763 }
764 \newcommand\SB@bskvfmt{}
765 \def\SB@bskvfmt[#1]{%
766 \setkeys{beginsong}{#1}%
767 \SB@@@beginsong%
768 }

```

`\SB@@@beginsong` Begin typesetting a song. Beginning a song involves typesetting the title and other info, adding entries to the indexes, and setting up the environment in which verses and choruses reside.

```

769 \newcommand\SB@@@beginsong{%
770 \global\SB@stanzafalse%
771 \setbox\SB@chorusbox\box\voidb@x%
772 \SB@gotchorusfalse%
773 \setbox\SB@songbox\vbox\bgroup\begingroup%
774 \ifnum\SB@numcols>\z@\hsize\SB@colwidth\fi%
775 \leftskip\z@skip\rightskip\z@skip%

```

```

776 \parfillskip\@flushglue\parskip\z@skip%
777 \SB@raggedright%
778 \global\SB@transposefactor\z@%
779 \global\SB@cr@{\}\%
780 \protected@edef\@currentlabel{\p@songnum\thesongnum}%
781 \setcounter{versenum}{1}%
782 \SB@prevversettrue%
783 \meter44%
784 \resettitles%
785 \SB@addtoindexes\songtitle\SB@rawrefs\songauthors%
786 \nexttitle%
787 \foreachtitle{\expandafter\SB@addtotitles\expandafter{\songtitle}}%
788 \resettitles%
789 \lyricfont%
790 \SB@setbaselineskip%
791 }

```

`\SB@endsong` Ending a song involves creating the song header (with `\makeprelude`), creating the song footer (with `\makepostlude`), and then assembling everything together into the `\SB@songbox`. The box is then submitted to the page-builder via `\SB@submitsong`. We do things this way instead of just contributing material directly to the main vertical list because submitting material song by song allows for a more sophisticated page-breaking algorithm than is possible with  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 's built-in algorithm.

```

792 \newcommand\SB@endsong{%
793 \ifSB@insong%
794 \ifSB@inverse\SB@erreov\endverse\fi%
795 \ifSB@inchorus\SB@erreoc\endchorus\fi%
796 \global\SB@skip\versesep%
797 \unskip%
798 \ifrepchorus\ifvoid\SB@chorusbox\else%
799 \ifSB@prevverse\ifvnumbered%
800 \marks\SB@cmarkclass{\SB@lastcmark}%
801 \fi\fi%
802 \fi\fi%
803 \endgroup\egroup%
804 \setbox\SB@songbox\vbox{%
805 \songmark%
806 \unvbox\SB@songwrites%
807 \ifnum\SB@numcols>\z@\hsize\SB@colwidth\fi%
808 \leftskip\z@skip\rightskip\z@skip%
809 \parfillskip\@flushglue\parskip\z@skip\parindent\z@%
810 \ifdim\sbarheight>\z@%
811 \hrule\@height\sbarheight\@width\hsize%
812 \nobreak\vskip5\p@%
813 \fi%
814 \ifpdfindex\beginngroup%
815 \ifx\pdfbookmark\undefined\else%
816 \ifx\pdfbookmark\relax\else%

```

```

817 \resettitles%
818 \pdfbookmark[\ifnum\c@section=\z@1\else2\fi]%
819 {\thesongnum. \songtitle}%
820 {song\theSB@songsnum-\thesongnum}%
821 \fi\fi%
822 \endgroup\fi%
823 \vbox{\makeprelude}%
824 \nobreak\vskip\SB@skip%
825 \vskip\afterpreludeskip\relax%
826 \unvbox\SB@songbox%
827 \nobreak\vskip\SB@skip%
828 \vskip\beforepostludeskip\relax%
829 \nointerlineskip%
830 \vbox{\makepostlude}%
831 \ifdim\sbarheight>\z@%
832 \nobreak\vskip2\p@\@plus\p@%
833 \nointerlineskip%
834 \hbox{\vrule\@height\sbarheight\@width\hsize}%
835 \fi%
836 }%
837 \SB@insongfalse%
838 \edef\SB@sgroup{\thesongnum}%
839 \global\SB@groupcnt\z@%
840 \SB@submitsong%
841 \ifnum\SB@grouplvl=\z@\let\SB@sgroup\@empty\fi%
842 \stepcounter{songnum}%
843 \else%
844 \ifSB@intersong\SB@erreor\SB@closeall%
845 \else\SB@erreot\fi%
846 \fi%
847 }

```

`\SB@setbaselineskip` Set the `\baselineskip` to an appropriate line height.

```

848 \newcommand\SB@setbaselineskip{%
849 \SB@dimen\fontsize\p@%
850 \baselineskip\SB@dimen\relax%
851 \ifchorded%
852 \setbox\SB@box\hbox{{\printchord{ABCDEFGH\shrp\flt/j7}}}%
853 \advance\baselineskip\ht\SB@box%
854 \advance\baselineskip2\p@%
855 \fi%
856 \ifslides%
857 \advance\baselineskip.2\SB@dimen\@plus.5\SB@dimen%
858 \@minus.2\SB@dimen%
859 \else%
860 \advance\baselineskip\z@\@plus.1\SB@dimen\relax%
861 \fi%
862 \advance\baselineskip\baselineadj%
863 }

```

`\SB@setversesep` Set the `\versesep` to an appropriate amount if has not already been explicitly set by the user.

```

864 \newcommand\SB@setversesep{%
865 \SB@dimen123456789sp%
866 \edef\SB@temp{\the\SB@dimen}%
867 \edef\SB@tempii{\the\versesep}%
868 \ifx\SB@temp\SB@tempii%
869 \begingroup%
870 \lyricfont%
871 \SB@dimen\fontsize\p@%
872 \ifchorded%
873 \setbox\SB@box\hbox{\printchord{ABCDEFG\shrp\flt/j7}}}%
874 \advance\SB@dimen\ht\SB@box%
875 \fi%
876 \ifslides%
877 \global\versesep1.2\SB@dimen\@plus.3\SB@dimen%
878 \@minus.3\SB@dimen%
879 \else%
880 \global\versesep.75\SB@dimen\@plus.25\SB@dimen%
881 \@minus.13\SB@dimen%
882 \fi%
883 \endgroup%
884 \fi%
885 }
```

`\makeprelude` Generate the material that begins each song. This macro is invoked at `\endsong` so that its code can access song info defined throughout the song.

```

886 \newcommand\makeprelude{%
887 \resettitles%
888 \ifslides%
889 \hbox to\hsize{%
890 \hfil\stitlefont\songtitle\hfil%
891 }%
892 \vskip5\p@%
893 \hbox to\hsize{%
894 \hfil%
895 \vbox{%
896 \divide\hsize\tw@\parskip\p@\relax%
897 \centering\small\extendprelude%
898 }%
899 \hfil%
900 }%
901 \else%
902 \ifdim\songnumwidth>\z@%
903 \setbox\SB@boxii\hbox{\SB@colorbox\snumbgcolor%
904 \hbox to\songnumwidth{%
905 \printsongnum{\thesongnum}\hfil%
906 }%
907 }%
908 \fi%
909 }
```

```

908 \fi%
909 \setbox\SB@box\vbox{%
910 \ifnum\SB@numcols>\z@\hsize\SB@colwidth\fi%
911 \ifdim\songnumwidth>\z@%
912 \advance\hsize-\wd\SB@boxii%
913 \advance\hsize-3\p@%
914 \fi%
915 \SB@raggedright\offinterlineskip\lineskip\p@%
916 {\stitlefont%
917 \songtitle\par%
918 \nexttitle%
919 \foreachtitle{(\songtitle)\par}}%
920 \ifdim\prevdepth=\z@\kern\p@\fi%
921 \parskip\p@\relax\tiny%
922 \extendprelude%
923 \kern\z@%
924 }%
925 \ifdim\songnumwidth>\z@%
926 \hbox{%
927 \ifdim\ht\SB@boxii>\ht\SB@box%
928 \box\SB@boxii%
929 \kern3\p@%
930 \vtop{\box\SB@box}%
931 \else%
932 \SB@colorbox\snubgcolor{\vbox to\ht\SB@box{{%
933 \hbox to\songnumwidth{%
934 \printsongnum{\thesongnum}\hfil%
935 }\vfil%
936 }}}%
937 \kern3\p@%
938 \box\SB@box%
939 \fi%
940 }%
941 \else%
942 \unvbox\SB@box%
943 \fi%
944 \fi%
945 }

```

`\makepostlude` Generate the material that ends each song.

```

946 \newcommand\makepostlude{%
947 \SB@raggedright\baselineskip\z@skip\parskip\z@skip\parindent\z@%
948 \tiny\extendpostlude%
949 }

```

`\showauthors` Display the author line in the prelude.

```

950 \newcommand\showauthors{%
951 \setbox\SB@box\hbox{\bfseries\sffcode‘.\@m\songauthors}%
952 \ifdim\wd\SB@box>\z@\unhbox\SB@box\par\fi%
953 }

```

`\showrefs` Display the scripture references in the prelude.

```

954 \newcommand\showrefs{%
955 \setbox\SB@box\hbox{\slshape\songrefs\vphantom,}%
956 \ifdim\wd\SB@box>\z@\unhbox\SB@box\par\fi%
957 }

```

`\SB@next` Several macros use `\futurelet` to look ahead in the input stream, and then take various actions depending on what is seen. In these macros, `\SB@next` is assigned the token seen, `\SB@dothis` is assigned the action to be taken on this loop iteration, and `\SB@donext` is assigned the action to be taken to continue (or terminate) the loop.

```

958 \newcommand\SB@next{}
959 \newcommand\SB@donext{}
960 \newcommand\SB@dothis{}

```

`\SB@nextname` Sometimes when scanning ahead we `\stringify` the name of the next token. When that happens, the name is stored in this macro for safekeeping.

```

961 \newcommand\SB@nextname{}

```

`\SB@appendsp` Append an explicit space token (catcode 10) to a token register. This is a useful macro to have around because inlining this code directly into a larger macro is harder than it seems: If you write the following code but with an explicit control sequence instead of `#1`, then the space immediately following the name will get stripped by the `TEX` parser. But invoking the following macro with a control sequence as an argument works fine, because in that case the explicit space has already been tokenized when this macro was first defined and won't be stripped as it is expanded.

```

962 \newcommand\SB@appendsp[1]{#1\expandafter{\the#1_}}

```

`\SB@parsetitles` Parse a list of song titles. This just involves removing leading and trailing spaces from around each title in the `\\`-separated list.

```

963 \newcommand\SB@parsetitles[1]{%
964 \begingroup%
965 \global\SB@titlelist{\\}%
966 \SB@toks{}%
967 \let\\ \SB@titlesep%
968 \let\SB@dothis\SB@pthead%
969 \SB@ptstart#1\SB@endparse%
970 \endgroup%
971 }

```

`\SB@ptstart` The iterator of the title parser loop just scans the next token.

```

972 \newcommand\SB@ptstart{\futurelet\SB@next\SB@dothis}

```

`\SB@pthead` While processing tokens at the head of a title, we skip over all spaces until we reach a non-space token.

```

973 \newcommand\SB@pthead{%
974 \ifcat\noexpand\SB@next\noexpand\@sptoken%
975 \expandafter\SB@ptsp%
976 \else%
977 \SB@toks{}%
978 \let\SB@dothis\SB@ptmain%
979 \expandafter\SB@ptmain%
980 \fi%
981 }

```

`\SB@ptmain` Once we've reached a non-space token in the title, we consume the remainder of the title as-is, except that space tokens should be trimmed from the end of each title.

```

982 \newcommand\SB@ptmain{%
983 \ifcat\noexpand\SB@next\noexpand\@sptoken%
984 \let\SB@donext\SB@ptsp%
985 \else\ifcat\noexpand\SB@next\noexpand\bgroup%
986 \let\SB@donext\SB@ptbg%
987 \else\ifx\SB@next\SB@endparse%
988 \global\SB@titlelist\expandafter\the\SB@titlelist\\}%
989 \let\SB@donext\@gobble%
990 \else%
991 \ifx\SB@next\\%
992 \SB@toks{}%
993 \let\SB@dothis\SB@pthead%
994 \fi%
995 \let\SB@donext\SB@ptstep%
996 \fi\fi\fi%
997 \SB@donext%
998 }

```

`\SB@ptstep` Consume a non-space, non-left-brace token and add it to the current song title. If any spaces preceded it, add those too.

```

999 \newcommand\SB@ptstep[1]{%
1000 \global\SB@titlelist\expandafter\expandafter\expandafter{%
1001 \expandafter\the\expandafter\SB@titlelist\the\SB@toks#1}%
1002 \SB@toks{}%
1003 \SB@ptstart%
1004 }

```

`\SB@ptbg` The next title token is a left-brace. It should be balanced, so consume the entire group and add it (along with its surrounding braces) as-is to the current title.

```

1005 \newcommand\SB@ptbg[1]{\SB@ptstep{{#1}}}

```



`\SB@ptsp` The next title token is a space. We won't know whether to include it in the title until we see what follows it. Strings of spaces followed by the `\` title-delimiter token, or that conclude a title argument, should be stripped. So rather than add the space token to the title, we remember it in a token register for possible later inclusion.

```

1006 \newcommand\SB@ptsp{
1007 \SB@appendsp\SB@toks%
1008 \afterassignment\SB@ptstart%
1009 \let\SB@next= }

```

`\SB@titlesep` While parsing song titles, we temporarily assign `\` a non-trivial top-level expansion (`\SB@titlesep`) in order to distinguish it from other macros.

```

1010 \newcommand\SB@titlesep{\SB@titlesep}

```

`\SB@endparse` The `\SB@endparse` token marks the end of a token sequence being parsed. If parsing works as intended, the macro should never be expanded, so produce an error if it is.

```

1011 \newcommand\SB@endparse{%
1012 \SB@Error{Title parsing failed}{This error should not occur.}%
1013 }

```

`\SB@testdigit` The following decides whether a token or `\let`-defined control sequence is a digit

`\SB@@testdigit` and sets conditional `\ifSB@test` accordingly.

```

1014 \newcommand\SB@testdigit[1]{%
1015 \SB@testfalse%
1016 \ifcat1\noexpand#1\SB@@testdigit#1\fi%
1017 }
1018 \newcommand\SB@@testdigit[1]{%
1019 \ifx0#1\SB@testtrue\else%
1020 \ifx1#1\SB@testtrue\else%
1021 \ifx2#1\SB@testtrue\else%
1022 \ifx3#1\SB@testtrue\else%
1023 \ifx4#1\SB@testtrue\else%
1024 \ifx5#1\SB@testtrue\else%
1025 \ifx6#1\SB@testtrue\else%
1026 \ifx7#1\SB@testtrue\else%
1027 \ifx8#1\SB@testtrue\else%
1028 \ifx9#1\SB@testtrue%
1029 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi%
1030 }

```

`\SB@parsesrefs` Assign the `\songrefs` macro a processed version of a scripture reference in which the following adjustments have been made: (1) Spaces not preceded by a comma or semicolon are made non-breaking. For example, 2 John 1:1 and Song of Solomon 1:1 become 2~John~1:1 and Song~of~Solomon~1:1, respectively. (2) Spaces between a semicolon and a book name are lengthened to en-spaces. (3) Single hyphens are lengthened to en-dashes (–). (4) Non-breaking, thin spaces are appended to commas not followed by a space. For example John 3:16,17

becomes `John~3:16,\nobreak\thinspace17`. (5) Everything within an explicit group is left unchanged, allowing the user to suppress all of the above as desired.

To achieve this, we must change all commas, hyphens, and spaces in the scripture reference into active characters. Unfortunately, the catcodes of everything in the text were set back when the full keyval list was digested as an argument to `\beginsong`, so we must unset and reset the catcodes. One obvious solution is to use `\scantokens` from  $\varepsilon$ -TeX to do this, but that doesn't allow us to suppress the re-catcoding process within groups, and we'd like to avoid introducing features that require  $\varepsilon$ -TeX anyway for compatibility reasons. Therefore, we build the following small scanner instead.

The scanner walks through the text token by token, replacing each important token by its active equivalent. No character codes are modified during this process and no tokens are inserted because some of these tokens might end up being arguments to multi-byte unicode character macros rather than being expanded directly. The `inputenc` package only cares about the character codes, not the category codes, so modifying only the category codes should be safe.

```

1031 \newcommand\SB@parsesrefs[1]{%
1032 \begingroup%
1033 \SB@toks{\begingroup\SB@sractives}%
1034 \SB@prloop#1\SB@endparse%
1035 \xdef\songrefs{\the\SB@toks\endgroup}%
1036 \endgroup%
1037 }
```

`\SB@prloop` The main loop of the scripture reference scanner identifies each space, hyphen, `\SB@prstep` and comma for special treatment.

```

\SB@@prstep 1038 \newcommand\SB@prloop{\futurelet\SB@next\SB@prstep}
1039 \newcommand\SB@prstep{%
1040 \ifcat\noexpand\SB@next A%
1041 \expandafter\SB@prcpy%
1042 \else%
1043 \expandafter\SB@@prstep%
1044 \fi%
1045 }
1046 \newcommand\SB@@prstep{%
1047 \ifcat\noexpand\SB@next\noexpand\@sptoken%
1048 \let\SB@donext\SB@prspace%
1049 \else\ifx\SB@next-%
1050 \let\SB@donext\SB@prhyphen%
1051 \else\ifx\SB@next,%
1052 \let\SB@donext\SB@prcomma%
1053 \else\ifx\SB@next\SB@endparse
1054 \let\SB@donext\@gobble%
1055 \else\ifcat\noexpand\SB@next\bgroup%
1056 \let\SB@donext\SB@prgr%
1057 \else%
1058 \let\SB@donext\SB@prcpy%
1059 \fi\fi\fi\fi\fi\fi%
```

```

1060 \SB@donext%
1061 }

\SB@prcpy Anything that isn't one of the special tokens above, and anything in a group, is
\SB@prgr copied without modification.
1062 \newcommand\SB@prcpy[1]{\SB@toks\expandafter{\the\SB@toks#1}\SB@prloop}
1063 \newcommand\SB@prgr[1]{\SB@toks\expandafter{\the\SB@toks{#1}}\SB@prloop}

\SB@prcomma Commas and hyphens are replaced with active equivalents.
\SB@prhyphen 1064 \newcommand\SB@prcomma[1]{%
1065 {\catcode'\active
1066 \gdef\SB@prcomma#1{\SB@toks\expandafter{\the\SB@toks,}\SB@prloop}}
1067 \newcommand\SB@prhyphen[1]{%
1068 {\catcode'\active
1069 \gdef\SB@prhyphen#1{\SB@toks\expandafter{\the\SB@toks-}\SB@prloop}}

\SB@prspace Spaces are made active as well, but doing so requires some specialized code since
\SB@@prspace they cannot be consumed as implicit macro arguments.
1070 \newcommand\SB@prspace[1]{%
1071 {\obeyspaces
1072 \gdef\SB@prspace{\SB@toks\expandafter{\the\SB@toks_\}\SB@@prspace}}
1073 \newcommand\SB@@prspace{\afterassignment\SB@prloop\let\SB@temp= }

\SB@sractives Assign macro definitions to active commas, hyphens, spaces, and returns when the
token list generated by \SB@parsesrefs is used to typeset a scripture reference
list.
1074 \newcommand\SB@sractives{%
1075 {\catcode'\active\catcode'\active\obeyspaces%
1076 \gdef\SB@sractives{%
1077 \let,\SB@srcomma\let-\SB@srhyphen\let_\SB@srspace%
1078 \SB@srspacing}%
1079 }

\SB@srspacing The space factors of semicolons and commas are what the active spaces within a
scripture reference text use to decide what came before. The following sets them
to their default values in case they have been changed, but sets all other space
factors to 1000.
1080 \newcommand\SB@srspacing{%
1081 \nonfrenchspacing\sfcode'\;=1500\sfcode'\;=1250\relax%
1082 }

\SB@srcomma Commas not already followed by whitespace are appended with a thin, non-
\SB@@srcomma breaking space.
1083 \newcommand\SB@srcomma{\futurelet\SB@next\SB@@srcomma}
1084 \newcommand\SB@@srcomma{%
1085 \ifx\SB@next\SB@srspace\else%
1086 \nobreak\thinspace%
1087 \fi%
1088 }

```

`\SB@srhyphen` Hyphens that are not already part of a ligature (an en- or em-dash) become en-  
`\SB@@srhyphen` dashes.

```

\SB@srdash 1089 \newcommand\SB@srhyphen{\futurelet\SB@next\SB@@srhyphen}
\SB@@srdash 1090 \newcommand\SB@@srhyphen{%
1091 \ifx\SB@next\SB@srhyphen\expandafter\SB@srdash\else--\fi%
1092 }
1093 \newcommand\SB@srdash[1]{\futurelet\SB@next\SB@@srdash}
1094 \newcommand\SB@@srdash{%
1095 \ifx\SB@next\SB@srhyphen---\expandafter\@gobble\else--\fi%
1096 }

```

`\SB@srspace` To compress consecutive whitespace, we ignore spaces immediately followed by more  
`\SB@@srspace` whitespace. Spaces not preceded by a semicolon or comma become non-breaking. Most spaces following a semicolon become en-spaces with favorable breakpoints, but a special case arises for spaces between a semicolon and a digit (see `\SB@srcso` below).

```

1097 \newcommand\SB@srspace{\futurelet\SB@next\SB@@srspace}
1098 \newcommand\SB@@srspace{%
1099 \let\SB@donext\relax%
1100 \ifx\SB@next\SB@srspace\else%
1101 \ifnum\spacefactor>\@m%
1102 \ifnum\spacefactor>1499 %
1103 \ifcat\noexpand\SB@next0%
1104 \let\SB@donext\SB@srcso%
1105 \else%
1106 \penalty-5\enskip%
1107 \fi%
1108 \else%
1109 \space%
1110 \fi%
1111 \else%
1112 \nobreak\space%
1113 \fi%
1114 \fi%
1115 \SB@donext%
1116 }

```

`\SB@srcso` A space between a semicolon and a digit could be within a list of verse references  
`\SB@@srcso` for a common book (e.g., Job 1:1; 2:2); or it could separate the previous book from a new book whose name starts with a number (e.g., Job 1:1; 1 John 1:1). In the former case, we should just use a regular space; but in the latter case we should be using an en-space with a favorable breakpoint. To distinguish between the two, we peek ahead at the next two tokens. If the second one is a space, assume the latter; otherwise assume the former.

```

1117 \newcommand\SB@srcso[1]{\futurelet\SB@temp\SB@@srcso}
1118 \newcommand\SB@@srcso{%
1119 \ifx\SB@temp\SB@srspace%
1120 \penalty-5\enskip%

```

```

1121 \else%
1122 \space%
1123 \fi%
1124 \SB@next%
1125 }

```

## 16.6 Verses and Choruses

The following programming typesets song contents, including verses, choruses, and textual notes.

```

\ifSB@stanza The following conditional remembers if we've seen any stanzas yet in the current
song.
1126 \newif\ifSB@stanza

\SB@stanzabreak End this song stanza and start a new one.
1127 \newcommand\SB@stanzabreak{%
1128 \ifhmode\par\fi%
1129 \ifSB@stanza%
1130 \SB@breakpoint{%
1131 \ifSB@inverse%
1132 \ifSB@prevverse\vvpentalty\else\cvpentalty\fi%
1133 \else%
1134 \ifSB@prevverse\vcpentalty\else\ccpentalty\fi%
1135 \fi%
1136 }%
1137 \vskip\versesep%
1138 \fi%
1139 }

\SB@breakpoint Insert a valid breakpoint into the vertical list comprising a song.
1140 \newcommand\SB@breakpoint[1]{%
1141 \begingroup%
1142 \ifnum#1<\@M%
1143 \SB@skip\colbotglue\relax%
1144 \SB@skip-\SB@skip%
1145 \else%
1146 \SB@skip\z@skip%
1147 \fi%
1148 \advance\SB@skip\lastskip%
1149 \unskip%
1150 \nobreak%
1151 \ifnum#1<\@M%
1152 \vskip\colbotglue\relax%
1153 \penalty#1%
1154 \fi%
1155 \vskip\SB@skip%
1156 \endgroup%
1157 }

```

`\SB@putbox` Unbox a vbox and follow it by vertical glue if its depth is unusually shallow. This ensures that verses and choruses will look equally spaced even if one of them has a final line with no letters that dangle below the baseline.

```

1158 \newcommand\SB@putbox[2]{%
1159 \begingroup%
1160 \SB@dimen\dp#2%
1161 #1#2%
1162 \setbox\SB@box\hbox{\lyricfont p}%
1163 \ifdim\SB@dimen<\dp\SB@box%
1164 \advance\SB@dimen-\dp\SB@box%
1165 \vskip-\SB@dimen%
1166 \fi%
1167 \setbox\SB@box\box\voidb@x%
1168 \endgroup%
1169 }
```

`\SB@obeylines` Within verses and choruses we would like to use `\obeylines` so that each *<return>* in the source file ends a paragraph without having to say `\par` explicitly. The L<sup>A</sup>T<sub>E</sub>X base code establishes the convention that short-term changes to `\par` will restore `\par` by setting it equal to `\@par`. Long-term (i.e., environment-long) changes to `\par` should therefore redefine `\@par` to restore the desired long-term definition. The following code starts a long-term redefinition of `\par` adhering to these conventions, and extends that definition to *<return>* as well.

```

1170 \newcommand\SB@obeylines{%
1171 \let\par\SB@par%
1172 \obeylines%
1173 \let\@par\SB@@par%
1174 }
```

`\SB@par` The following replacement definition of `\par` constructs paragraphs in which page-breaks are disallowed, since no wrapped line in a song should span a page- or column-break. It then inserts an interlinepenalty after the paragraph so that such penalties will appear between consecutive lines in each verse. (Note: The `\endgraf` macro must not be uttered within a local group since this prevents parameters like `\hangindent` from being reset at the conclusion of each paragraph.)

```

1175 \newcommand\SB@par{%
1176 \ifhmode%
1177 \SB@cnt\interlinepenalty%
1178 \interlinepenalty\@M%
1179 \endgraf%
1180 \interlinepenalty\SB@cnt%
1181 \ifSB@inchorus%
1182 \ifdim\cbarwidth>\z@\nobreak\else\SB@ilpenalty\fi%
1183 \else%
1184 \SB@ilpenalty%
1185 \fi%
1186 \fi%
1187 }
```

`\SB@ilpenalty` By default, breaking a vertical list between paragraphs incurs a penalty of zero. Thus, we only insert an explicit penalty between lines if `\interlinepenalty` is non-zero. This avoids cluttering the vertical list with superfluous zero penalties.

```

1188 \newcommand\SB@ilpenalty{%
1189 \ifnum\interlinepenalty=z@\else%
1190 \penalty\interlinepenalty%
1191 \fi%
1192 }

```

`\SB@@par` This replacement definition of `\@par` restores the `\SB@par` definition of `\par` and then ends the paragraph.

```

1193 \newcommand\SB@@par{\let\par\SB@par\par}

```

`\SB@parindent` Reserve a length to remember the current `\parindent`.

```

1194 \newdimen\SB@parindent

```

`\SB@everypar` Reserve a control sequence to hold short-term changes to `\everypar`.

```

1195 \newcommand\SB@everypar{}

```

`\SB@raggedright` Perform `\raggedright` except don't nuke the `\parindent`.

```

1196 \newcommand\SB@raggedright{%
1197 \SB@parindent\parindent%
1198 \raggedright%
1199 \parindent\SB@parindent%
1200 }

```

`\vnumbered` The following conditional remembers whether this verse is being numbered or not (i.e., it distinguishes between `\beginverse` and `\beginverse*`).

```

1201 \newif\ifvnumbered

```

`\ifSB@prevverse` Reserve a conditional to remember if the previous block in this song was a verse.

```

1202 \newif\ifSB@prevverse

```

Before replacing the little-used `verse` environment with a new one, issue a warning if the current definition of `\verse` is not the L<sup>A</sup>T<sub>E</sub>X-default one. This may indicate a package clash.

```

1203 \CheckCommand\verse{%
1204 \let\\@centercr%
1205 \list{}{%
1206 \itemsep\z@%
1207 \itemindent-1.5em%
1208 \listparindent\itemindent%
1209 \rightmargin\leftmargin%
1210 \advance\leftmargin1.5em%
1211 }%
1212 \item\relax%
1213 }

```

**verse** Begin a new verse. This can be done by beginning a **verse** environment or by using **verse\*** the **\beginverse** macro. The latter must check for a trailing star to determine if **\beginverse** this verse should be numbered. We use **\@ifstar** to scan ahead for the star, but this needs to be done carefully because while scanning we might encounter tokens that should be assigned different catcodes once the verse really begins. Thus, we temporarily invoke **\SB@loadactives** for the duration of **\@ifstar** so that everything gets the right catcode.

```

1214 \renewenvironment{verse}
1215 {\vnumberedfalse\SB@beginverse}
1216 {\SB@endverse}
1217 \newenvironment{verse*}
1218 {\vnumberedtrue\SB@beginverse}
1219 {\SB@endverse}
1220 \newcommand\beginverse{%
1221 \begingroup%
1222 \SB@loadactives%
1223 \@ifstar{\endgroup\vnumberedfalse\SB@beginverse}%
1224 {\endgroup\vnumberedtrue\SB@beginverse}%
1225 }
```

**\SB@beginverse** Start the body of a verse. We begin by inserting a mark if **\repchoruses** is active and this verse was preceded by a numbered verse (making this an eligible place to insert a chorus later).

Verse numbering is implemented using **\everypar** so that if there is any vertical material between the **\beginverse** and the first line of the verse, that material will come before the verse number. Intervening horizontal material (e.g., **\textnote**) can temporarily clear **\everypar** to defer the verse number until later.

```

1226 \newcommand\SB@beginverse{%
1227 \ifSB@insong%
1228 \ifSB@inverse\SB@errbv\endverse\fi%
1229 \ifSB@inchorus\SB@errbvc\endchorus\fi%
1230 \else%
1231 \SB@errbvt\beginsong{Unknown Song}%
1232 \fi%
1233 \ifrepchorus\ifvoid\SB@chorusbox\else%
1234 \SB@gotchorustrue%
1235 \ifSB@prevverse\ifvnumbered%
1236 \marks\SB@cmarkclass{\SB@cmark}%
1237 \fi\fi%
1238 \fi\fi%
1239 \SB@inversetrue%
1240 \def\SB@closeall{\endverse\endsong}%
1241 \SB@stanzabreak%
1242 \versemark\nobreak%
1243 \global\SB@stanzatrue%
1244 \SB@ifempty\SB@cr{\memorize{\replay[]}}%
1245 \setbox\SB@box\vbox\bgroup\begingroup%
1246 \ifvnumbered%
```



```

1247 \protected@edef\@currentlabel{\p@versenum\theversenum}%
1248 \def\SB@everypar{%
1249 \setbox\SB@box\hbox{%
1250 \printversenum{\theversenum}%
1251 }}%
1252 \ifdim\wd\SB@box<\versenumwidth%
1253 \setbox\SB@box%
1254 \hbox to\versenumwidth{\unhbox\SB@box\hfil}%
1255 \fi%
1256 \ifchorded\vrule\@height\baselineskip\@width\z@\@depth\z@\fi%
1257 {\placeversenum\SB@box}%
1258 \gdef\SB@everypar{%
1259 }%
1260 \else%
1261 \def\SB@everypar{%
1262 \ifchorded\vrule\@height\baselineskip\@width\z@\@depth\z@\fi%
1263 \gdef\SB@everypar{%
1264 }%
1265 \fi%
1266 \everypar{\SB@everypar\everypar}}%
1267 \versefont\versejustify%
1268 \SB@loadactives%
1269 \SB@obeylines%
1270 \penalty12345 %
1271 \everyverse\relax%
1272 }

```

**\SB@endverse** End a verse. This involves unboxing the verse material with **\SB@putbox**, which corrects for last lines that are unusually shallow.

```

1273 \newcommand\SB@endverse{%
1274 \ifSB@insong%
1275 \ifSB@inverse%
1276 \unpenalty%
1277 \endgroup\egroup%
1278 \SB@putbox\unvbox\SB@box%
1279 \SB@inversefalse%
1280 \def\SB@closeall{\endsong}%
1281 \ifvnumbered\stepcounter{versenum}\fi%
1282 \SB@prevversettrue%
1283 \else\ifSB@inchorus\SB@errevc\endchorus%
1284 \else\SB@errevo\fi\fi%
1285 \else%
1286 \SB@errevt%
1287 \fi%
1288 }

```

**\ifSB@chorustop** When a chorus is broken in to several pieces by column-breaks (via **\brk**), the following conditional remembers whether the current piece is the topmost one for this chorus.

```

1289 \newif\ifSB@chorustop

```

`\SB@chorusbox` When `\repchoruses` is used, the first sequence of consecutive choruses is remembered in the following box register.

```
1290 \newbox\S@chorusbox
```

`\ifSB@gotchorus` The following conditional remembers whether we've completed storing the first block of consecutive choruses.

```
1291 \newif\ifSB@gotchorus
```

`\SB@cmarkclass` The `\repeatchoruses` feature requires the use of two extended mark classes provided by  $\epsilon$ -TeX. We use the `\newmarks` macro to allocate these classes, if it's available. If `\newmarks` doesn't exist, then that means the user has an  $\epsilon$ -TeX compatible version of L<sup>A</sup>T<sub>E</sub>X, but no `etex` style file to go with it; we just have to pick two mark classes and hope that nobody else is using them.

`\SB@nocmarkclass`

```
1292 \ifSB@etex
1293 \@ifundefined{newmarks}{
1294 \@ifundefined{newmark}{
1295 \mathchardef\S@cmarkclass83
1296 \mathchardef\S@nocmarkclass84
1297 }{
1298 \newmark\S@cmarkclass
1299 \newmark\S@nocmarkclass
1300 }
1301 }{
1302 \newmarks\S@cmarkclass
1303 \newmarks\S@nocmarkclass
1304 }
1305 \fi
```

`\SB@cmark` To determine where choruses should be inserted when `\repchoruses` is active, three kinds of marks are inserted into song boxes: `\SB@cmark` is used to mark places where a chorus might be inserted between verses, and `\SB@lastcmark` marks a place where a chorus might be inserted after the last verse of the song. Both marks are  $\epsilon$ -TeX marks of class `\SB@cmarkclass`, to avoid disrupting the use of standard TeX marks. Each time a chorus is automatically inserted, `\SB@nocmark` is inserted with mark class `\SB@nocmarkclass` just above it (and at the top of each additional page it spans). This inhibits future chorus inserts until the already-inserted chorus has been fully committed to the output file. Otherwise some choruses could get auto-inserted multiple times at the same spot, possibly even leading to an infinite loop!

`\SB@lastcmark`

`\SB@nocmark`

```
1306 \newcommand\S@cmark{}
1307 \def\S@cmark{\SB@cmark}
1308 \newcommand\S@lastcmark{}
1309 \def\S@lastcmark{\SB@lastcmark}
1310 \newcommand\S@nocmark{}
1311 \def\S@nocmark{\SB@nocmark}
```

**chorus** Start a new chorus. If `\repchoruses` is active and this is part of the first set of consecutive choruses in the song, then include it and its preceding vertical material in the `\SB@chorusbox` for possible later duplication elsewhere.

```

1312 \newenvironment{chorus}{\beginchorus}{\SB@endchorus}
1313 \newcommand\beginchorus{%
1314 \ifSB@insong
1315 \ifSB@inverse\SB@errbcv\endverse\fi%
1316 \ifSB@inchorus\SB@errbcc\endchorus\fi%
1317 \else%
1318 \SB@errbct\beginsong{Unknown Song}%
1319 \fi%
1320 \SB@inchorustrue%
1321 \def\SB@closeall{\endchorus\endsong}%
1322 \SB@chorustoptrue%
1323 \vnumberedfalse%
1324 \SB@stanzabreak%
1325 \chorusmark%
1326 \ifrepchorus%
1327 \ifSB@gotchorus\else\ifSB@prevverse\else%
1328 \global\setbox\SB@chorusbox\vbox{%
1329 \unvbox\SB@chorusbox%
1330 \SB@stanzabreak%
1331 \chorusmark%
1332 }%
1333 \fi\fi%
1334 \fi%
1335 \global\SB@stanzatrue%
1336 \replay[]%
1337 \SB@@beginchorus%
1338 \everychorus\relax%
1339 }
```

**\SB@@beginchorus** Begin the body of a chorus, or continue the body of a chorus after `\brk` has paused it to insert a valid breakpoint. We insert an empty class-`\SB@cmarkclass` mark here so that this chorus will not be duplicated elsewhere on the same page(s) where it initially appears.

```

1340 \newcommand\SB@@beginchorus{%
1341 \ifrepchorus\marks\SB@cmarkclass{}\fi%
1342 \setbox\SB@box\vbox\bgroup\begingroup%
1343 \ifchorded%
1344 \def\SB@everypar{%
1345 \vrule\@height\baselineskip\@width\z@\@depth\z@%
1346 }\gdef\SB@everypar{}%
1347 }%
1348 \everypar{\SB@everypar\everypar{}}%
1349 \fi%
1350 \chorusfont\chorusjustify%
1351 \SB@loadactives%
1352 \SB@obeylines%
```

```

1353 \penalty12345 %
1354 }

```

**\SB@endchorus** End a chorus. This involves creating the vertical line to the left of the chorus and then unboxing the chorus material that was previously accumulated.

```

1355 \newcommand\SB@endchorus{%
1356 \ifSB@insong%
1357 \ifSB@inchorus%
1358 \unpenalty%
1359 \endgroup\egroup%
1360 \SB@inchorusfalse%
1361 \def\SB@closeall{\endsong}%
1362 \setbox\SB@box\vbox{%
1363 \SB@chorusbar\SB@box%
1364 \SB@putbox\unvbox\SB@box%
1365 }
1366 \ifrepchorus\ifSB@gotchorus\else%
1367 \global\setbox\SB@chorusbox\vbox{%
1368 \unvbox\SB@chorusbox%
1369 \unvcopy\SB@box%
1370 }%
1371 \fi\fi%
1372 \unvbox\SB@box%
1373 \SB@prevversefalse%
1374 \else\ifSB@inverse\SB@errecev\endverse%
1375 \else\SB@erreco\fi\fi%
1376 \else%
1377 \SB@errect%
1378 \fi%
1379 }

```

**\SB@cbarshift** Increase \leftskip to accommodate the chorus bar, if any.

```

1380 \newcommand\SB@cbarshift{%
1381 \ifSB@inchorus\ifdim\cbarwidth>\z@%
1382 \advance\leftskip\cbarwidth%
1383 \advance\leftskip5\p@\relax%
1384 \fi\fi%
1385 }

```

**\SB@chorusbar** Create the vertical bar that goes to the left of a chorus. Rather than boxing up the chorus in order to put the bar to the left, the bar is introduced as leaders directly into the vertical list of the main song box. This allows it to stretch and shrink when a column is typeset by the page-builder.

```

1386 \newcommand\SB@chorusbar[1]{%
1387 \ifdim\cbarwidth>\z@%
1388 \SB@dimen\ht#1%
1389 \SB@dimenii\dp#1%
1390 \advance\SB@dimen%
1391 \ifSB@chorustop\ifchorded\else2\fi\fi\SB@dimenii%

```

```

1392 \SB@skip\SB@dimen\relax%
1393 \SB@computess\SB@skip1\@plus#1%
1394 \SB@computess\SB@skip{-1}\@minus#1%
1395 \nointerlineskip\null\nobreak%
1396 \leaders\vrule\@width\cbarwidth\vskip\SB@skip%
1397 \ifSB@chorustop\ifchorded\else%
1398 \advance\SB@skip-\SB@dimenii%
1399 \fi\fi%
1400 \nobreak\vskip-\SB@skip%
1401 \fi%
1402 }

```

`\SB@computess` This computes the stretchability or shrinkability of a vbox and stores the result in the skip register given by  $\langle arg1 \rangle$ . If  $\langle arg2 \rangle = 1$  and  $\langle arg3 \rangle$  is “plus”, then the stretchability of box  $\langle arg4 \rangle$  is added to the plus component of  $\langle arg1 \rangle$ . If  $\langle arg2 \rangle = -1$  and  $\langle arg3 \rangle$  is “minus”, then the shrinkability of the box is added to the minus component of  $\langle arg1 \rangle$ . If the stretchability or shrinkability is infinite, then we guess 1fil for that component.

```

1403 \newcommand\SB@computess[4]{%
1404 \beginingroup%
1405 \vbadness\@M\vfuZZ\maxdimen%
1406 \SB@dimen4096\p@%
1407 \setbox\SB@box\vbox spread#2\SB@dimen{\unvcopy#4}%
1408 \ifnum\badness=\z@%
1409 \global\advance#1\z@#31fil\relax%
1410 \else%
1411 \SB@dimenii\SB@dimen%
1412 \loop%
1413 \SB@dimenii.5\SB@dimenii%
1414 \ifnum\badness<100 %
1415 \advance\SB@dimen\SB@dimenii%
1416 \else
1417 \advance\SB@dimen-\SB@dimenii%
1418 \fi%
1419 \setbox\SB@box\vbox spread#2\SB@dimen{\unvcopy#4}%
1420 \ifnum\badness=100 \SB@dimenii\z@\fi%
1421 \ifdim\SB@dimenii>.1\p@\repeat%
1422 \ifdim\SB@dimen<.1\p@\SB@dimen\z@\fi%
1423 \global\advance#1\z@#3\SB@dimen\relax%
1424 \fi%
1425 \endgroup%
1426 }

```

`\brk` Placing `\brk` within a line in a verse or chorus tells T<sub>E</sub>X to break the line at that point (if it needs to be broken at all).

Placing `\brk` on a line by itself within a chorus stops the chorus (and its vertical bar), inserts a valid breakpoint, and then restarts the chorus with no intervening space so that if the breakpoint isn’t used, there will be no visible effect. Placing it on a line by itself within a verse just inserts a breakpoint.

Placing `\brk` between songs forces a column- or page-break, but only if generating a non-partial list of songs. When generating a partial list, `\brk` between songs is ignored.

```

1427 \newcommand\brk{%
1428 \ifSB@insong%
1429 \ifhmode\penalty-5 \else%
1430 \unpenalty%
1431 \ifSB@inchorus%
1432 \ifdim\cbarwidth=\z@%
1433 \ifrepchorus\marks\SB@cmarkclass{}\fi%
1434 \SB@breakpoint\brkpenalty%
1435 \else%
1436 \endgroup\egroup%
1437 \ifrepchorus\ifSB@gotchorus\else%
1438 \global\setbox\SB@chorusbox\vbox{%
1439 \unvbox\SB@chorusbox%
1440 \SB@chorusbar\SB@box%
1441 \unvcopy\SB@box%
1442 \SB@breakpoint\brkpenalty%
1443 }%
1444 \fi\fi%
1445 \SB@chorusbar\SB@box%
1446 \unvbox\SB@box%
1447 \SB@breakpoint\brkpenalty%
1448 \SB@chorustopfalse%
1449 \SB@@beginchorus%
1450 \fi%
1451 \else%
1452 \SB@breakpoint\brkpenalty%
1453 \fi%
1454 \fi%
1455 \else%
1456 \ifpartiallist\else\SB@nextcol\@ne\colbotglue\fi%
1457 \fi%
1458 }

```

`\SB@boxup` Typeset a shaded box containing a textual note to singers or musicians. We first try typesetting the note on a single line. If it's too big, then we try again in paragraph mode with full justification.

```

1459 \newcommand\SB@boxup[1]{%
1460 \setbox\SB@box\hbox{\notefont#1}}%
1461 \SB@dimen\wd\SB@box%
1462 \advance\SB@dimen6\p@%
1463 \advance\SB@dimen\leftskip%
1464 \advance\SB@dimen\rightskip%
1465 \ifdim\SB@dimen>\hsize%
1466 \vbox{%
1467 \advance\hsize-6\p@%
1468 \advance\hsize-\leftskip%

```

```

1469 \advance\hsize-\rightskip%
1470 \notejustify%
1471 \unhbox\SB@box\par%
1472 \kern\z@%
1473 }}%
1474 \else%
1475 \vbox{\box\SB@box\kern\z@}%
1476 \fi%
1477 }

```

**\textnote** Create a textual note for singers and musicians. If the note begins a verse or chorus, it should not be preceded by any spacing. Verses and choruses begin with the sentinel penalty 12345, so we check `\lastpenalty` to identify this case. When typesetting the note, we must be sure to temporarily clear `\everypar` to inhibit any verse numbering that might be pending. We also readjust the `\baselineskip` as if we weren't doing chords, since no chords go above a textual note.

```

1478 \newcommand\textnote[1]{%
1479 \ifhmode\par\fi%
1480 \ifnum\lastpenalty=12345\else%
1481 \ifSB@inverse%
1482 \vskip2\p@\relax%
1483 \else\ifSB@inchorus%
1484 \vskip2\p@\relax%
1485 \else\ifSB@stanza%
1486 \nobreak\vskip\versesep%
1487 \fi\fi\fi%
1488 \fi%
1489 \begingroup%
1490 \everypar{}%
1491 \ifchorded\chordedfalse\SB@setbaselineskip\chordedtrue\fi%
1492 \placenote{\SB@colorbox\notebgcolor{\SB@boxup{#1}}}%
1493 \endgroup%
1494 \nobreak%
1495 \ifSB@inverse%
1496 \vskip2\p@\relax%
1497 \else\ifSB@inchorus%
1498 \vskip2\p@\relax%
1499 \else\ifSB@stanza\else%
1500 \nobreak\vskip\versesep%
1501 \fi\fi\fi%
1502 }

```

**\musicnote** Create a textual note for musicians.

```

1503 \newcommand\musicnote[1]{\ifchorded\textnote{#1}\fi}

```

**\echo** Typeset an echo part in the lyrics. Echo parts will be oblique and parenthesized.

**\SB@echo** We toggle between oblique and upright shapes like `\emph`, but we use `\slshape` instead of `\itshape` because it tends to look nicer with the larger fonts used in slides mode.

The `\echo` macro must be able to accept chords in its argument. This complicates the implementation because chord macros should change catcodes, but if we grab `\echo`'s argument in the usual way then all the catcodes will be set before the chord macros have a chance to change them. This would disallow chord name abbreviations like `#` and `&` within `\echo` parts.

If we're using  $\varepsilon$ -TeX then the solution is easy: we use `\scantokens` to re-scan the argument and thereby re-assign the catcodes. (One subtlety: Whenever L<sup>A</sup>T<sub>E</sub>X consumes an argument to a macro, it changes `#` to `##` so that when the argument text is substituted into the body of the macro, the replacement text will not contain unsubstituted parameters (such as `#1`). If `\scantokens` is used on the replacement text and the scanned tokens assign a new catcode to `#`, that causes `#`'s to be doubled in the *output*, which was not the intent. To avoid this problem, we use `\@sanitize` before consuming the argument to `\echo`, which sets the catcodes of most special tokens (including `#`) to 12, so that L<sup>A</sup>T<sub>E</sub>X will not recognize any of them as parameters and will therefore not double any of them.)

```

1504 \ifSB@etex
1505 \newcommand\echo{\begingroup\@sanitize\SB@echo}
1506 \newcommand\SB@echo[1]{%
1507 \endgroup%
1508 \begingroup%
1509 \ifdim\fontdimen\@ne\font>\z@\upshape\else\slshape\fi%
1510 \endlinechar\m@ne%
1511 \scantokens{(#1)}%
1512 \endgroup%
1513 }
1514 \else

```

If we're not using  $\varepsilon$ -TeX, we must do something more complicated. We set up the appropriate font within a local group and finish with `\hbox` so that the argument to `\echo` is treated as the body of the box. Control is reacquired after the box using `\aftergroup`, whereupon we unbox the box and insert the closing parenthesis. This almost works except that if the last thing in an echo part is a long chord name atop a short lyric, the closing parenthesis will float out away from the lyric instead of being sucked under the chord. I can find no solution to this problem, so to avoid it users must find a version of L<sup>A</sup>T<sub>E</sub>X that is  $\varepsilon$ -TeX compatible.

```

1515 \newcommand\echo{%
1516 \begingroup%
1517 \ifdim\fontdimen\@ne\font>\z@\upshape\else\slshape\fi%
1518 \afterassignment\SB@echo%
1519 \setbox\SB@box\hbox%
1520 }
1521 \newcommand\SB@echo{\aftergroup\SB@@echo{}}
1522 \newcommand\SB@@echo{\unhbox\SB@box}\endgroup}
1523 \fi

```

`\rep` Place `\rep{<n>}` at the end of a line to indicate that it should be sung `<n>` times.

```

1524 \newcommand\rep[1]{%
1525 (\raise.25ex\hbox{%

```



```

1526 \fontencoding{OMS}\fontfamily{cmsy}\selectfont\char\tw@%
1527 }#1)%
1528 }

```

## 16.7 Scripture Quotations

The macros in this section typeset scripture quotations and other between-songs environments.

**songgroup** A **songgroup** environment associates all enclosed environments with the enclosed song. When generating a partial list, all the enclosed environments are contributed if and only if the enclosed song is contributed.

```

1529 \newenvironment{songgroup}{%
1530 \ifnum\SB@grouplvl=\z@%
1531 \edef\SB@sgroup{\thesongnum}%
1532 \global\SB@groupcnt\m@ne%
1533 \fi%
1534 \advance\SB@grouplvl\@ne%
1535 }{%
1536 \advance\SB@grouplvl\m@ne%
1537 \ifnum\SB@grouplvl=\z@\let\SB@sgroup\@empty\fi%
1538 }

```

**\SB@grouplvl** Count the **songgroup** environment nesting depth.

```

1539 \newcount\SB@grouplvl

```

**intersong** An **intersong** block contributes vertical material to a column between the songs of a songs section. It is subject to the same column-breaking algorithm as real songs, but receives none of the other formatting applied to songs.

```

1540 \newenvironment{intersong}{%
1541 \ifSB@insong\SB@errbro\SB@closeall\fi%
1542 \ifSB@intersong\SB@errbrr\SB@closeall\fi%
1543 \setbox\SB@chorusbox\box\voidb@x%
1544 \SB@intersongtrue%
1545 \def\SB@closeall{\end{intersong}}%
1546 \setbox\SB@songbox\vbox\bgroup\begingroup%
1547 \ifnum\SB@numcols>\z@\hsize\SB@colwidth\fi%
1548 \ifdim\sbarheight>\z@%
1549 \hrule\@height\sbarheight\@width\hsize%
1550 \nobreak%
1551 \fi%
1552 }{%
1553 \ifSB@intersong
1554 \ifdim\sbarheight>\z@%
1555 \ifhmode\par\fi%
1556 \SB@skip\lastskip%
1557 \unskip\nobreak\vskip\SB@skip%
1558 \hbox{\vrule\@height\sbarheight\@width\hsize}%
1559 \fi%

```

```

1560 \endgroup\egroup%
1561 \ifSB@omitscrip%
1562 \setbox\SB@songbox\box\voidb@x%
1563 \else%
1564 \SB@submitsong%
1565 \fi%
1566 \SB@intersongfalse%
1567 \else%
1568 \ifSB@insong\SB@errero\SB@closeall\else\SB@errert\fi%
1569 \fi%
1570 }

```

The starred form contributes page-spanning vertical material directly to the top of the nearest fresh page.

```

1571 \newenvironment{intersong*}{%
1572 \ifSB@insong\SB@errbro\SB@closeall\fi%
1573 \ifSB@intersong\SB@errbrr\SB@closeall\fi%
1574 \setbox\SB@chorusbox\box\voidb@x%
1575 \SB@intersongtrue%
1576 \def\SB@closeall{\end{intersong*}}%
1577 \setbox\SB@songbox\vbox\bgroup\begin{group}%
1578 }{%
1579 \ifSB@intersong%
1580 \endgroup\egroup%
1581 \ifSB@omitscrip%
1582 \setbox\SB@songbox\box\voidb@x%
1583 \else%
1584 \def\SB@stype{\SB@styppage}%
1585 \SB@submitsong%
1586 \def\SB@stype{\SB@stypcol}%
1587 \fi%
1588 \SB@intersongfalse%
1589 \else%
1590 \ifSB@insong\SB@errero\SB@closeall\else\SB@errert\fi%
1591 \fi%
1592 }

```

**\SB@srbox** The following box register holds the citation information that is to be typeset at the end of a scripture quotation.

```

1593 \newbox\SB@srbox

```

**scripture** Begin a scripture quotation. We first store the reference in a box for later use, and  
**\beginscripture** then set up a suitable environment for the quotation. Quotations cannot typically be reworded if line-breaking fails, so we set **\emergencystretch** to a relatively high value at the outset.

```

1594 \newenvironment{scripture}{\beginscripture}{\SB@endscripture}
1595 \newcommand\beginscripture[1]{%
1596 \begin{intersong}%
1597 \SB@parsesrefs{#1}%
1598 \setbox\SB@srbox\hbox{\printsrcite\songrefs}}%

```

```

1599 \def\SB@closeall{\endscripture}%
1600 \nobreak\vskip5\p@%
1601 \SB@parindent\parindent\parindent\z@%
1602 \parskip\z@skip\parfillskip\@flushglue%
1603 \leftskip\SB@parindent\rightskip\SB@parindent\relax%
1604 \scripturefont%
1605 \baselineskip\fontsize\p@\@plus\p@\relax%
1606 \advance\baselineskip\p@\relax%
1607 \emergencystretch.3em%
1608 }

```

`\SB@endscripture` End a scripture quotation.

```

1609 \newcommand\SB@endscripture{%
1610 \ifSB@intersong
1611 \scitehere%
1612 \ifhmode\par\fi%
1613 \vskip-3\p@%
1614 \end{intersong}%
1615 \fi%
1616 }

```

`\scitehere` Usually the scripture citation should just come at the `\endscripture` line, but at times the user might want to invoke this macro explicitly at a more suitable point. A good example is when something near the end of the scripture quotation drops T<sub>E</sub>X into vertical mode. In such cases, it is often better to issue the citation before leaving horizontal mode.

In any case, this macro should work decently whether in horizontal or vertical mode. In horizontal mode life is easy: we just append the reference to the current horizontal list using the classic code from p. 106 of The T<sub>E</sub>Xbook. However, if we're now in vertical mode, the problem is a little harder. We do the best we can by using `\lastbox` to remove the last line, then adding the reference and re-typesetting it. This isn't as good as the horizontal mode solution because T<sub>E</sub>X only gets to reevaluate the last line instead of the whole paragraph, but usually the results are passable.

```

1617 \newcommand\scitehere{%
1618 \ifSB@intersong%
1619 \ifvoid\SB@srbox\else%
1620 \ifvmode%
1621 \setbox\SB@box\lastbox%
1622 \nointerlineskip\noindent\hskip-\leftskip%
1623 \unhbox\SB@box\unskip%
1624 \fi%
1625 \unskip\nobreak\hfil\penalty50\hskip.8em\null\nobreak\hfil%
1626 \box\SB@srbox\kern-\SB@parindent%
1627 {\parfillskip\z@\finalhyphendemerits2000\par}%
1628 \fi%
1629 \else%
1630 \SB@errscrip\scitehere%

```

```

1631 \fi%
1632 }

\Acolon Typeset a line of poetry in a scripture quotation.
\Bcolon 1633 \newcommand\Acolon{\SB@colon2\Acolon}
1634 \newcommand\Bcolon{\SB@colon1\Bcolon}

\SB@colon Begin a group of temporary definitions that will end at the next <return>. The
<return> will end the paragraph and close the local scope.
1635 \newcommand\SB@colon[2]{%
1636 \ifSB@intersong\else%
1637 \SB@errscrip#2%
1638 \beginscripture{Unknown}%
1639 \fi%
1640 \ifhmode\par\fi%
1641 \begingroup%
1642 \rightskip\SB@parindent\@plus4em%
1643 \advance\leftskip2\SB@parindent%
1644 \advance\parindent-#1\SB@parindent%
1645 \def\par{\endgraf\endgroup}%
1646 \obeylines%
1647 }

\strophe Insert blank space indicative of a strophe division in a scripture quotation.
1648 \newcommand\strophe{%
1649 \ifSB@intersong\else%
1650 \SB@errscrip\strophe\beginscripture{Unknown}%
1651 \fi%
1652 \vskip.9ex\@plus.45ex\@minus.68ex\relax%
1653 }

\scripindent Create an indented sub-block within a scripture quotation.
\scripoutdent 1654 \newcommand\SB@scripindent[2]{%
\SB@scripindent 1655 \ifSB@intersong\else%
1656 \SB@errscrip#2\beginscripture{Unknown}%
1657 \fi%
1658 \ifhmode\par\fi%
1659 \advance\leftskip#1\SB@parindent\relax%
1660 }
1661 \newcommand\scripindent{\SB@scripindent1\scripindent}
1662 \newcommand\scripoutdent{\SB@scripindent-\scripoutdent}

\shiftdblquotes The Zaph Chancery font used by default to typeset scripture quotations seems to
\SB@ldqlleft have some kerning problems with double-quote ligatures. The \shiftdblquotes
\SB@ldqrigh macro allows one to modify the spacing around all double-quotes until the current
\SB@rdqlleft group ends.
\SB@rdqrigh 1663 \newcommand\SB@quotesactive{%
\SB@scanlq 1664 \catcode'\active%
\SB@scanrq 1665 \catcode'\active%
\SB@dqlq
\SB@dorq

```

```

1666 }
1667 \newcommand\shiftdblquotes[4]{%
1668 \newcommand\SB@ldqleft{}
1669 \newcommand\SB@ldqright{}
1670 \newcommand\SB@rdqleft{}
1671 \newcommand\SB@rdqright{}
1672 \newcommand\SB@scanlq{}
1673 \newcommand\SB@scanrq{}
1674 \newcommand\SB@dolq{}
1675 \newcommand\SB@dorq{}
1676 {
1677 \SB@quotesactive
1678 \gdef\shiftdblquotes#1#2#3#4{%
1679 \def\SB@ldqleft{\kern#1}%
1680 \def\SB@ldqright{\kern#2}%
1681 \def\SB@rdqleft{\kern#3}%
1682 \def\SB@rdqright{\kern#4}%
1683 \SB@quotesactive%
1684 \def'\{\futurelet\SB@next\SB@scanlq}%
1685 \def'\{\futurelet\SB@next\SB@scanrq}%
1686 }
1687 \gdef\SB@scanlq{%
1688 \ifx\SB@next'%
1689 \expandafter\SB@dolq%
1690 \else%
1691 \expandafter\lq%
1692 \fi%
1693 }
1694 \gdef\SB@scanrq{%
1695 \ifx\SB@next'%
1696 \expandafter\SB@dorq%
1697 \else%
1698 \expandafter\rq%
1699 \fi%
1700 }
1701 \gdef\SB@dolq'{%
1702 \ifvmode\leavevmode\else\/\fi%
1703 \vadjust{}%
1704 \SB@ldqleft\lq\lq\SB@ldqright%
1705 \vadjust{}%
1706 }
1707 \gdef\SB@dorq'{%
1708 \ifvmode\leavevmode\else\/\fi%
1709 \vadjust{}%
1710 \SB@rdqleft\rq\rq\SB@rdqright%
1711 \vadjust{}%
1712 }
1713 }

```

## 16.8 Transposition

The macros that transpose chords are contained in this section.

`\SB@transposefactor` This counter identifies the requested number of halfsteps by which chords are to be transposed (from  $-11$  to  $+11$ ).

```
1714 \newcount\SB@transposefactor
```

`\ifSB@convertnotes` Even when transposition is not requested, the transposition logic can be used to automatically convert note names to another form. The following conditional turns that feature on or off.

```
1715 \newif\ifSB@convertnotes
```

`\notenameA` Reserve a control sequence for each note of the diatonic scale. These will be used  
`\notenameB` to identify which token sequences the input file uses to denote the seven scale  
`\notenameC` degrees. Their eventual definitions *must* consist entirely of uppercase letters, and  
`\notenameD` they must be assigned using `\def`, but that comes later.

```
\notenameE 1716 \newcommand\notenameA{}
```

```
\notenameF 1717 \newcommand\notenameB{}
```

```
\notenameG 1718 \newcommand\notenameC{}
```

```
1719 \newcommand\notenameD{}
```

```
1720 \newcommand\notenameE{}
```

```
1721 \newcommand\notenameF{}
```

```
1722 \newcommand\notenameG{}
```

`\printnoteA` These control sequences are what the transposition logic actually outputs to denote  
`\printnoteB` each scale degree. They can include any *L*<sup>A</sup>T<sub>E</sub>X code that is legal in horizontal mode.

```
\printnoteC 1723 \newcommand\printnoteA{}
```

```
\printnoteD 1724 \newcommand\printnoteB{}
```

```
\printnoteE 1725 \newcommand\printnoteC{}
```

```
\printnoteF 1726 \newcommand\printnoteD{}
```

```
\printnoteG 1727 \newcommand\printnoteE{}
```

```
1728 \newcommand\printnoteF{}
```

```
1729 \newcommand\printnoteG{}
```

`\notenamesin` Set the note names used by the input file.

```
1730 \newcommand\notenamesin[7]{%
```

```
1731 \def\notenameA{#1}%
```

```
1732 \def\notenameB{#2}%
```

```
1733 \def\notenameC{#3}%
```

```
1734 \def\notenameD{#4}%
```

```
1735 \def\notenameE{#5}%
```

```
1736 \def\notenameF{#6}%
```

```
1737 \def\notenameG{#7}%
```

```
1738 \SB@convertnotestrue%
```

```
1739 }
```

**\notenamesout** Set the note names that are output by the transposition logic.

```

1740 \newcommand\notenamesout[7]{%
1741 \def\printnoteA{#1}%
1742 \def\printnoteB{#2}%
1743 \def\printnoteC{#3}%
1744 \def\printnoteD{#4}%
1745 \def\printnoteE{#5}%
1746 \def\printnoteF{#6}%
1747 \def\printnoteG{#7}%
1748 \SB@convertnotestrue%
1749 }

```

**\notenames** Set an identical input name and output name for each scale degree.

```

1750 \newcommand\notenames[7]{%
1751 \notenamesin{#1}{#2}{#3}{#4}{#5}{#6}{#7}%
1752 \notenamesout{#1}{#2}{#3}{#4}{#5}{#6}{#7}%
1753 \SB@convertnotesfalse%
1754 }

```

**\alphascale** Predefine scales for alphabetic names and solfedge names, and set alphabetic scales

**\solfedge** to be the default.

```

1755 \newcommand\alphascale{\notenames ABCDEFG}
1756 \newcommand\solfedge{\notenames{LA}{SI}{DO}{RE}{MI}{FA}{SOL}}
1757 \alphascale

```

**\ifSB@prefshrps** When a transposed chord falls on a black key, the code must choose which enharmonically equivalent name to give the new chord. (For example, should C transposed by +1 be named C# or Db?) A heuristic is used to guess which name is most appropriate. The following conditional records whether the current key signature is sharpened or flattened according to this heuristic guess.

```

1758 \newif\ifSB@prefshrps

```

**\ifSB@needkey** The first chord seen is usually the best indicator of the key of the song. (Even when the first chord isn't the tonic, it will often be the dominant or subdominant, which usually has the same kind of accidental in its key signatures as the actual key.) This conditional remembers if the current chord is the first one seen in the song, and should therefore be used to guess the key of the song.

```

1759 \newif\ifSB@needkey

```

**\transpose** The **\transpose** macro sets the transposition adjustment factor and informs the transposition logic that the next chord seen will be the first one in the new key.

```

1760 \newcommand\transpose[1]{%
1761 \advance\SB@transposefactor by#1\relax%
1762 \SB@cnt\SB@transposefactor%
1763 \divide\SB@cnt12 %
1764 \multiply\SB@cnt12 %
1765 \advance\SB@transposefactor-\SB@cnt%
1766 \SB@needkeytrue%
1767 }

```

`\capo` Specifying a `\capo` normally just causes a textual note to musicians to be typeset, but if the `\transposecapos` option is active, it activates transposition of the chords.

```

1768 \newcommand\capo[1]{%
1769 \iftranscapos\transpose{#1}\else\musicnote{capo #1}\fi%
1770 }

```

`\prefersharp` One of these macros is called after the first chord has been seen to register that we're transposing to a key with a sharped or flatted key signature.

```

1771 \newcommand\prefersharp{\SB@prefshrpstrue\SB@needkeyfalse}
1772 \newcommand\preferflat{\SB@prefshrpfalse\SB@needkeyfalse}

```

`\transposehere` If automatic transposition has been requested, yield the given chord transposed by the requested amount. Otherwise return the given chord verbatim.

```

1773 \newcommand\transposehere[1]{%
1774 \ifnum\SB@transposefactor=\z@%
1775 \ifSB@convertnotes%
1776 \SB@dottranspose{#1}%
1777 \the\SB@toks%
1778 \else%
1779 #1%
1780 \fi%
1781 \else%
1782 \ifSB@convertnotes%
1783 {\SB@transposefactor\z@%
1784 \SB@dottranspose{#1}%
1785 \xdef\SB@tempv{\the\SB@toks}}%
1786 \else%
1787 \def\SB@tempv{#1}%
1788 \fi%
1789 \SB@dottranspose{#1}%
1790 \expandafter\trchordformat\expandafter{\SB@tempv}{\the\SB@toks}%
1791 \fi%
1792 }

```

`\notrans` Suppress chord transposition without suppressing note name conversion. When a `\notrans{<text>}` macro appears within text undergoing transposition, the `\notrans` macro and the group will be preserved verbatim by the transposition parser. When it is then expanded after parsing, we must therefore re-invoke the transposition logic on the argument, but in an environment where the transposition factor has been temporarily set to zero. This causes note name conversion to occur without actually transposing.

```

1793 \newcommand\notrans[1]{%
1794 \begingroup%
1795 \SB@transposefactor\z@%
1796 \transposehere{#1}%
1797 \endgroup%
1798 }

```



`\SB@dotranspose` Parse the argument to a chord macro, yielding the transposed equivalent in the `\SB@toks` token register.

```
1799 \newcommand\SB@dotranspose[1]{%
1800 \SB@toks{}}%
1801 \let\SB@dothis\SB@trmain%
1802 \SB@trscan#1\SB@trend%
1803 }
```

`\trchordformat` By default, transposing means replacing old chords with new chords in the new key. However, sometimes the user may want to typeset something more sophisticated, like old chords followed by new chords in parentheses so that musicians who use capos and those who don't can play from the same piece of music. Such typesetting is possible by redefining the following macro to something like `#1 (#2)` instead of `#2`.

```
1804 \newcommand\trchordformat[2]{#2}
```

`\SB@trscan` This is the entrypoint to the code that scans over the list of tokens comprising a chord and transposes note names as it goes. Start by peeking ahead at the next symbol without consuming it.

```
1805 \newcommand\SB@trscan{\futurelet\SB@next\SB@dothis}
```

`\SB@trmain` Test to see if the token was a begin-brace, end-brace, or space. These tokens require special treatment because they cannot be accepted as implicit arguments to macros.

```
1806 \newcommand\SB@trmain{%
1807 \ifx\SB@next\bgroup%
1808 \let\SB@donext\SB@trgroup%
1809 \else\ifx\SB@next\egroup%
1810 \SB@toks\expandafter{\the\SB@toks\egroup}%
1811 \let\SB@donext\SB@trskip%
1812 \else\ifcat\noexpand\SB@next\noexpand\@sptoken%
1813 \SB@appendsp\SB@toks%
1814 \let\SB@donext\SB@trskip%
1815 \else%
1816 \let\SB@donext\SB@trstep%
1817 \fi\fi\fi%
1818 \SB@donext%
1819 }
```

`\SB@trgroup` A begin-group brace lies next in the input stream. Consume the entire group as an argument to this macro, and append it, including the begin- and end-group tokens, to the list of tokens processed so far. No transposition takes place within a group; they are copied verbatim because they probably contain macro code.

```
1820 \newcommand\SB@trgroup[1]{%
1821 \SB@toks\expandafter{\the\SB@toks{#1}}%
1822 \SB@trscan%
1823 }
```

`\SB@trspace` A space or end-brace lies next in the input stream. It has already been added to the token list, so skip over it.

```
1824 \newcommand\SB@trskip{%
1825 \afterassignment\SB@trscan%
1826 \let\SB@next= }
```

`\SB@trstep` A non-grouping token lies next in the input stream. Consume it as an argument to this macro, and then test it to see if it's a note letter or some other recognized item. If so, process it; otherwise just append it to the token list and continue scanning.

```
1827 \newcommand\SB@trstep[1]{%
1828 \let\SB@donext\SB@trscan%
1829 \ifcat\noexpand\SB@next A%
1830 \ifnum\uccode'#1='#1%
1831 \def\SB@temp{#1}%
1832 \let\SB@dothis\SB@trnote%
1833 \else%
1834 \SB@toks\expandafter{\the\SB@toks#1}%
1835 \fi%
1836 \else\ifx\SB@next\SB@trend
1837 \let\SB@donext\relax%
1838 \else%
1839 \SB@toks\expandafter{\the\SB@toks#1}%
1840 \fi\fi%
1841 \SB@donext%
1842 }
```

`\SB@trnote` We're in the midst of processing a sequence of uppercase letters that might comprise a note name. Check to see if the next token is an accidental (sharp or flat), or yet another letter.

```
1843 \newcommand\SB@trnote{%
1844 \ifcat\noexpand\SB@next A%
1845 \let\SB@donext\SB@trnotestep%
1846 \else\ifnum\SB@transposefactor=\z0%
1847 \SB@cnt\z0%
1848 \let\SB@donext\SB@trtrans%
1849 \else\ifx\SB@next\flt%
1850 \SB@cnt\m0ne%
1851 \let\SB@donext\SB@tracc%
1852 \else\ifx\SB@next\shrp%
1853 \SB@cnt\@ne%
1854 \let\SB@donext\SB@tracc%
1855 \else%
1856 \SB@cnt\z0%
1857 \let\SB@donext\SB@trtrans%
1858 \fi\fi\fi\fi%
1859 \SB@donext%
1860 }
```

`\SB@trnotestep` The next token is a letter. Consume it and test to see if it is an uppercase letter. If so, add it to the note name being assembled; otherwise reinsert it into the input stream and jump directly to the transposition logic.

```

1861 \newcommand\SB@trnotestep[1]{%
1862 \ifnum\uccode'#1='#1%
1863 \expandafter\def\expandafter\SB@temp\expandafter{\SB@temp#1}%
1864 \expandafter\SB@trscan%
1865 \else%
1866 \SB@cnt\z@%
1867 \expandafter\SB@trtrans\expandafter#1%
1868 \fi%
1869 }
```

`\SB@tracc` We've encountered an accidental (sharp or flat) immediately following a note name. Peek ahead at the next token without consuming it, and then jump to the transposition logic. This is done because the transposition logic might need to infer the key signature of the song, and if the next token is an m (for minor), then that information can help.

```

1870 \newcommand\SB@tracc[1]{\futurelet\SB@next\SB@trtrans}
```

`\SB@trtrans` We've assembled a sequence of capital letters (in `\SB@temp`) that might comprise a note name to be transposed. If the letters were followed by a `\shrp` then `\SB@cnt` is 1; if they were followed by a `\flt` then it is -1; otherwise it is 0. If the assembled letters turn out to not match any valid note name, then do nothing and return to scanning. Otherwise compute a new transposed name.

```

1871 \newcommand\SB@trtrans{%
1872 \advance\SB@cnt%
1873 \ifx\SB@temp\notenameA\z@%
1874 \else\ifx\SB@temp\notenameB\tw@%
1875 \else\ifx\SB@temp\notenameC\thr@@%
1876 \else\ifx\SB@temp\notenameD5 %
1877 \else\ifx\SB@temp\notenameE7 %
1878 \else\ifx\SB@temp\notenameF8 %
1879 \else\ifx\SB@temp\notenameG10 %
1880 \else-99 \fi\fi\fi\fi\fi\fi\fi%
1881 \ifnum\SB@cnt<\m@ne%
1882 \SB@toks\expandafter\expandafter\expandafter{%
1883 \expandafter\the\expandafter\SB@toks\SB@temp}%
1884 \else%
1885 \advance\SB@cnt\SB@transposefactor%
1886 \ifnum\SB@cnt<\z@\advance\SB@cnt12 \fi%
1887 \ifnum\SB@cnt>11 \advance\SB@cnt-12 \fi%
1888 \ifSB@needkey\ifnum\SB@transposefactor=\z@\else\SB@setkeysig\fi\fi%
1889 \edef\SB@temp{%
1890 \the\SB@toks%
1891 \ifSB@prefshrps%
1892 \ifcase\SB@cnt\printnoteA\or\printnoteA\noexpand\shrp\or%
1893 \printnoteB\or\printnoteC\or\printnoteC\noexpand\shrp\or%
```

```

1894 \printnoteD\or\printnoteD\noexpand\shrp\or\printnoteE\or%
1895 \printnoteF\or\printnoteF\noexpand\shrp\or\printnoteG\or%
1896 \printnoteG\noexpand\shrp\fi%
1897 \else%
1898 \ifcase\SB@cnt\printnoteA\or\printnoteB\noexpand\flt\or%
1899 \printnoteB\or\printnoteC\or\printnoteD\noexpand\flt\or%
1900 \printnoteD\or\printnoteE\noexpand\flt\or\printnoteE\or%
1901 \printnoteF\or\printnoteG\noexpand\flt\or\printnoteG\or%
1902 \printnoteA\noexpand\flt\fi%
1903 \fi}%
1904 \SB@toks\expandafter{\SB@temp}%
1905 \fi%
1906 \let\SB@dothis\SB@trmain%
1907 \SB@trscan%
1908 }

```

**\SB@setkeysig** If this is the first chord of the song, assume that this is the tonic of the key, and select whether to use a sharped or flatted key signature for the rest of the song based on that. Even if this isn't the tonic, it's probably the dominant or sub-dominant, which almost always has a number of sharps or flats similar to the tonic. If the bottom note of the chord turns out to be a black key, we choose the enharmonic equivalent that is closest to C on the circle of fifths (i.e., the one that has fewest sharps or flats).

```

1909 \newcommand\SB@setkeysig{%
1910 \global\SB@needkeyfalse%
1911 \ifcase\SB@cnt%
1912 \global\SB@prefshrpstrue\or% A
1913 \global\SB@prefshrpsfalse\or% Bb
1914 \global\SB@prefshrpstrue\or% B
1915 \ifx\SB@next m% C
1916 \global\SB@prefshrpsfalse%
1917 \else%
1918 \global\SB@prefshrpstrue%
1919 \fi\or%
1920 \global\SB@prefshrpstrue\or% C#
1921 \ifx\SB@next m% D
1922 \global\SB@prefshrpsfalse%
1923 \else%
1924 \global\SB@prefshrpstrue%
1925 \fi\or%
1926 \global\SB@prefshrpsfalse\or% Eb
1927 \global\SB@prefshrpstrue\or% E
1928 \global\SB@prefshrpsfalse\or% F
1929 \global\SB@prefshrpstrue\or% F#
1930 \ifx\SB@next m% G
1931 \global\SB@prefshrpsfalse%
1932 \else%
1933 \global\SB@prefshrpstrue%
1934 \fi\or%

```

```

1935 \global\SB@prefshrpsfalse\else% Ab
1936 \global\SB@needkeytrue% non-chord
1937 \fi%
1938 }

```

**\SB@trend** The following macro marks the end of chord text to be processed. It should always be consumed and discarded by the chord-scanning logic above, so generate an error if it is ever expanded.

```

1939 \newcommand\SB@trend{%
1940 \SB@Error{Internal Error: Transposition failed}%
1941 {This error should not occur.}%
1942 }

```

## 16.9 Measure Bars

The following code handles the typesetting of measure bars.

**\SB@metertop** These macros remember the current numerator and denominator of the meter.

```

\SB@meterbot 1943 \newcommand\SB@metertop{}
1944 \newcommand\SB@meterbot{}

```

**\meter** Set the current meter without producing an actual measure bar yet.

```

1945 \newcommand\meter[2]{\gdef\SB@metertop{#1}\gdef\SB@meterbot{#2}}

```

**\SB@measuremark** Normally measure bar boxes should be as thin as possible so that they can be slipped into lyrics without making them hard to read. But when two measure bars appear consecutively, they need to be spaced apart more so that they look like two separate lines instead of one thick line. To achieve this, there needs to be a way to pull a vbox off the current list and determine whether or not it is a box that contains a measure bar. The solution is to insert a mark (**\SB@measuremark**) at the top of each measure bar vbox. We can then see if this measure bar immediately follows another measure bar by using **\vsplit** on **\lastbox**.

```

1946 \newcommand\SB@measuremark{SB@IsMeasure}

```

**\SB@makembar** Typeset a measure bar. If provided, *<arg1>* is the numerator and *<arg2>* is the denominator of the meter to be rendered above the bar. If those arguments are left blank, render a measure bar without a meter marking.

```

1947 \newcommand\SB@makembar[2]{%
1948 \ifSB@inverse\else%
1949 \ifSB@inchorus\else\SB@errmbar\fi%
1950 \fi%
1951 \ifhmode%
1952 \SB@skip\lastskip\unskip%
1953 \setbox\SB@box\lastbox%
1954 \copy\SB@box%
1955 \ifvbox\SB@box%
1956 \begingroup%
1957 \setbox\SB@boxii\copy\SB@box%

```

```

1958 \vbadness\@M\vfuzz\maxdimen%
1959 \setbox\SB@boxii%
1960 \vsplit\SB@boxii to\maxdimen%
1961 \endgroup%
1962 \long\edef\SB@temp{\splitfirstmark}%
1963 \ifx\SB@temp\SB@measuremark%
1964 \penalty100\hskip1em%
1965 \else%
1966 \penalty100\hskip\SB@skip%
1967 \fi%
1968 \else%
1969 \penalty100\hskip\SB@skip%
1970 \fi%
1971 \fi%
1972 \ifvmode\leavevmode\fi%
1973 \setbox\SB@box\hbox{\tiny\sffamily{#1}}%
1974 \setbox\SB@boxii\hbox{\tiny\sffamily{#2}}%
1975 \ifdim\wd\SB@box>\wd\SB@boxii%
1976 \SB@dimen\wd\SB@box\relax%
1977 \else%
1978 \SB@dimen\wd\SB@boxii\relax%
1979 \fi%
1980 \ifdim\SB@dimen<.5\p@%
1981 \SB@dimen.5\p@%
1982 \fi%
1983 \SB@dimenii\baselineskip%
1984 \advance\SB@dimenii-2\p@%
1985 \advance\SB@dimenii-\ht\SB@box%
1986 \advance\SB@dimenii-\dp\SB@box%
1987 \advance\SB@dimenii-\ht\SB@boxii%
1988 \advance\SB@dimenii-\dp\SB@boxii%
1989 \vbox{%
1990 \mark{\SB@measuremark}%
1991 \hbox to\SB@dimen{%
1992 \hfil%
1993 \box\SB@box%
1994 \hfil%
1995 }%
1996 \nointerlineskip%
1997 \hbox to\SB@dimen{%
1998 \hfil%
1999 \box\SB@boxii%
2000 \hfil%
2001 }%
2002 \nointerlineskip%
2003 \hbox to\SB@dimen{%
2004 \hfil%
2005 \vrule\@width.5\p@\@height\SB@dimenii%
2006 \hfil%
2007 }%

```

```

2008 }%
2009 }

\mbar The \mbar macro invokes \SB@mbar, which gets redefined by macros and options
 that turn measure bars on and off.
2010 \newcommand\mbar{\SB@mbar}

\measurebar Make a measure bar using the most recently defined meter. Then set the meter to
 nothing so that the next measure bar will not display any meter unless the meter
 changes.
2011 \newcommand\measurebar{%
2012 \mbar\SB@metertop\SB@meterbot%
2013 \meter{}{}}%
2014 }

\SB@repcolon Create the colon that preceeds or follows a repeat sign.
2015 \newcommand\SB@repcolon{%
2016 \usefont{OT1}{cmss}{m}{n}\selectfont%
2017 \ifchorded%
2018 \baselineskip.5\SB@dimen%
2019 \vbox{\hbox{:}\hbox{:}\kern.5\p}%
2020 \else%
2021 \raise.5\p\hbox{:}%
2022 \fi%
2023 }}

\lrep Create a begin-repeat sign.
2024 \newcommand\lrep{%
2025 \SB@dimen\baselineskip%
2026 \advance\SB@dimen-2\p%
2027 \vrule\@width1.5\p\@height\SB@dimen\@depth\p%
2028 \kern1.5\p%
2029 \vrule\@width.5\p\@height\SB@dimen\@depth\p%
2030 \SB@repcolon%
2031 }

\rrep Create an end-repeat sign.
2032 \newcommand\rrep{%
2033 \SB@dimen\baselineskip%
2034 \advance\SB@dimen-2\p%
2035 \SB@repcolon%
2036 \vrule\@width.5\p\@height\SB@dimen\@depth\p%
2037 \kern1.5\p%
2038 \vrule\@width1.5\p\@height\SB@dimen\@depth\p%
2039 }

```

## 16.10 Lyric Scanning

The obvious way to create a chord macro is as a normal macro with two arguments, one for the chord name and one for the lyrics to go under the chord—e.g. `\chord{<chordname>{<lyric>}}`. However, in practice such a macro is extremely cumbersome and difficult to use. The problem is that in order to use such a macro properly, the user must remember a bunch of complex style rules that govern what part of the lyric text needs to go in the `<lyric>` parameter and what part should be typed after the closing brace. To avoid separating a word from its trailing punctuation, the `<lyric>` parameter must often include punctuation but not certain special punctuation like hyphens, should include the rest of the word but not if there's another chord in the word, should omit measure bars but only if measure bars are being shown, etc. This is way too difficult for the average user.

To avoid this problem, we define chords using a one-argument macro (the argument is the chord name), but with no explicit argument for the lyric part. Instead, the macro scans ahead in the input stream, automatically determining what portion of the lyric text that follows should be sucked in as an implicit second argument. The following code does this look-ahead scanning.

```

\ifSB@wordends Chord macros must look ahead in the input stream to see if this chord is immedi-
\ifSB@brokenword ately followed by whitespace or the remainder of a word. If the latter, hyphenation
might need to be introduced. These macros keep track of the need for hyphenation,
if any.
2040 \newif\ifSB@wordends
2041 \newif\ifSB@brokenword

\SB@lyric Lyrics appearing after a chord are scanned into the following token list register.
2042 \newtoks\SB@lyric

\SB@numhyphs Hyphens appearing in lyrics require special treatment. The following counter coun-
ts the number of explicit hyphens ending the lyric syllable that follows the current
chord.
2043 \newcount\SB@numhyphs

\SB@lyricnohyp When a lyric syllable under a chord ends in exactly one hyphen, the following
token register is set to be the syllable without the hyphen.
2044 \newtoks\SB@lyricnohyp

\SB@lyricbox The following two boxes hold the part of the lyric text that is to be typeset under
\SB@chordbox the chord, and the chord text that is to be typeset above.
2045 \newbox\SB@lyricbox
2046 \newbox\SB@chordbox

\SB@chbstok When \MultiwordChords is active, the following reserved control sequence
remembers the first (space) token not yet included into the \SB@lyricbox box.
2047 \newcommand\SB@chbstok{}
```



`\SB@setchord` The following macro typesets its argument as a chord and stores the result in box `\SB@chordbox` for later placement into the document. The hat token ( $\hat{\phantom{x}}$ ) is redefined so that outside of math mode it suppresses chord memorization, but inside of math mode it retains its usual superscript meaning. If memorization is active, the chord's token sequence is stored in the current replay register. If `\SB@chordbox` is non-empty, the new chord is appended to it rather than replacing it. This allows consecutive chords not separated by whitespace to be typeset as a single chord sequence atop a common lyric.

```

2048 \newcommand\SB@setchord{}
2049 {
2050 \catcode'\active
2051 \catcode'\!7
2052 \gdef\SB@setchord#1{%
2053 \SB@gettabindtrue\SB@nohattrue%
2054 \setbox\SB@chordbox\hbox{%
2055 \unhbox\SB@chordbox%
2056 \begingroup%
2057 \ifSB@trackch%
2058 \def\SB@activehat{\ifmmode!\else\global\SB@nohatfalse\fi}%
2059 \else%
2060 \def\SB@activehat{%
2061 \ifmmode!\else\SB@lop\SB@ctail\SB@toks\the\SB@toks\fi%
2062 }%
2063 \fi%
2064 \let\SB@activehat%
2065 \printchord{%
2066 \ifSB@firstchord\else\kern.15em\fi%
2067 \vphantom/%
2068 \transposehere{#1}%
2069 \kern.2em%
2070 }%
2071 \endgroup%
2072 }%
2073 \SB@gettabindfalse%
2074 \ifSB@trackch\ifSB@nohat%
2075 \global\SB@creg\expandafter{\the\SB@creg#1\\}%
2076 \fi\fi%
2077 \let\SB@noreplay\@firstofone%
2078 }
2079 }
```

`\SB@outertest` The lyric-scanning code must preemptively determine if the next token is a macro declared `\outer` before it tries to accept that token as an argument. Otherwise  $\TeX$  will abort with a parsing error. Macros declared `\outer` are not allowed in arguments, so determining if a token is `\outer` is a delicate process. The following does so by consulting `\meaning`.

```

2080 \newcommand\SB@outertest{}
2081 \edef\SB@outertest#1{%
2082 \noexpand\SB@@outertest#1%
```

```

2083 \string\outer%
2084 \noexpand\SB@@outertest%
2085 }
2086 \newcommand\SB@@outertest{}
2087 \expandafter\def\expandafter\SB@@outertest%
2088 \expandafter#\expandafter1\string\outer#2\SB@@outertest{%
2089 \def\SB@temp{#2}%
2090 \ifx\SB@temp\@empty\else\SB@testtrue\fi%
2091 }

\SB@UTFtest To support UTF-8 encoded LATEX source files, we need to be able to identify
\SB@two multibyte characters during the lyric scanning process. Alas, the utf8.def file
\SB@three provides no clean way of identifying the macros it defines for this purpose. The
\SB@four best solution seems to be to look for any token named \UTFviii@...@octets in
\SB@UTFtester the top-level expansion of the macro.

2092 \newcommand\SB@UTFtest{}
2093 \edef\SB@UTFtest#1{%
2094 \noexpand\SB@UTFtester#1%
2095 \string\UTFviii@zero@octets%
2096 \noexpand\SB@UTFtester%
2097 }
2098 \begingroup
2099 \escapechar\m@ne
2100 \xdef\SB@two{\string\two}
2101 \xdef\SB@three{\string\three}
2102 \xdef\SB@four{\string\four}
2103 \xdef\SB@temp{\string\@octets}
2104 \endgroup
2105 \edef\SB@temp{##1\string\UTFviii@##2\SB@temp##3}
2106 \expandafter\def\expandafter\SB@UTFtester\SB@temp\SB@UTFtester{%
2107 \def\SB@temp{#2}%
2108 \ifx\SB@temp\SB@two%
2109 \SB@cnt\tw@%
2110 \else\ifx\SB@temp\SB@three%
2111 \SB@cnt\thr@@%
2112 \else\ifx\SB@temp\SB@four%
2113 \SB@cnt4 %
2114 \else%
2115 \SB@cnt\z@%
2116 \fi\fi\fi%
2117 }

\DeclareLyricChar When scanning the lyric text that follows a chord, it is necessary to distinguish
\DeclareNonLyric accents and other intra-word macros (which should be included in the under-
\DeclareNoHyphen chord lyric text) from other macros (which should be pushed out away from the
\SB@declare text). The following macros allow users to declare a token to be lyric-continuing
or lyric-ending.

2118 \newcommand\SB@declare[3]{%
2119 \afterassignment\iffalse\let\SB@next= #3\relax\fi%

```

```

2120 \expandafter\SB@UTFtest\expandafter{\meaning\SB@next}%
2121 \ifcase\SB@cnt%
2122 \ifcat\noexpand#3\relax%
2123 \SB@addNtest\SB@macrotests#1#2#3%
2124 \else\ifcat\noexpand#3.%
2125 \SB@addDtest\SB@othertests#1#2#3%
2126 \else\ifcat\noexpand#3A%
2127 \SB@addDtest\SB@lettertests#1#2#3%
2128 \else%
2129 \SB@addDtest\relax0#2#3%
2130 \fi\fi\fi%
2131 \or%
2132 \SB@addNtest\SB@macrotests#1#2#3%
2133 \else%
2134 \SB@addMtest\SB@multitests#1#2{#3}%
2135 \fi%
2136 }
2137 \newcommand\DeclareLyricChar{\SB@declare\SB@testtrue0}
2138 \newcommand\DeclareNonLyric{%
2139 \SB@declare\SB@testfalse\SB@testfalse%
2140 }
2141 \newcommand\DeclareNoHyphen{%
2142 \SB@declare\SB@testfalse\SB@testtrue%
2143 }

```

\SB@lettertests For speed, token tests introduced by \DeclareLyricChar and friends are broken out into separate macros based on category codes.

```

\SB@multitests 2144 \newcommand\SB@lettertests{}
\SB@othertests 2145 \newcommand\SB@macrotests{}
\SB@hyphtests 2146 \newcommand\SB@multitests{}
 2147 \newcommand\SB@othertests{}
 2148 \newcommand\SB@hyphtests{}

```

The following macros add tests to the test macros defined above. In each,  $\langle arg1 \rangle$  is the test macro to which the test should be added,  $\langle arg2 \rangle$  and  $\langle arg3 \rangle$  is the code to be executed at scanning-time and at hyphenation-time if the test succeeds (or “0” if no action is to be performed), and  $\langle arg4 \rangle$  is the token to which the currently scanned token should be compared to determine if it matches.

\SB@addtest Append the top-level expansion of  $\langle arg2 \rangle$  to the control sequence name given by  $\langle arg1 \rangle$ .

```

2149 \newcommand\SB@addtest[2]{%
2150 \expandafter\gdef\expandafter#1\expandafter{#1#2}%
2151 }

```

\SB@addDtest A definition-test: The test succeeds if the *definition* at test-time of the next lyric token matches the *definition at test-time* of the control sequence that was given to the \Declare macro.

```

2152 \newcommand\SB@addDtest[4]{%

```

```

2153 \ifx0#2\else\SB@addtest#1{\ifx\SB@next#4#2\fi}\fi%
2154 \ifx0#3\else\SB@addtest\SB@hyphtests{\ifx\SB@next#4#3\fi}\fi%
2155 }

```

**\SB@addNtest** A name-test: The test succeeds if the next token is a non-`\outer` macro or active character and its `\stringified` name matches the `\stringified` name of the control sequence that was given to the `\Declare` macro.

```

2156 \newcommand\SB@addNtest[4]{%
2157 \ifx0#2\else%
2158 \SB@addtest#1{%
2159 \edef\SB@temp{\string#4}\ifx\SB@temp\SB@nextname#2\fi%
2160 }%
2161 \fi%
2162 \ifx0#3\else%
2163 \SB@addtest\SB@hyphtests{%
2164 \edef\SB@temp{\string#4}\ifx\SB@temp\SB@nextname#3\fi%
2165 }%
2166 \fi%
2167 }

```

**\SB@addMtest** A multibyte-test: The test succeeds if the next lyric token is the beginning of a UTF-8 encoded multibyte character sequence that matches the multibyte sequence given to the `\Declare` macro.

```

2168 \newcommand\SB@addMtest[4]{%
2169 \ifx0#2\else%
2170 \SB@addtest#1{\def\SB@temp{#4}\ifx\SB@next\SB@temp#2\fi}%
2171 \fi%
2172 \ifx0#3\else\SB@addtest\SB@hyphtests{%
2173 \def\SB@temp{#4}\ifx\SB@next\SB@temp#3\fi}%
2174 \fi%
2175 }

```

The following code declares the common intra-word macros provided by  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  (as listed on p. 52 of *The  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ book*) to be lyric-continuing.

```

2176 \DeclareLyricChar\`
2177 \DeclareLyricChar\'
2178 \DeclareLyricChar\^
2179 \DeclareLyricChar\"
2180 \DeclareLyricChar\~
2181 \DeclareLyricChar\=
2182 \DeclareLyricChar\.
2183 \DeclareLyricChar\u
2184 \DeclareLyricChar\v
2185 \DeclareLyricChar\H
2186 \DeclareLyricChar\t
2187 \DeclareLyricChar\c
2188 \DeclareLyricChar\d
2189 \DeclareLyricChar\b
2190 \DeclareLyricChar\oe

```

```

2191 \DeclareLyricChar\OE
2192 \DeclareLyricChar\ae
2193 \DeclareLyricChar\AE
2194 \DeclareLyricChar\aa
2195 \DeclareLyricChar\AA
2196 \DeclareLyricChar\o
2197 \DeclareLyricChar\O
2198 \DeclareLyricChar\l
2199 \DeclareLyricChar\L
2200 \DeclareLyricChar\ss
2201 \DeclareLyricChar\i
2202 \DeclareLyricChar\j
2203 \DeclareLyricChar\/
2204 \DeclareLyricChar\-
2205 \DeclareLyricChar\discretionary

```

We declare `\par` to be lyric-ending without introducing hyphenation. The `\par` macro doesn't actually appear in most verses because we use `\obeylines`, but we include a check for it in case the user says `\par` explicitly somewhere.

```

2206 \DeclareNoHyphen\par

```

`\SB@bracket` This macro gets invoked by the `\[` macro whenever a chord begins. It gets redefined by code that turns chords on and off, so its initial definition doesn't matter.

```

2207 \newcommand\SB@bracket{}

```

`\SB@chord` Begin parsing a chord macro. While parsing the chord name argument, we set some special catcodes so that chord names can use `#` and `&` for sharps and flats.

```

2208 \newcommand\SB@chord{\SB@begincname\SB@@chord}

```

`\SB@begincname` While parsing a chord name, certain characters such as `#` and `&` are temporarily set active so that they can be used as abbreviations for sharps and flats. To accomplish this, `\SB@begincname` must always be invoked before any macro whose argument is a chord name, and `\SB@endcname` must be invoked at the start of the body of any macro whose argument is a chord name. To aid in debugging, we also temporarily set *return* characters and chord macros `\outer`. This will cause  $\TeX$  to halt with a runaway argument error on the correct source line if the user forgets to type a closing end-brace (a common typo). Colon characters are also set non-active to avoid a conflict between the Babel French package and the `\gtab` macro.

`\SB@endcname`

```

2209 \newcommand\SB@begincname{}
2210 {\catcode'\^M\active
2211 \gdef\SB@begincname{%
2212 \begingroup%
2213 \catcode'\#\active\catcode'\&\active%
2214 \catcode':12\relax%
2215 \catcode'\^M\active\SB@outer\def^^M{}%
2216 \SB@outer\def\{}%
2217 \chordlocals%
2218 }
2219 }

```

```

2220 \newcommand\SB@endcname{}
2221 \let\SB@endcname\endgroup

\SB@nbsp Non-breaking spaces (~) should be treated as spaces by the lyric-scanner code
that follows. Although ~ is usually an active character that creates a non-breaking
space, some packages (e.g., the Babel package) redefine it to produce accents, which
are typically not lyric-ending. To distinguish the real ~ from redefined ~, we need
to create a macro whose definition is the non-breaking space definition normally
assigned to ~.
2222 \newcommand\SB@nbsp{}
2223 \def\SB@nbsp{\nobreakspace{}}

\SB@firstchord The following conditional is true when the current chord is the first chord in a
sequence of one or more chord macros.
2224 \newif\ifSB@firstchord\SB@firstchordtrue

\SB@@chord Finish processing the chord name and then begin scanning the implicit lyric
argument that follows it. This is the main entrypoint to the lyric-scanner code.
2225 \newcommand*\SB@@chord{}
2226 \def\SB@@chord#1{%
2227 \SB@endcname%
2228 \ifSB@firstchord%
2229 \setbox\SB@lyricbox\hbox{\kern\SB@tabindent}%
2230 \global\SB@tabindent\z@%
2231 \SB@lyric{}%
2232 \SB@numhyps\z@%
2233 \SB@spcinit%
2234 \setbox\SB@chordbox\box\voidb@x%
2235 \fi%
2236 \SB@setchord{#1}%
2237 \SB@firstchordfalse%
2238 \let\SB@dothis\SB@chstart%
2239 \SB@chscan%
2240 }

\MultiwordChords The \SB@spcinit macro is invoked at the beginning of the lyric scanning process.
\SB@spcinit By default it does nothing, but if \MultiwordChords is invoked, it initializes the
lyric-scanner state to process spaces as part of lyrics.
2241 \newcommand\SB@spcinit{}
2242 \newcommand\MultiwordChords{%
2243 \def\SB@spcinit{%
2244 \let\SB@chdone\SB@chlyrdone%
2245 \def\SB@chimpspace{\let\SB@donext\SB@chdone}%
2246 \def\SB@chexpspace{\let\SB@donext\SB@chdone}%
2247 \let\SB@chespace\SB@chendspace%
2248 }%
2249 }

```

`\SB@chscan` This is the main loop of the lyric-scanner. Peek ahead at the next token without consuming it, then execute a loop body based on the current state (`\SB@dothis`), and finally go to the next iteration (`\SB@donext`).

```

2250 \newcommand\SB@chscan{%
2251 \let\SB@nextname\relax%
2252 \futurelet\SB@next\SB@chmain%
2253 }
2254 \newcommand\SB@chmain{\SB@dothis\SB@donext}

```

Warning: In the lyric-scanner macros that follow, `\SB@next` might be a macro declared `\outer`. This means that it must *never* be passed as an argument to a macro and it must never explicitly appear in any untaken branch of a conditional. If it does, the  $\TeX$  parser will complain of a runaway argument when it tries to skip over an `\outer` macro while consuming tokens at high speed.

`\SB@chstart` We begin lyric-scanning with two special cases: (1) If the chord macro is immediately followed by another chord macro with no intervening whitespace, drop out of the lyric scanner and reenter it when the second macro is parsed. The chord texts will get concatenated together above the lyric that follows. (2) If the chord macro is immediately followed by one or more quote tokens, then consume them all and output them *before* the chord. This causes the chord to sit above the actual word instead of the left-quote or left-double-quote symbol, which looks better.

```

2255 \newcommand\SB@chstart{%
2256 \ifx\SB@next\[%
2257 \let\SB@donext\relax%
2258 \else\ifx\SB@next\SB@activehat%
2259 \let\SB@donext\relax%
2260 \else\ifx\SB@next\ch%
2261 \let\SB@donext\relax%
2262 \else\ifx\SB@next\mch%
2263 \let\SB@donext\relax%
2264 \else\ifx\SB@next'%
2265 \let\SB@donext\SB@chstep%
2266 \else\ifx\SB@next'%
2267 \let\SB@donext\SB@chstep%
2268 \else\ifx\SB@next"%
2269 \let\SB@donext\SB@chstep%
2270 \else%
2271 \the\SB@lyric%
2272 \SB@lyric{%
2273 \SB@firstchordtrue%
2274 \let\SB@dothis\SB@chnorm%
2275 \SB@chnorm%
2276 \fi\fi\fi\fi\fi\fi\fi%
2277 }

```

`\SB@chnorm` First, check to see if the lyric token is a letter. Since that's the most common case, we do this check first for speed.

```

2278 \newcommand\SB@chnorm{%
2279 \ifcat\noexpand\SB@next A%
2280 \SB@testtrue\SB@lettertests%
2281 \ifSB@test%
2282 \SB@chespace\let\SB@donext\SB@chstep%
2283 \else%
2284 \let\SB@donext\SB@chdone%
2285 \fi%
2286 \else%
2287 \SB@chtrymacro%
2288 \fi%
2289 }
```

`\SB@chtrymacro` Next, check to see if it's a macro or active character. We do these checks next because these are the only cases when the token might be `\outer`. Once we eliminate that ugly possibility, we can write the rest of the code without having to worry about putting `\SB@next` in places where `\outer` tokens are illegal.

```

2290 \newcommand\SB@chtrymacro{%
2291 \ifcat\noexpand\SB@next\relax%
2292 \SB@chmacro%
2293 \else%
2294 \SB@chother%
2295 \fi%
2296 }
```

`\SB@chother` The token is not a letter, macro, or active character. The only other cases of interest are spaces, braces, and hyphens. If it's one of those, take the appropriate action; otherwise end the lyric here. Since we've eliminated the possibility of macros and active characters, we can be sure that the token isn't `\outer` at this point.

```

2297 \newcommand\SB@chother{%
2298 \ifcat\noexpand\SB@next\noexpand\@sptoken%
2299 \SB@chexpspace%
2300 \else\ifcat\noexpand\SB@next\noexpand\bgroup%
2301 \SB@chespace\let\SB@donext\SB@chbgroup%
2302 \else\ifcat\noexpand\SB@next\noexpand\egroup%
2303 \SB@chespace\let\SB@donext\SB@chegroup%
2304 \else\ifx\SB@next-%
2305 \SB@numhyps\@ne\relax%
2306 \SB@lyricnohyp\expandafter{\the\SB@lyric}%
2307 \let\SB@dothis\SB@chhyph%
2308 \SB@chespace\let\SB@donext\SB@chstep%
2309 \else\ifcat\noexpand\SB@next.%
2310 \SB@testtrue\SB@othertests%
2311 \ifSB@test%
2312 \SB@chespace\let\SB@donext\SB@chstep%
2313 \else%
2314 \let\SB@donext\SB@chdone%
```



```

2315 \fi%
2316 \else%
2317 \SB@chespace\let\SB@donext\SB@chstep%
2318 \fi\fi\fi\fi\fi%
2319 }

```

`\SB@chmacro` The lyric-scanner has encountered a macro or active character. If it's `\outer`, it should never be used in an argument, so stop here.

```

2320 \newcommand\SB@chmacro{%
2321 \SB@testfalse%
2322 \expandafter\SB@outertest\expandafter{\meaning\SB@next}%
2323 \ifSB@test%
2324 \let\SB@donext\SB@chdone%
2325 \else%
2326 \let\SB@donext\SB@chgetname%
2327 \fi%
2328 }

```

`\SB@chgetname` We've encountered a non-`\outer` macro or active character. Use `\string` to get its name, but insert the token back into the input stream since we haven't decided whether to consume it yet.

```

2329 \newcommand\SB@chgetname[1]{%
2330 \edef\SB@nextname{\string#1}%
2331 \SB@@chmacro\SB@donext#1%
2332 }

```

`\SB@@chmacro` The lyric-scanner has encountered a non-`\outer` macro or active character. Its `\stringified` name has been stored in `\SB@nextname`. Test to see if it's a known macro or the beginning of a multibyte-encoded international character. If the former, dispatch some macro-specific code to handle it. If the latter, grab the full multibyte sequence and include it in the lyric.

```

2333 \newcommand\SB@@chmacro{%
2334 \ifx\SB@next\SB@activehat%
2335 \let\SB@donext\SB@chdone%
2336 \else\ifx\SB@next\SB@par%
2337 \let\SB@donext\SB@chdone%
2338 \else\ifx\SB@next\measurebar%
2339 \SB@chmbar%
2340 \else\ifx\SB@next\mbar%
2341 \SB@chmbar%
2342 \else\ifx\SB@next\ch%
2343 \SB@chespace\let\SB@donext\SB@chlig%
2344 \else\ifx\SB@next\mch%
2345 \SB@chespace\let\SB@donext\SB@mchlig%
2346 \else\ifx\SB@next\ %
2347 \SB@chimpspace%
2348 \else\ifx\SB@next\SB@nbsp%
2349 \SB@chimpspace%
2350 \else%

```

```

2351 \expandafter\SB@UTFtest\expandafter{\meaning\SB@next}%
2352 \ifcase\SB@cnt\SB@chothermac%
2353 \or\or\SB@chespace\let\SB@donext\SB@chsteptwo%
2354 \or\SB@chespace\let\SB@donext\SB@chstepthree%
2355 \or\SB@chespace\let\SB@donext\SB@chstepfour\fi%
2356 \fi\fi\fi\fi\fi\fi\fi\fi%
2357 }

```

`\SB@chothermac` The lyric-scanner has encountered a macro or active character that is not `\outer`, not a known macro that requires special treatment, and not a multibyte international character. First, check the macro's name (stored in `\SB@nextname`) to see if it begins with a non-escape character. If so, it's probably an accenting or punctuation character made active by the `inputenc` or `babel` packages. Most such characters should be included in the lyric, so include it by default; otherwise exclude it by default. The user can override the defaults using `\DeclareLyricChar` and friends.

```

2358 \newcommand\SB@chothermac{%
2359 \SB@testfalse%
2360 \afterassignment\iffalse%
2361 \SB@cnt\expandafter'\SB@nextname x\fi%
2362 \ifnum\the\catcode\SB@cnt=\z@\else\SB@testtrue\fi%
2363 \SB@macrotests%
2364 \ifSB@test%
2365 \SB@chespace\let\SB@donext\SB@chstep%
2366 \else%
2367 \let\SB@donext\SB@chdone%
2368 \fi%
2369 }

```

`\SB@chstep` We've encountered one or more tokens that should be included in the lyric text.  
`\SB@chsteptwo` (More than one means we've encountered a multibyte encoding of an international  
`\SB@chstepthree` character.) Consume them (as arguments to this macro) and add them to the list  
`\SB@chstepfour` of tokens we've already consumed.

```

\SB@chmulti 2370 \newcommand\SB@chstep[1]{%
\SB@chmstop 2371 \SB@lyric\expandafter{\the\SB@lyric#1}%
2372 \SB@chscan%
2373 }
2374 \newcommand\SB@chsteptwo[2]{\SB@chmulti{#1#2}}
2375 \newcommand\SB@chstepthree[3]{\SB@chmulti{#1#2#3}}
2376 \newcommand\SB@chstepfour[4]{\SB@chmulti{#1#2#3#4}}
2377 \newcommand\SB@chmulti[1]{%
2378 \def\SB@next{#1}%
2379 \let\SB@nextname\relax%
2380 \SB@testtrue\SB@multitests%
2381 \ifSB@test%
2382 \SB@lyric\expandafter{\the\SB@lyric#1}%
2383 \expandafter\SB@chscan%
2384 \else%
2385 \expandafter\SB@chmstop%

```

```

2386 \fi%
2387 }
2388 \newcommand\SB@chmstop{\expandafter\SB@chdone\SB@next}

\SB@chhyph We've encountered a hyphen. Continue to digest hyphens, but terminate as soon
as we see anything else.
2389 \newcommand\SB@chhyph{%
2390 \ifx\SB@next-%
2391 \advance\SB@numhyps\@ne\relax%
2392 \let\SB@donext\SB@chstep%
2393 \else%
2394 \let\SB@donext\SB@chdone%
2395 \fi%
2396 }

\SB@chinspace We've encountered an implicit or explicit space. Normally this just ends the lyric,
\SB@chexpSPACE but if \MultiwordChords is active, these macros both get redefined to process the
space.
2397 \newcommand\SB@chinspace{\let\SB@donext\SB@chdone}
2398 \newcommand\SB@chexpSPACE{\let\SB@donext\SB@chdone}

\SB@chespace The \SB@chespace macro gets invoked by the lyric-scanner just before a non-
\SB@chendSPACE space token is about to be accepted as part of an under-chord lyric. Normally
it does nothing; however, if \MultiwordChords is active, it gets redefined to do
one of three things: (1) Initially it is set equal to \SB@chendSPACE so that if
the very first token following the chord macro is not a space, the lyric-scanner
macros are redefined to process any future spaces encountered. Otherwise the
very first token is a space, and the lyric ends immediately. (2) While scanning
non-space lyric tokens, it is set to nothing, since no special action needs to be
taken until we encounter a sequence of one or more spaces. (3) When a space
token is encountered (but not the very first token after the chord macro), it is
set equal to \SB@chendSPACE again so that \SB@chendSPACE is invoked once the
sequence of one or more space tokens is finished.
2399 \newcommand\SB@chespace{}
2400 \newcommand\SB@chendSPACE{%
2401 \let\SB@chdone\SB@chlyrdone%
2402 \def\SB@chexpSPACE{\SB@chSPACE\SB@chexpSPACE}%
2403 \def\SB@chinspace{\SB@chSPACE\SB@chinspace}%
2404 \def\SB@chespace{}%
2405 }

\SB@chSPACE The \SB@chSPACE macro gets invoked when \MultiwordChords is active and the
\SB@chgetSPACE lyric-scanner has encountered a space token that was immediately preceded by
a non-space token. Before processing the space, we add all lyrics seen so far to
the \SB@lyricbox and check its width. If we've seen enough lyrics to match or
exceed the width of the chord, a space stops the lyric-scanning process. (This is
important because it minimizes the size of the chord box, providing as many line
breakpoints as possible to the paragraph-formatter.)

```

Otherwise we begin scanning space tokens without adding them to the lyric until we see what the next non-space token is. If the next non-space token would have ended the lyric anyway, roll back and end the lyric here, reinserting the space tokens back into the token stream. If the next non-space token would have been included in the lyric, the lyric-scanner proceeds as normal.

```

2406 \newcommand\SB@chbspace{%
2407 \setbox\SB@lyricbox\hbox{%
2408 \unhbox\SB@lyricbox%
2409 \the\SB@lyric%
2410 }%
2411 \SB@lyric}%
2412 \ifdim\wd\SB@lyricbox<\wd\SB@chordbox%
2413 \let\SB@chbstok= \SB@next%
2414 \def\SB@chexpspace{\let\SB@donext\SB@chgetspace}%
2415 \def\SB@chimpspace{\let\SB@donext\SB@chstep}%
2416 \let\SB@chespace\SB@chendspace%
2417 \let\SB@chdone\SB@chspcdone%
2418 \else%
2419 \def\SB@chimpspace{\let\SB@donext\SB@chdone}%
2420 \def\SB@chexpspace{\let\SB@donext\SB@chdone}%
2421 \fi%
2422 }
2423 \newcommand\SB@chgetspace{%
2424 \SB@appendsp\SB@lyric%
2425 \let\SB@nextname\relax%
2426 \afterassignment\SB@chscan%
2427 \let\SB@next= }

```

`\SB@chmbar` We've encountered a measure bar. Either ignore it or end the lyric text, depending on whether measure bars are being displayed.

```

2428 \newcommand\SB@chmbar{%
2429 \ifmeasures%
2430 \let\SB@donext\SB@chdone%
2431 \else%
2432 \SB@chespace\let\SB@donext\SB@chstep%
2433 \fi%
2434 }

```

`\SB@chbgroup` We've encountered a begin-group brace. Consume the entire group that it begins, and add it to the list of tokens including the begin and end group tokens.

```

2435 \newcommand\SB@chbgroup[1]{%
2436 \SB@lyric\expandafter{\the\SB@lyric{#1}}%
2437 \SB@chscan%
2438 }

```

`\SB@chegroup` We've encountered an end-group brace whose matching begin-group brace must have come before the chord macro itself. This forcibly ends the lyric text.  
`\SB@chegrpscan` Before stopping, we must set `\SB@next` to the token following the brace and  
`\SB@chegrpmacro`  
`\SB@chegrpouter`  
`\SB@chegrpname`  
`\SB@chegrpdone`

\SB@nextname to its \stringified name so that \SB@emitichord will know whether to add hyphenation. Therefore, we temporarily consume the end-group brace, then scan the next token without consuming it, and finally reinsert the end-group brace and stop.

```

2439 \newcommand\SB@chegroup{%
2440 \let\SB@nextname\relax%
2441 \afterassignment\SB@chegrpscan%
2442 \let\SB@next= }
2443 \newcommand\SB@chegrpscan{%
2444 \futurelet\SB@next\SB@chegrpmacro%
2445 }
2446 \newcommand\SB@chegrpmacro{%
2447 \ifcat\noexpand\SB@next\relax%
2448 \expandafter\SB@chegrpouter%
2449 \else%
2450 \expandafter\SB@chegrpdone%
2451 \fi%
2452 }
2453 \newcommand\SB@chegrpouter{%
2454 \SB@testfalse%
2455 \expandafter\SB@outertest\expandafter{\meaning\SB@next}%
2456 \ifSB@test%
2457 \expandafter\SB@chegrpdone%
2458 \else%
2459 \expandafter\SB@chegrpname%
2460 \fi%
2461 }
2462 \newcommand\SB@chegrpname[1]{%
2463 \edef\SB@nextname{\string#1}%
2464 \SB@chegrpdone#1%
2465 }
2466 \newcommand\SB@chegrpdone{\SB@chdone\egroup}

```

\SB@chlig We've encountered a \ch chord-over-ligature macro, or an \mch measurebar-and-chord-over-ligature macro. Consume it and all of its arguments, and load them into some registers for future processing. (Part of the ligature might fall into this lyric text or might not, depending on if we decide to add hyphenation.) Then end the lyric text here.

```

2467 \newcommand\SB@chlig[5]{%
2468 \gdef\SB@ligpre{#{3}}%
2469 \gdef\SB@ligpost{[#2]{#4}}%
2470 \gdef\SB@ligfull{\[\SB@noreplay{\hphantom{\{\lyricfont#3}\}}#2]{#5}}%
2471 \SB@chdone%
2472 }
2473 \newcommand\SB@mchlig[5]{%
2474 \SB@lyric\expandafter{the\SB@lyric#3}%
2475 \let\SB@next\measurebar%
2476 \edef\SB@nextname{\string\measurebar}%
2477 \gdef\SB@ligpost{\measurebar[#2]{#4}}%

```

```

2478 \gdef\SB@ligfull{\measurebar\[#2]{#4}}%
2479 \SB@chdone%
2480 }

\SB@chdone The \SB@chdone macro is invoked when we've decided to end the lyric text
\SB@chlyrdone (usually because we've encountered a non-lyric token). Normally this expands to
\SB@chspcdone \SB@chlyrdone, which adds any uncontributed lyric material to the \SB@lyricbox
and jumps to the main chord formatting macro. However, if \MultiwordChords
is active and if the lyric ended with a sequence of one or more space tokens, then
we instead reinsert the space tokens into the token stream without contributing
them to the \SB@lyricbox.

2481 \newcommand\SB@chlyrdone{%
2482 \setbox\SB@lyricbox\hbox{%
2483 \unhbox\SB@lyricbox%
2484 \ifnum\SB@numhyphs=\@ne%
2485 \the\SB@lyricnohyp%
2486 \else%
2487 \the\SB@lyric%
2488 \fi%
2489 }%
2490 \SB@emitichord%
2491 }
2492 \newcommand\SB@chspcdone{%
2493 \let\SB@nextname\relax%
2494 \let\SB@next= \SB@chbstok%
2495 \expandafter\SB@emitichord\the\SB@lyric%
2496 }
2497 \newcommand\SB@chdone{}
2498 \let\SB@chdone\SB@chlyrdone

\SB@ligpre The following three macros record arguments passed to a \ch macro that concludes
\SB@ligpost the lyric text of the \[] macro currently being processed.
\SB@ligfull 2499 \newcommand\SB@ligpre{}
2500 \newcommand\SB@ligpost{}
2501 \newcommand\SB@ligfull{}

\SB@clearlig Clear all ligature-chord registers.

2502 \newcommand\SB@clearlig{%
2503 \gdef\SB@ligpre{}%
2504 \gdef\SB@ligpost{}%
2505 \gdef\SB@ligfull{}%
2506 }

```

## 16.11 Chords

**\SB@emitichord** The **\SB@emitichord** macro does the actual work of typesetting chord text over lyric text, introducing appropriate hyphenation when necessary. We begin by consulting **\SB@next**, which should have been set by the lyric-scanning code in §16.10

to the token that immediately follows the lyric under this chord, to determine whether the lyric text ends on a word boundary.

```

2507 \newcommand\SB@emit chord{%
2508 \ifSB@inverse\else\ifSB@inchorus\else\SB@err chord\fi\fi%
2509 \SB@test false%
2510 \ifcat\noexpand\SB@next\noexpand\@sptoken\SB@test true\fi%
2511 \ifcat\noexpand\SB@next.\SB@test true\fi%
2512 \ifx\SB@next\SB@par\SB@test true\fi%
2513 \ifx\SB@next\egroup\SB@test true\fi%
2514 \ifx\SB@next\endgroup\SB@test true\fi%
2515 \SB@hyphtests%
2516 \ifSB@test\SB@wordendstrue\else\SB@wordendsfalse\fi%

```

Next, compare the width of the lyric to the width of the chord to determine if hyphenation might be necessary. The original lyric text might have ended in a string of one or more explicit hyphens, enumerated by `\SB@numhyps`. If it ended in exactly one, the lyric-scanning code suppresses that hyphen so that we can here add a new hyphen that floats out away from the word when the chord above it is long. If it ended in more than one (e.g., the encoding of an en- or em-dash) then the lyric-scanner leaves it alone; we must not add any hyphenation or float the dash away from the word.

There is also code here to insert a penalty that discourages linebreaking immediately before lyricless chords. Beginning a wrapped line with a lyricless chord is undesirable because it makes it look as though the wrapped line is extra-indented (due to the empty lyric space below the chord). It should therefore happen only as a last resort.

```

2517 \SB@dimen\wd\SB@chordbox%
2518 \ifvmode\leavevmode\fi%
2519 \SB@brokenword false%
2520 \ifdim\wd\SB@lyricbox>\z@%
2521 \ifdim\SB@dimen>\wd\SB@lyricbox%
2522 \ifSB@wordends\else\SB@brokenword true\fi%
2523 \fi%
2524 \else%
2525 \SB@skip\lastskip%
2526 \unskip\penalty200\hskip\SB@skip%
2527 \fi%
2528 \ifnum\SB@numhyps>\z@%
2529 \ifnum\SB@numhyps>\@ne%
2530 \SB@brokenword false%
2531 \else%
2532 \SB@brokenword true%
2533 \fi%
2534 \fi%

```

If lyrics are suppressed on this line (e.g., by using `\nolyrics`), then just typeset the chord text on the natural baseline.

```

2535 \SB@test false%
2536 \ifnolyrics\ifdim\wd\SB@lyricbox=\z@\SB@test true\fi\fi%

```

```

2537 \ifSB@test%
2538 \unhbox\SB@chordbox%
2539 \gdef\SB@temp{\expandafter\SB@clearlig\SB@ligfull}%
2540 \else%

```

Otherwise, typeset the chord above the lyric on a double-height line.

```

2541 \vbox{\clineparams\relax%
2542 \ifSB@brokenword%
2543 \global\setbox\SB@lyricbox\hbox{%
2544 \unhbox\SB@lyricbox%
2545 \SB@ligpre%
2546 }%
2547 \SB@maxmin\SB@dimen<{\wd\SB@lyricbox}%
2548 \advance\SB@dimen.5em%
2549 \hbox to\SB@dimen{\unhbox\SB@chordbox\hfil}%
2550 \hbox to\SB@dimen{%
2551 \unhcopy\SB@lyricbox\hfil\char\hyphenchar\font\hfil%
2552 }%
2553 \global\SB@cnt\@m%
2554 \gdef\SB@temp{\expandafter\SB@clearlig\SB@ligpost}%
2555 \else%
2556 \box\SB@chordbox%
2557 \hbox{%
2558 \unhcopy\SB@lyricbox%
2559 \global\SB@cnt\spacefactor%
2560 \hfil%
2561 }%
2562 \gdef\SB@temp{\expandafter\SB@clearlig\SB@ligfull}%
2563 \fi%
2564 }%

```

If the chord is lyricless, inhibit a linebreak immediately following it. This prevents sequences of lyricless chords (which often end lines) from being wrapped in the middle, which looks very unsightly and makes them difficult to read. If the chord has a lyric but it doesn't end on a word boundary, insert an appropriate penalty to prevent linebreaking without hyphenation. Also preserve the spacefactor in this case, which allows L<sup>A</sup>T<sub>E</sub>X to fine-tune the spacing between consecutive characters in the word that contains the chord.

```

2565 \ifSB@wordends%
2566 \ifdim\wd\SB@lyricbox>\z@\else\nobreak\fi%
2567 \else%
2568 \penalty%
2569 \ifnum\SB@numhyps>\z@\exhyphenpenalty%
2570 \else\ifSB@brokenword\hyphenpenalty%
2571 \else\@M\fi\fi%
2572 \spacefactor\SB@cnt%
2573 \fi%
2574 \fi%

```

Finally, end the macro with some code that handles the special case that this chord is immediately followed by a chord-over-ligature macro. The code above



sets `\SB@temp` to the portion of the ligature that should come after this chord but before the chord that tops the ligature. This text must be inserted here.

```
2575 \SB@temp%
2576 }
```

`\SB@accidental` Typeset an accidental symbol as a superscript within a chord. Since chord names are often in italics but math symbols like sharp and flat are not, we need to do some kerning adjustments before and after the accidental to position it as if it were italicized. The pre-adjustment is just a simple italic correction using `\/`. The post-adjustment is based on the current font's slant-per-point metric.

```
2577 \newcommand\SB@accidental[1]{\%
2578 \/%
2579 \m@th#1%
2580 \SB@dimen-\fontdimen\@ne\font%
2581 \advance\SB@dimen.088142\p@%
2582 \ifdim\SB@dimen<\z@%
2583 \kern\f@size\SB@dimen%
2584 \fi%
2585 }}
```

`\sharpsymbol` When changing the sharp or flat symbol, change these macros rather than changing `\flatsymbol` `\shrp` or `\flt`. This will ensure that other shortcuts like `#` and `&` will reflect your change.

```
2586 \newcommand\sharpsymbol{\ensuremath{\sim\#}}
2587 \newcommand\flatsymbol{\raise.5ex\hbox{\{\SB@flatsize\flat}}}
```

`\shrp` These macros typeset sharp and flat symbols.

```
\flt 2588 \newcommand\shrp{\SB@accidental\sharpsymbol}
2589 \newcommand\flt{\SB@accidental\flatsymbol}
```

`\DeclareFlatSize` The `\flat` math symbol is too small for properly typesetting chord names. (Its size was designed for staff notation not textual chord names.) The correct size for the symbol should be approximately 30% larger than the current superscript size, or 90% of the base font size  $b$ . However, simply computing  $0.9b$  does not work well because most fonts do not render well in arbitrary sizes. To solve the problem, we must therefore choose an appropriate size individually for each possible base font size  $b$ . This is the solution adopted by the rest of  $\text{\LaTeX}$  for such things. For example,  $\text{\LaTeX}$ 's `\DeclareMathSizes` macro defines an appropriate superscript size for each possible base font size. The macro below creates a similar macro that defines an appropriate flat-symbol size for each possible base font size.

```
2590 \newcommand\DeclareFlatSize[2]{\%
2591 \expandafter\edef\csname SB@flatsize@#1\endcsname{#2}%
2592 }
2593 \DeclareFlatSize\@vpt\@vpt
2594 \DeclareFlatSize\@vipt\@vipt
2595 \DeclareFlatSize\@viipt\@viipt
2596 \DeclareFlatSize\@viiipt\@viiipt
2597 \DeclareFlatSize\@ixpt\@ixpt
```

```

2598 \DeclareFlatSize\@xpt\@ixpt
2599 \DeclareFlatSize\@xipt\@xpt
2600 \DeclareFlatSize\@xipt\@xipt
2601 \DeclareFlatSize\@xivpt\@xipt
2602 \DeclareFlatSize\@xvipt\@xivpt
2603 \DeclareFlatSize\@xxpt\@xvipt
2604 \DeclareFlatSize\@xxvpt\@xxpt

```

**\SB@flatsize** Select the correct flat symbol size based on the current font size.

```

2605 \newcommand\SB@flatsize{%
2606 \ifundefined{SB@flatsize@f@size}{}{%
2607 \expandafter\fontsize%
2608 \csname SB@flatsize@f@size\endcsname\font@baselineskip%
2609 \selectfont%
2610 }%
2611 }

```

In the following code, the `\ch`, `\mch`, `\[`, and `\^` macros are each defined to be a single macro that then expands to the real definition. This is necessary because the top-level definitions of each must stay the same in order to allow the lyric-scanning code to uniquely identify them, yet their internal definitions must be redefined by code that turns chords and/or measure bars on and off. Such code redefines `\SB@ch`, `\SB@mch`, `\SB@bracket`, and `\SB@rechord` to effect a change of mode without touching the top-level definitions.

**\ch** The `\ch` macro puts a chord atop a ligature without breaking the ligature. Normally this just means placing the chord midway over the unbroken ligature (ignoring `\SB@ch` the third argument completely). However, when a previous chord macro encounters it while scanning ahead in the input stream to parse its lyric, the `\ch` macro `\SB@ch` itself is not actually expanded at all. Instead, the chord macro scans ahead, spots `\SB@@@ch` the `\ch` macro, gobbles it, and then steals its arguments, breaking the ligature with hyphenation. Thus, the `\ch` macro is only actually expanded when the ligature shouldn't be broken.

```

2612 \newcommand\ch{\SB@ch}
2613 \newcommand\SB@ch{}
2614 \newcommand\SB@ch@on{\SB@begincname\SB@ch}
2615 \newcommand*\SB@@ch[1]{\SB@endcname\SB@@@ch{#1}}
2616 \newcommand*\SB@@@ch[4]{\[\SB@noreplay{\hphantom{#2}}#1]#4}
2617 \newcommand*\SB@ch@off[4]{#4}

```

**\mch** The `\mch` macro is like `\ch` except that it also introduces a measure bar.

```

\SB@mch 2618 \newcommand\mch{\SB@mch}
\SB@mch@m 2619 \newcommand\SB@mch{}
\SB@mch@on 2620 \newcommand*\SB@mch@m[4]{#2\measurebar#3}
\SB@mch 2621 \newcommand\SB@mch@on{\SB@begincname\SB@mch}
\SB@@@mch 2622 \newcommand*\SB@@mch[1]{\SB@endcname\SB@@@mch{#1}}
2623 \newcommand*\SB@@@mch[4]{#2\measurebar\[#1]#3}

```

`\SB@activehat` This macro must always contain the current definition of the `^` chord-replay active character, in order for the lyric scanner to properly identify it and insert proper hyphenation when necessary.

```

2624 \newcommand\SB@activehat{%
2625 \ifmmode^~\else\expandafter\SB@rechord\fi%
2626 }

```

`\SB@loadactives` It's cumbersome to have to type `\shrp`, `\flt`, and `\mbar` every time you want a sharp, flat, or measure bar, so within verses and choruses we allow the hash, ampersand, and pipe symbols to perform the those functions too. It's also cumbersome to have to type something like `\chord{Am}{lyric}` to produce each chord. As an easier alternative, we here define `\[Am]` to typeset chords.

```

2627 \newcommand\SB@loadactives{}
2628 {
2629 \catcode'\&\active
2630 \catcode'\#\active
2631 \catcode'\|\active
2632 \catcode'\^~\active
2633 \global\let&\flt
2634 \global\let#\shrp
2635 \global\let|\measurebar
2636 \global\let^~\SB@activehat
2637 \gdef\SB@loadactives{%
2638 \catcode'\^~\ifchorded\active\else9 \fi%
2639 \catcode'\|\ifmeasures\active\else9 \fi%
2640 \def\[{ \SB@bracket }]%
2641 }
2642 }

```

## 16.12 Chord Replaying

`\SB@trackch` While inside a verse where the chord history is being remembered for future verses, `\SB@trackch` is true.

```

2643 \newif\ifSB@trackch

```

`\SB@cr@` Reserve token registers to record a history of the chords seen in a verse.

```

2644 \newtoks\SB@cr@
2645 \newtoks\SB@ctail

```

`\SB@creg` The following control sequence equals the token register being memorized into or replayed from.

```

2646 \newcommand\SB@creg{}

```

`\newchords` Allocate a new chord-replay register to hold memorized chords.

```
2647 \newcommand\newchords[1]{%
2648 \@ifundefined{SB@cr@#1}{%
2649 \expandafter\newtoks\csname SB@cr@#1\endcsname%
2650 \global\csname SB@cr@#1\endcsname{\}%
2651 }\@SB@errdup{#1}}%
2652 }
```

`\memorize` Saying `\memorize` throws out any previously memorized list of chords and starts memorizing chords until the end of the current verse or chorus.

```
2653 \newcommand\memorize{%
2654 \@ifnextchar[\@SB@memorize{\@SB@memorize[]}%
2655 }
2656 \newcommand\@SB@memorize{%
2657 \def\@SB@memorize[#1]{%
2658 \@ifundefined{SB@cr@#1}{\@SB@errreg{#1}}{%
2659 \SB@trackchtrue%
2660 \global\expandafter\let\expandafter\SB@creg%
2661 \csname SB@cr@#1\endcsname%
2662 \global\SB@creg{\}%
2663 }%
2664 }
```

`\replay` Saying `\replay` stops any memorization and begins replaying memorized chords.

```
\@SB@replay 2665 \newcommand\replay{\@ifnextchar[\@SB@replay\@SB@@replay}
\@SB@@replay 2666 \newcommand\@SB@replay{%
2667 \def\@SB@replay[#1]{%
2668 \@ifundefined{SB@cr@#1}{\@SB@errreg{#1}}{%
2669 \SB@trackchfalse%
2670 \global\expandafter\let\expandafter\SB@creg%
2671 \csname SB@cr@#1\endcsname%
2672 \global\SB@ctail\SB@creg%
2673 }%
2674 }
2675 \newcommand\@SB@@replay{%
2676 \SB@trackchfalse%
2677 \global\SB@ctail\SB@creg%
2678 }
```

`\SB@rechord` Replay the same chord that was in a previous verse.

```
\@SB@@rechord 2679 \newcommand\@SB@rechord{%
2680 \newcommand\@SB@@rechord{%
2681 \SB@ifempty\SB@ctail{%
2682 \SB@errreplay%
2683 \SB@toks{}}%
2684 \let\SB@donext\@gobble%
2685 }%
2686 \SB@lop\SB@ctail\SB@toks%
2687 \let\SB@donext\SB@chord%
```

```

2688 \let\SB@noreplay\@gobble%
2689 }%
2690 \expandafter\SB@donext\the\SB@toks]%
2691 }

```

**\ifSB@nohat** The **\ifSB@nohat** conditional is set to false when a chord macro contains a  $\hat$  in its argument. This suppresses the recording mechanism momentarily so that replays will skip this chord.

```

2692 \newif\ifSB@nohat

```

**\SB@noreplay** Sometimes material must be added to a chord but omitted when the chord is replayed. We accomplish this by enclosing such material in **\SB@noreplay** macros, which are set to **\@gobble** just before a replay and reset to **\@firstofone** at other times.

```

2693 \newcommand\SB@noreplay{}
2694 \let\SB@noreplay\@firstofone

```

## 16.13 Guitar Tablatures

The song book software not only supports chord names alone, but can also typeset guitar tablature diagrams. The macros for producing these diagrams are found here.

**\SB@fretwidth** Set the width of each vertical string in the tablature diagram.

```

2695 \newlength\SB@fretwidth
2696 \setlength\SB@fretwidth{6\p@}

```

**\SB@fretnum** Typeset a fret number to appear to the left of the diagram.

```

2697 \newcommand\SB@fretnum[1]{\%
2698 \sffamily\fontsize\@xpt\@xpt\selectfont#1%
2699 }

```

**\SB@onfret** Typeset one string of one fret with  $\langle arg1 \rangle$  typeset overtop of it (usually a dot or nothing at all).

```

2700 \newcommand\SB@onfret[1]{\%
2701 \rlap{\hbox to\SB@fretwidth{\hfil\vrule\@height6\p@\hfil}}\%
2702 \hbox to\SB@fretwidth{\hfil#1\hfil}\%
2703 }

```

**\SB@atopfret** Typeset material (given by  $\langle arg1 \rangle$ ) to be placed above a string in the tablature diagram.

```

2704 \newcommand\SB@atopfret[1]{\%
2705 \hbox to\SB@fretwidth{\hfil#1\hfil}\%
2706 }

```

`\SB@fretbar` Typeset a horizontal fret bar of width `\SB@dimen`.

```

2707 \newcommand\SB@fretbar{%
2708 \nointerlineskip%
2709 \hbox to\SB@dimen{%
2710 \advance\SB@dimen-\SB@fretwidth%
2711 \advance\SB@dimen.4\p@%
2712 \hfil%
2713 \vrule\@width\SB@dimen\@height.4\p@\@depth\z@%
2714 \hfil%
2715 }%
2716 \nointerlineskip%
2717 }
```

`\SB@topempty` Above a string in a tablature diagram there can be nothing, an  $\times$ , or an  $\circ$ .

```

\SB@topX 2718 \newcommand\SB@topempty{\SB@atopfret\relax}
\SB@topO 2719 \newcommand\SB@topX{\SB@atopfret{%
2720 \hbox{%
2721 \kern-.2\p@%
2722 \fontencoding{OMS}\fontfamily{cmsy}%
2723 \fontseries{m}\fontshape{n}%
2724 \fontsize\@viipt\@viipt\selectfont\char\tw@%
2725 \kern-.2\p@%
2726 }%
2727 }}
2728 \newcommand\SB@topO{\SB@atopfret{%
2729 \vrule\@width\z@\@height4.3333\p@\@depth.8333\p@%
2730 \lower.74\p@\hbox{%
2731 \fontencoding{OMS}\fontfamily{cmsy}%
2732 \fontseries{m}\fontshape{n}%
2733 \fontsize\@xpt\@xpt\selectfont\char14%
2734 }%
2735 }}
```

`\SB@fretempty` On a string in a fret diagram there can be nothing or a filled circle.

```

\SB@frethit 2736 \newcommand\SB@fretempty{\SB@onfret\relax}
2737 \newcommand\SB@frethit{\SB@onfret{%
2738 \hbox{%
2739 \fontencoding{OMS}\fontfamily{cmsy}%
2740 \fontseries{m}\fontshape{n}%
2741 \fontsize\@xiipt\@xiipt\selectfont\char15%
2742 }%
2743 }}
```

`\SB@finger` If we're including fingering info in the tablature diagram, then below each string there might be a number.

```

2744 \newcommand\SB@finger[1]{%
2745 \SB@atopfret{\sfamily\fontsize\@viipt\@viipt\selectfont#1}%
2746 }
```

`\ifSB@gettabind` Lyrics under tablature diagrams look odd if they aren't aligned with the leftmost string of the diagram. To accomplish this, the following two macros record the amount by which a lyric under this tablature diagram must be indented to position it properly.

```
2747 \newif\ifSB@gettabind\SB@gettabindfalse
2748 \newdimen\SB@tabindent
```

`\SB@targfret` Reserve some macro names in which to store the three pieces of the second argument to the `\gtab` macro. The first is for the fret number, the second is for the `\SB@targstr` *<strings>* info, and the last is for the *<fingering>* info.

```
2749 \newcommand\SB@targfret{}
2750 \newcommand\SB@targstr{}
2751 \newcommand\SB@targfing{}
```

In general `\gtab` macros often appear inside chord macros, which means that their arguments have already been scanned by the time the `\gtab` macro itself is expanded. This means that catcodes cannot be reassigned (without resorting to  $\varepsilon$ -TEX).

We therefore adopt the alternative strategy of converting each token in the *<strings>* and *<fingering>* arguments of a `\gtab` macro into a control sequence (using `\csname`). We can then temporarily assign meanings to those control sequences and replay the arguments to achieve various effects.

`\SB@csify` Convert all tokens in the first argument to control sequences and store the resulting sequence into the macro given by the first argument. Store the length in tokens into counter register `\SB@cnt`.

```
2752 \newcommand\SB@csify[2]{%
2753 \SB@toks{}%
2754 \SB@cnt\z@%
2755 \SB@@csify#2\SB@csify%
2756 \edef#1{\the\SB@toks}%
2757 }
2758 \newcommand\SB@@csify[1]{%
2759 \ifx#1\SB@@csify\else%
2760 \advance\SB@cnt\@ne%
2761 \SB@toks\expandafter{\the\SB@toks\csname#1\endcsname}%
2762 \expandafter\SB@@csify%
2763 \fi%
2764 }
```

`\SB@gttop` Different meanings are assigned to digits, X's, and O's depending on whether we `\SB@gtinit` are currently typesetting the material overtop the diagram, the interior of the diagram, or the fingering numbers below the diagram. These meanings are set by `\SB@gtset` `\SB@gttop`, `\SB@gtinit` & `\SB@gtinc`, and `\SB@gtset`, respectively.

```
2765 \newcommand\SB@gttop{%
2766 \let\X\SB@topX\let\O\SB@topO\let\0\0\let\1\SB@topempty%
2767 \let\2\1\let\3\1\let\4\1\let\5\1%
2768 \let\6\1\let\7\1\let\8\1\let\9\1%
```

```

2769 }
2770 \newcommand\SB@gtinit{%
2771 \let\X\SB@fretempty\let\0\X\let\0\X\let\1\SB@frethit%
2772 \let\2\X\let\3\X\let\4\X\let\5\X%
2773 \let\6\X\let\7\X\let\8\X\let\9\X%
2774 }
2775 \newcommand\SB@gtinc{%
2776 \let\9\8\let\8\7\let\7\6\let\6\5\let\5\4%
2777 \let\4\3\let\3\2\let\2\1\let\1\0%
2778 }
2779 \newcommand\SB@gtset[2]{%
2780 \let\X#1\let\0\X\let\0\X%
2781 \def\1{#21}\def\2{#22}\def\3{#23}%
2782 \def\4{#24}\def\5{#25}\def\6{#26}%
2783 \def\7{#27}\def\8{#28}\def\9{#29}%
2784 }

```

**\SB@gtmax** To compute the height of the tablature diagram, we must identify the maximum fret number in the *<strings>* argument. This is accomplished by using the following macro in combination with **\SB@gtset** above.

```

2785 \newcommand\SB@gtmax[1]{\ifnum\SB@cnt<#1\SB@cnt#1\fi}

```

**\gtab** A **\gtab** macro begins by setting catcodes suitable for parsing a chord name as its first argument. This allows tokens like # and & to be used for sharp and flat even when **\gtab** is used outside a chord macro. Colon is reset to a non-active character while processing the second argument to avoid a potential conflict with Babel French.

```

2786 \newcommand\gtab{\SB@begincname\SB@gtab}
2787 \newcommand*\SB@gtab[1]{%
2788 \SB@endcname%
2789 \begingroup%
2790 \catcode'\:12\relax%
2791 \SB@@gtab{#1}%
2792 }

```

**\SB@@gtab** If transposition is currently taking place, allow the user to customize the behavior by redefining **\gtabtrans**. Using **\gtab** within **\gtabtrans** should go directly to **\SB@@@gtab** (otherwise an infinite loop would result!).

```

2793 \newcommand*\SB@@gtab[2]{%
2794 \endgroup%
2795 \ifnum\SB@transposefactor=\z@%
2796 \SB@@@gtab{#1}{#2}%
2797 \else%
2798 \begingroup%
2799 \let\gtab\SB@@@gtab%
2800 \gtabtrans{#1}{#2}%
2801 \endgroup%
2802 \fi%
2803 }

```



`\gtabtrans` By default, transposed guitar tablatures just display the transposed chord name and omit the diagram. Transposing a tablature diagram requires manual judgment calls for most stringed instruments, so we can't do it automatically.

2804 `\newcommand\gtabtrans[2]{\transposehere{#1}}`

`\SB@@@gtab` Typeset a full tablature diagram. Text  $\langle arg1 \rangle$  is a chord name placed above the diagram. Text  $\langle arg2 \rangle$  consists of a colon-separated list of: (1) an optional fret number placed to the left of the diagram; (2) a sequence of tokens, each of which can be X (to place an  $\times$  above the string), 0 or O (to place an  $\circ$  above the string), or one of 1 through 9 (to place a filled circle on that string at the fret of the given number); and (3) an optional sequence of tokens, each of which is either 0 (no fingering information for that string), or one of 1 through 4 (to place the given number under that string).

```

2805 \newcommand\SB@@@gtab[2]{%
2806 \let\SB@targfret\@empty%
2807 \let\SB@targstr\@empty%
2808 \let\SB@targfing\@empty%
2809 \SB@tabargs#2::\SB@tabargs%
2810 \ifx\SB@targstr\@empty%
2811 \def\SB@targstr{\0\0\0\0\0}%
2812 \fi%
2813 \ifvmode\leavevmode\fi%
2814 \vbox{%
2815 \normalfont\normalsize%
2816 \setbox\SB@box\hbox{%
2817 \thinspace{\printchord{\transposehere{#1}\strut}}\thinspace%
2818 }%
2819 \setbox\SB@boxii\hbox{\SB@fretnum{\SB@targfret}}%
2820 \setbox\SB@boxiii\hbox{\SB@gttop\SB@targstr}%
2821 \hsize\wd\SB@box%
2822 \ifSB@gettabind%
2823 \global\SB@tabindent\wd\SB@boxii%
2824 \global\advance\SB@tabindent.5\SB@fretwidth%
2825 \global\advance\SB@tabindent-.5\p@%
2826 \fi%
2827 \SB@dimen\wd\SB@boxii%
2828 \advance\SB@dimen\wd\SB@boxiii%
2829 \ifdim\hsize<\SB@dimen%
2830 \hsize\SB@dimen%
2831 \else\ifSB@gettabind%
2832 \SB@dimenii\hsize%
2833 \advance\SB@dimenii-\SB@dimen%
2834 \divide\SB@dimenii\tw@%
2835 \global\advance\SB@tabindent\SB@dimenii%
2836 \fi\fi%
2837 \hbox to\hsize{\hfil\unhbox\SB@box\hfil}%
2838 \kern-\p@\nointerlineskip%
2839 \hbox to\hsize{%
2840 \hfil%
```

```

2841 \vtop{\kern\p@\kern2\p@\box\SB@boxii}%
2842 \vtop{%
2843 \SB@dimen\wd\SB@boxiii%
2844 \box\SB@boxiii%
2845 \SB@cnt\minfrets%
2846 \SB@gtset\relax\SB@gtmax\SB@targstr%
2847 \SB@gtinit%
2848 \loop%
2849 \SB@fretbar\hbox{\SB@targstr}%
2850 \advance\SB@cnt\m@ne%
2851 \ifnum\SB@cnt>\z@\SB@gtinc\repeat%
2852 \SB@fretbar%
2853 \ifx\SB@targsfing\@empty\else%
2854 \kern1.5\p@%
2855 \SB@gtset\SB@topempty\SB@finger%
2856 \hbox{\SB@targfing}%
2857 \fi%
2858 }%
2859 \hfil%
2860 }%
2861 \kern3\p@%
2862 }%
2863 \SB@gettabindfalse%
2864 }

```

**\SB@tabargs** Break the second argument to a **\gtab** macro into three sub-arguments. The possible forms are: (a) *<strings>*, (b) *<fret>: <strings>*, (c) *<strings>: <fingering>*, or (d) *<fret>: <strings>: <fingering>*. To distinguish forms (b) and (c), we count the number of tokens before the first colon. If there is only one token, we assume it must be form (b), since frets larger than 9 and 1-stringed instruments are both rare. Otherwise we assume form (c).

```

2865 \newcommand\SB@ctoken{} \def\SB@ctoken{:}
2866 \newcommand\SB@tabargs{}
2867 \def\SB@tabargs#1:#2:#3:#4\SB@tabargs{%
2868 \def\SB@temp{#4}%
2869 \ifx\SB@temp\@empty%
2870 \SB@csify\SB@targstr{#1}%
2871 \else\ifx\SB@temp\SB@ctoken%
2872 \SB@csify\SB@targstr{#1}%
2873 \ifnum\SB@cnt>\@ne%
2874 \SB@cntii\SB@cnt%
2875 \SB@csify\SB@targfing{#2}%
2876 \SB@cnt\SB@cntii%
2877 \else%
2878 \def\SB@targfret{#1}%
2879 \SB@csify\SB@targstr{#2}%
2880 \fi%
2881 \else%
2882 \def\SB@targfret{#1}%

```

```

2883 \SB@csify\SB@targfing{#3}%
2884 \SB@csify\SB@targstr{#2}%
2885 \fi\fi%
2886 }

```

## 16.14 Book Sectioning

The following macros divide the song book into distinct sections, each with different headers, different song numbering styles, different indexes, etc.

**\songchapter** Format the chapter header for a chapter in a song book. By default, chapter headers on a song book omit the chapter number, but do include an entry in the pdf index or table of contents. Thus, the chapter has a number; it's just not displayed at the start of the chapter.

```

2887 \newcommand\songchapter{%
2888 \let\SB@temp\@seccntformat%
2889 \def\@seccntformat##1{%
2890 \@startsection{chapter}{0}{\z@}%
2891 {3.5ex\@plus1ex\@minus.2ex}%
2892 {.4ex\let\@seccntformat\SB@temp}%
2893 {\sffamily\bfseries\LARGE\centering}%
2894 }

```

**\songsection** Format the section header for a section in a song book. This is the same as for chapter headers except at the section level.

```

2895 \newcommand\songsection{%
2896 \let\SB@temp\@seccntformat%
2897 \def\@seccntformat##1{%
2898 \@startsection{section}{1}{\z@}%
2899 {3.5ex\@plus1ex\@minus.2ex}%
2900 {.4ex\let\@seccntformat\SB@temp}%
2901 {\sffamily\bfseries\LARGE\centering}%
2902 }

```

**songs** Begin and end a book section. The argument is a list of indexes with which to associate songs in this section.

```

2903 \newenvironment{songs}[1]{%
2904 \ifSB@songsenv\SB@errnse\fi%
2905 \gdef\SB@indexlist{#1}%
2906 \SB@chkidxlst%
2907 \stepcounter{SB@songsnum}%
2908 \setcounter{songnum}{1}%
2909 \let\SB@sgroup\@empty%
2910 \ifinner\else\ifdim\pagetotal>\z@%
2911 \null\nointerlineskip%
2912 \fi\fi%
2913 \songcolumns\SB@numcols%
2914 \SB@songsenvtrue%
2915 }{%

```

```

2916 \commitsongs%
2917 \global\let\SB@indexlist\@empty%
2918 \ifinner\else\clearpage\fi%
2919 \SB@songsenvfalse%
2920 }

```

Each `songs` section needs a unique number to aid in hyperlinking.

```

2921 \newcounter{SB@songsnum}

```

## 16.15 Index Generation

The following macros generate the various types of indexes. At present there are four types:

1. A “large” index has a separate section for each capital letter and is printed in two columns.
2. A “small” index has only a single column, centered, and has no sections.
3. A “scripture” index has three columns and each entry has a comma-separated list of references.
4. An “author” index is like a large index except in bold and without the sectioning.

“Large” and “small” indexes will be chosen automatically based on the number of index entries when building a song index. The other two types are designated by the user.

As is typical of  $\text{\LaTeX}$  indexes, generation of song book indexes requires two passes of document compilation. During the first pass, data files are generated with song titles, authors, and scripture references. An external program is then used to produce  $\text{\LaTeX}$  source files from those data files. During the second pass of document compilation, those source files are imported to typeset all the indexes and display them in the document.

Internally, this package code uses a *four* step process to move the index data from the source `.tex` file to the `.sxd` data files.

1. While the current song box is in the midst of construction, the data is stored in a box of non-immediate write whatsit nodes.
2. The whatsits are migrated out to the top of the song box when it is finalized at `\endsong`.
3. When the song box is shipped out to the output file,  $\text{\TeX}$  expands the whatsits, causing the data to be written to the `.sxc` auxiliary file.
4. At the `\end{document}` line, the `.sxc` is processed multiple times—once for each index—to split the data into the respective `.sxd` files.

The first and second steps allow index references to point to the beginning of the song no matter where the indexing commands appear within the song. The third step allows  $\TeX$  to drop index entries that refer to songs that do not actually appear in the output (e.g., because of `\includeonlysongs`). It also allows index entries to refer to information that is only decided at shipout time, such as page numbers. The fourth step allows all indexing to be accomplished with at most one write register.  $\LaTeX$  provides extremely few write registers, so using as few as possible is essential for supporting books with many indexes.

`\SB@indexlist` This macro records the comma-separated list of the identifiers of indexes associated with the current book section.

```
2922 \newcommand\SB@indexlist{}
```

`\SB@allindexes` This macro records a comma-separated list of all the index identifiers for the entire document.

```
2923 \newcommand\SB@allindexes{}
```

```
2924 \let\SB@allindexes\@empty
```

`\SB@out` The `\SB@out` control sequence is reserved for the write register allocated by the package code, if one is needed. (It is allocated at the first index declaration.)

```
2925 \newcommand\SB@out{}
```

```
2926 \let\SB@out\relax
```

`\SB@newindex` Initialize a new title, author, or scripture index.

```
2927 \newcommand\SB@newindex[4]{%
```

```
2928 \expandafter\newcommand\csname SB@idxfilename@#3\endcsname{#4}%
```

```
2929 \expandafter\newcommand\csname SB@idxsel@#3\endcsname[3]{###1}%
```

```
2930 \expandafter\newcommand\csname SB@idxref@#3\endcsname{\thesongnum}%
```

```
2931 \xdef\SB@allindexes{%
```

```
2932 \ifx\SB@allindexes\@empty\else\SB@allindexes,\fi#3%
```

```
2933 }%
```

```
2934 \if@files%
```

```
2935 \ifx\SB@out\relax%
```

```
2936 \newwrite\SB@out%
```

```
2937 \immediate\openout\SB@out=\jobname.sxc\relax%
```

```
2938 \fi%
```

```
2939 \immediate\write\SB@out{\noexpand\SB@iwrite{#3}{#2}}%
```

```
2940 \fi%
```

```
2941 }
```

`\newindex` Define a new title index. The first argument is an identifier for the index (used in constructing index-specific control sequence names). The second argument is a filename root; auxiliary file `\arg2.sxd` is where the index data is stored at the end of processing.

```
2942 \newcommand\newindex{\SB@newindex1{TITLE INDEX DATA FILE}}
```

```
2943 \@onlypreamble\newindex
```

`\newscripindex` Define a new scripture index. This is exactly like `\newindex` except that scripture references are added to the auxiliary file instead of titles.

```

2944 \newcommand\newscripindex{\SB@newindex2{SCRIPTURE INDEX DATA FILE}}
2945 \@onlypreamble\newscripindex

```

`\newauthorindex` Define a new author index. This is exactly like `\newindex` except that author info is added to the auxiliary file instead of titles.

```

2946 \newcommand\newauthorindex{\SB@newindex3{AUTHOR INDEX DATA FILE}}
2947 \@onlypreamble\newauthorindex

```

`\SB@cwwrite` Write index data to a Song index Combined (`.sxc`) auxiliary file. The first argument is the identifier for the index to which the data ultimately belongs. The second argument is the data itself. The write is non-immediate so that it is only output if its enclosing song is ultimately shipped to the output file.

```

2948 \newcommand\SB@cwwrite[2]{%
2949 \ifx\SB@out\relax\else%
2950 \protected@write\SB@out{}\protect\SB@iwrite{#1}{#2}}%
2951 \fi%
2952 }

```

`\SB@iwrite` The line contributed by `\SB@cwwrite` to the `.sxc` file is an `\SB@iwrite` macro that re-outputs the data to an appropriate `.sxd` file.

```

2953 \newcommand\SB@iwrite[2]{%
2954 \def\SB@tempii{#1}%
2955 \ifx\SB@temp\SB@tempii%
2956 \SB@toks{#2}%
2957 \immediate\write\SB@out{\the\SB@toks}%
2958 \fi%
2959 }

```

`\SB@uncombine` At the end of the document, the `.sxc` file can be processed multiple times to produce all the `.sxd` files without resorting to multiple write registers. Each pass activates the subset of the `\SB@iwrite` commands that apply to one index.

```

2960 \newcommand\SB@uncombine{%
2961 \ifx\SB@out\relax\else%
2962 \immediate\closeout\SB@out%
2963 \ifsongindexes%
2964 \@for\SB@temp:=\SB@allindexes\do{%
2965 \immediate\openout\SB@out=%
2966 \csname SB@idxfilename@\SB@temp\endcsname.sxd\relax%
2967 \begingroup\makeatletter\input{\jobname.sxc}\endgroup%
2968 \immediate\closeout\SB@out%
2969 }%
2970 \fi%
2971 \fi%
2972 }
2973 \AtEndDocument{\SB@uncombine}

```

`\SB@songwrites` The following box register stores index data until it can be migrated to the top of the song box currently under construction.

2974 `\newbox\SB@songwrites`

`\SB@addtoindex` Queue data  $\langle arg2 \rangle$  associated with the current song for eventual writing to the index whose identifier is given by  $\langle arg1 \rangle$ .

```

2975 \newcommand\SB@addtoindex[2]{%
2976 \protected@edef\SB@tempii{#2}%
2977 \ifx\SB@tempii\empty\else%
2978 \global\setbox\SB@songwrites\vbox{%
2979 \unvbox\SB@songwrites%
2980 \SB@cwrite{#1}{#2}%
2981 \SB@cwrite{#1}{\csname SB@idxref@#1\endcsname}%
2982 \SB@cwrite{#1}{song\theSB@songsnum-\thesongnum.%
2983 \ifnum\c@section=\z@1\else2\fi}%
2984 }%
2985 \fi%
2986 }
```

`\SB@addtoindexes` Add  $\langle arg1 \rangle$  to all title indexes,  $\langle arg2 \rangle$  to all scripture indexes, and  $\langle arg3 \rangle$  to all author indexes.

```

2987 \newcommand\SB@addtoindexes[3]{%
2988 \@for\SB@temp:=\SB@indexlist\do{%
2989 \SB@addtoindex\SB@temp%
2990 {\csname SB@idxsel@\SB@temp\endcsname{#1}{#2}{#3}}%
2991 }%
2992 }
```

`\SB@addtotitles` Add  $\langle arg1 \rangle$  to all title indexes, but leave other indexes unaffected.

```

2993 \newcommand\SB@addtotitles[1]{%
2994 \@for\SB@temp:=\SB@indexlist\do{%
2995 \csname SB@idxsel@\SB@temp\endcsname%
2996 {\SB@addtoindex\SB@temp{#1}}{}{}%
2997 }%
2998 }
```

`\SB@chkidxlst` Check the current list of indexes and flag an error if any are undefined.

```

2999 \newcommand\SB@chkidxlst{%
3000 \let\SB@temp\SB@indexlist%
3001 \let\SB@indexlist\empty%
3002 \@for\SB@tempii:=\SB@temp\do{%
3003 \@ifundefined{SB@idxsel@\SB@tempii}{\SB@errnoidx\SB@tempii}{%
3004 \ifx\SB@indexlist\empty%
3005 \SB@toks\expandafter{\SB@tempii}%
3006 \else%
3007 \SB@toks\expandafter\expandafter\expandafter{%
3008 \expandafter\SB@indexlist\expandafter,\SB@tempii}%
3009 \fi%
3010 \edef\SB@indexlist{\the\SB@toks}%

```

```

3011 }%
3012 }%
3013 }

\indexentry \SB@addtoindexes will be called automatically for each song in a section. How-
\SB@idxentry ver, \indexentry may be called by the user in order to add an alternative index
\SB@@idxentry entry for the given song. Usually this is done to index the song by its first line or
some other memorable line in a chorus or verse somewhere.

3014 \newcommand\indexentry{\@ifnextchar[{\SB@idxentry*}{\SB@@idxentry*}}
3015 \newcommand\SB@idxentry{}
3016 \def\SB@idxentry#1[#2]#3{%
3017 \def\SB@indexlist{#2}%
3018 \SB@chkidxlst%
3019 \SB@addtoindexes{#1#3}{#3}{#3}%
3020 }}
3021 \newcommand\SB@@idxentry[2]{\SB@addtotitles{#1#2}}

\indextitleentry \indextitleentry may be used to add an alternate title for the song to the index.
(The only difference between the effects of \indexentry and \indextitleentry
is that the latter are italicized in the rendered index and the former are not.)

3022 \newcommand\indextitleentry{%
3023 \@ifnextchar[{\SB@idxentry*}{\SB@@idxentry*}}%
3024 }

\indexsongsas The following macro allows the user to change how songs are indexed on the right
side of index entries. By default, the song's number is listed.

3025 \newcommand\indexsongsas[1]{%
3026 \@ifundefined{SB@idxref@#1}%
3027 {\SB@errnoidx{#1}\@gobble}%
3028 {\expandafter\renewcommand\csname SB@idxref@#1\endcsname}%
3029 }

\SB@percent Assign a literal % character to \SB@percent in order to output it to index.sxd
files.

3030 \newcommand\SB@percent{}
3031 {\catcode'\%=12\gdef\SB@percent{}}

\authseppword The songidx index-generation program understands several different directives
\authbyword that each dictate various aspects of how index entries are parsed, sorted, and
\authignoreword displayed. Such directives should typically appear at the start of the .sxd file just
\titlprefixword after the header line that identifies the type of index.

\SB@idxcmd 3032 \newcommand\SB@idxcmd[2]{%
3033 \ifx\SB@out\relax\else%
3034 \immediate\write\SB@out{%
3035 \noexpand\SB@iwrite{#1}{\noexpand\SB@percent#2}%
3036 }%
3037 \fi%
3038 }

```



```

3039 \newcommand\authseppword[1]{\SB@idxcmd\SB@authinit{sep #1}}
3040 \@onlypreamble\authseppword
3041 \newcommand\authbyword[1]{\SB@idxcmd\SB@authinit{after #1}}
3042 \@onlypreamble\authbyword
3043 \newcommand\authignoreword[1]{\SB@idxcmd\SB@authinit{ignore #1}}
3044 \@onlypreamble\authignoreword
3045 \newcommand\titleprefixword[1]{\SB@idxcmd\SB@titleinit{prefix #1}}
3046 \@onlypreamble\titleprefixword

```

`\SB@idxtitlebox` Define a box to hold the index title.

```

3047 \newbox\SB@idxtitlebox

```

`\SB@idxlineskip` Set the spacing between lines in an index.

```

3048 \newcommand\SB@idxlineskip[1]{%
3049 \vskip#1\p@\@plus#1\p@\@minus#1\p@%
3050 }

```

When rendering an index entry  $X \dots Y$  that is too long to fit on one physical line, we must break text  $X$  and/or  $Y$  up into multiple lines. Text  $X$  should be typeset as a left-justified paragraph with a right margin of about 2em; however, it's final line must not be so long that it cannot fit even the first item of list  $Y$ . Text  $Y$  should be typeset as a right-justified paragraph whose first line begins on the last line of  $X$ . However, breaking  $Y$  up the way paragraphs are normally broken up doesn't work well because that causes most of  $Y$  to be crammed into the first few lines, leaving the last line very short. This looks strange and is hard to read. It looks much better to instead break  $Y$  up in such a way that the portion of  $Y$  that is placed on each line is of approximately equal width (subject to the constraint that we don't want to introduce any more lines than are necessary). This makes it visually clear that all of these lines are associated with  $X$ . The following code performs the width computations that do this horizontal-balancing of text.

`\SB@ellipspread` Typeset an index entry of the form  $X \dots Y$ . In the common case, the entire entry fits on one line so we just typeset it in the usual way. If it doesn't fit on one line, we call `\SB@balancerows` for a more sophisticated treatment.

```

3051 \newcommand\SB@ellipspread[2]{%
3052 \begingroup%
3053 \SB@dimen\z@%
3054 \def\SB@temp{#1}%
3055 \SB@toks{#2}%
3056 \setbox\SB@box\hbox{%
3057 \SB@temp%
3058 \leaders\hbox to.5em{\hss.\hss}\hskip2em\@plus1fil%
3059 {\the\SB@toks}%
3060 }%
3061 \ifdim\wd\SB@box>\hsize%
3062 \SB@balancerows%
3063 \else%
3064 \hbox to\hsize{\unhbox\SB@box}\par%

```

```

3065 \fi%
3066 \endgroup%
3067 }

```

`\SB@balancerows` Typeset an index entry of the form  $X \dots Y$  that doesn't fit on one line, where  $X$  is the content of macro `\SB@temp` and  $Y$  is the content of token register `\SB@toks`.

First, we must pre-compute the width  $w_1$  of the final line of  $X$  when  $X$  is typeset as a left-justified paragraph, storing it in `\SB@dimenii`. This is necessary because in order to force  $\text{\TeX}$  to typeset the first line of  $Y$  at some chosen width  $w_2$ , we must insert leaders of width  $c - w_1 - w_2$  into the paragraph between  $X$  and  $Y$ , where  $c$  is the column width.

Computing this width  $w_1$  is a bit tricky. We must tell  $\text{\TeX}$  that the last line of  $X$  must not be so long that it does not even have room for the first item of  $Y$ . Thus, we must strip off the first item of  $Y$  and add it (or a non-breaking space of equivalent width) to the end of  $X$  to typeset the paragraph. Then we use `\lastbox` to pull off the final line and check its width.

```

3068 \newcommand\SB@balancerows{%
3069 \edef\SB@tempii{\the\SB@toks}%
3070 \setbox\SB@box\vbox{%
3071 \SB@toks\expandafter{\expandafter\\the\SB@toks\\}%
3072 \SB@lop\SB@toks\SB@toks%
3073 \settowidth\SB@dimen{\the\SB@toks}%
3074 \advance\SB@dimen-.5em%
3075 \leftskip.5cm%
3076 {\hbadness\@M\hfuzz\maxdimen%
3077 \hskip-.5cm\relax\SB@temp\unskip\nobreak%
3078 \hskip\SB@dimen\nobreak%
3079 \rightskip2em\@plus1fil\par}%
3080 \setbox\SB@box\lastbox%
3081 \setbox\SB@box\hbox{%
3082 \unhbox\SB@box%
3083 \unskip\unskip\unpenalty%
3084 \unpenalty\unskip\unpenalty%
3085 }%
3086 \expandafter%
3087 }%
3088 \expandafter\SB@dimenii\the\wd\SB@box\relax%

```

Next, compute the smallest width  $w_2$  such that the index entry text produced by `\SB@multiline` with `\SB@dimen= $w_2$`  has no more lines than with `\SB@dimen` set to the maximum available width for the right-hand side. This effectively horizontal-balances the right-hand side of the index entry text, making all lines of  $Y$  roughly equal in width without introducing any extra lines.

```

3089 \SB@dimen\hsize%
3090 \advance\SB@dimen-.5cm%
3091 \setbox\SB@box\vbox{%
3092 \SB@multiline{\hbadness\@M\hfuzz\maxdimen}%
3093 }%
3094 \SB@dimeniii.5\SB@dimen%

```

```

3095 \SB@dimeniv\SB@dimeniii%
3096 \loop%
3097 \SB@dimeniv.5\SB@dimeniv%
3098 \setbox\SB@boxii\vbox{%
3099 \SB@dimen\SB@dimeniii%
3100 \SB@multiline{\hbadness\@M\hfuzz\maxdimen}%
3101 }%
3102 \ifnum\SB@cnt<\@M%
3103 \ifdim\ht\SB@boxii>\ht\SB@box%
3104 \advance\SB@dimeniii\SB@dimeniv%
3105 \else%
3106 \SB@dimen\SB@dimeniii%
3107 \advance\SB@dimeniii-\SB@dimeniv%
3108 \fi%
3109 \else%
3110 \advance\SB@dimeniii\SB@dimeniv%
3111 \fi%
3112 \ifdim\SB@dimeniv>2\p@\repeat%
3113 \setbox\SB@box\box\voidb@x%
3114 \setbox\SB@boxii\box\voidb@x%

```

Finally, typeset the results based on the quantities computed above.

```

3115 \SB@multiline\relax%
3116 }

```

**\SB@multiline** Create a paragraph containing text  $X \dots Y$  where  $X$  is the content of `\SB@temp`,  $Y$  is the content of `\SB@tempii`, and  $Y$  is restricted to width `\SB@dimen` (but may span multiple lines of that width). Dimen register `\SB@dimenii` must be set with the expected width of the final line of  $X$ . The first argument contains any parameter definitions that should be in effect when  $X$  is processed.

Note that the expansion of `\SB@tempii`, which may contain `\SB@idxitemsep`, depends on `\SB@dimen`. Therefore, the redefinition of `\SB@dimen` at the start of this macro must not be removed!

```

3117 \newcommand\SB@multiline[1]{%
3118 \begingroup%
3119 \SB@dimen-\SB@dimen%
3120 \advance\SB@dimen\hsize%
3121 \SB@dimenii-\SB@dimenii%
3122 \advance\SB@dimenii\SB@dimen%
3123 {\hskip-.5cm\relax\SB@temp\unskip\nobreak%
3124 \SB@maxmin\SB@dimenii<\{1.5em\}%
3125 \leftskip.5cm\rightskip2em\@plus1fil%
3126 \interlinepenalty\@M%
3127 \leaders\hbox to.5em{\hss.\hss}\hskip\SB@dimenii\@plus1fill%
3128 \nobreak{\SB@tempii\kern-2em}%
3129 \par\global\SB@cnt\badness}%
3130 \endgroup%
3131 }%

```

`\SB@idxitemsep` If text  $Y$  in index entry  $X \dots Y$  has multiple items in a list, those items should be separated by `\` macros instead of by commas. The `\` macro will be assigned the definition of `\SB@idxitemsep` during index generation, which produces the comma along with the complex spacing required if  $Y$  ends up being broken into multiple lines. In particular, it forces each wrapped line of  $Y$  to be right-justified with left margin at least `\SB@dimen`.

```
3132 \newcommand\SB@idxitemsep{%
3133 ,\kern-2em\penalty-8\hskip2.33em\@minus.11em%
3134 \hskip-\SB@dimen\@plus-1fill%
3135 \vadjust{}\nobreak%
3136 \hskip\SB@dimen\@plus1fill\relax%
3137 }
```

The following set of macros and environments are intended for use in the `.sbx` files that are automatically generated by an index-generating program; they shouldn't normally appear in the user's `.tex` or `.sbd` files directly. However, they are named as exported macros (no `@` symbols) since they are used outside the package code and are therefore not strictly internal.

`idxblock` Some indexes are divided into blocks (e.g., one for each letter of the alphabet or one for each book of the bible). Each such block should be enclosed between `\begin{idxblock}{X}` and `\end{idxblock}` lines, where  $X$  is the title of the block. The actual definition of the `idxblock` environment is set within the initialization code for each type of index (below).

```
3138 \newenvironment{idxblock}[1]{}{}
```

`\idxentry` Within each `idxblock` environment there should be a series of `\idxentry` and/or `\idxaltentry` macros, one for each line of the index. Again, the exact definitions of these macros will vary between index types.

```
3139 \newcommand\idxentry[2]{}
3140 \newcommand\idxaltentry[2]{}

```

`SB@lgidx` Some indexes actually have two definitions for each `idxblock` environment—one for use when there are few enough entries to permit a small style index, and another for use in a large style index. These macros will be redefined appropriately within the initialization code for each type of index.

```
3141 \newenvironment{SB@lgidx}[1]{}{ }
3142 \newenvironment{SB@smidx}[1]{}{ }
```

`\SB@idxsetup` Set various parameters for a multicolumn index environment.

```
3143 \newcommand\SB@idxsetup[1]{%
3144 \hsize\SB@colwidth%
3145 \parskip\z@skip\parfillskip\z@skip\parindent\z@%
3146 \baselineskip\fontsize\p@\@plus\p@\@minus\p@%
3147 \lineskiplimit\z@\lineskip\p@\@plus\p@\@minus\p@%
3148 \hyphenpenalty\@M\exhyphenpenalty\@M%
3149 }
```

`\SB@makeidxcolumn` Break off enough material from `\SB@box` to create one column of the index.

```

3150 \newcommand\SB@makeidxcolumn[1]{%
3151 \ifdim\ht\SB@box=\z@%
3152 \hskip\hsize\relax%
3153 \else%
3154 \splittopskip\z@skip\splitmaxdepth\maxdepth%
3155 \vsplit\SB@box to\SB@dimen%
3156 \global\setbox\SB@box\vbox{%
3157 \SB@idxsetup{#1}%
3158 \splitbotmark%
3159 \unvbox\SB@box%
3160 }%
3161 \fi%
3162 }
```

`\SB@oneidxpage` Construct one full page of the index. The definition of `\SB@oneidxpage` is generated dynamically based on the type of index and number of columns.

```

3163 \newcommand\SB@oneidxpage{}
```

`\SB@displayindex` Create an index with title  $\langle arg2 \rangle$  and with  $\langle arg1 \rangle$  columns (must be a literal constant). Input the index contents from external file  $\langle arg3 \rangle$ , which is expected to be a  $\text{\TeX}$  file.

```

3164 \newcommand\SB@displayindex[3]{%
3165 \ifsongindexes\beginngroup%
3166 \SB@colwidth\hsize%
3167 \advance\SB@colwidth-#1\columnsep%
3168 \advance\SB@colwidth\columnsep%
3169 \divide\SB@colwidth#1%
3170 \setbox\SB@idxtitlebox\vbox{%
3171 \let\SB@temp\songsection%
3172 \ifx\chapter\undefined\else%
3173 \ifx\chapter\relax\else%
3174 \let\SB@temp\songchapter%
3175 \fi%
3176 \fi%
3177 \SB@temp{#2}%
3178 }%
```

The `.sbx` index file might not exist (e.g., if this is the first pass through the  $\text{\TeX}$  compiler). If it exists, first try typesetting its content as a small index (one column, centered, with no divisions).

```

3179 \IfFileExists{\csname SB@idxfilename@#3\endcsname.sbx}{%
3180 \ifx\hyperlink\undefined\let\hyperlink\@secondoftwo\fi%
3181 \ifx\hyperlink\relax\let\hyperlink\@secondoftwo\fi%
3182 \ifsepindexes%
3183 \global\setbox\SB@box\vbox{%
3184 \null%
3185 \vfil%
3186 \unvcopy\SB@idxtitlebox%
```

```

3187 \vskip.5in\@minus.3in\relax%
3188 \hbox to\hsize{%
3189 \hfil%
3190 \vbox{%
3191 \hsize\SB@colwidth%
3192 \renewenvironment{idxblock}[1]%
3193 {\begin{SB@smidx}{####1}}{\end{SB@smidx}}%
3194 \let\SB@idxitemsep%
3195 \input{\csname SB@idxfilename@#3\endcsname.sbx}%
3196 }%
3197 \hfil%
3198 }%
3199 \vskip\z@\@plus2fil\relax%
3200 }%

```

Test whether the resulting small index fits within one page. If not, re-typeset it as a large index.

```

3201 {\vbadness\@M\vfuZZ\maxdimen%
3202 \splitmaxdepth\maxdepth\splittopskip\z@skip%
3203 \global\setbox\SB@boxii\vsplit\SB@box to\textheight}%
3204 \ifvoid\SB@box%
3205 \box\SB@boxii%
3206 \else%
3207 \SB@lgindex{#1}{#3}%
3208 \fi%
3209 \else%
3210 \SB@lgindex{#1}{#3}%
3211 \fi%
3212 }%

```

If the .sbx file doesn't exist, then instead typeset a page with a message on it indicating that the document must be compiled a second time in order to generate the index.

```

3213 {%
3214 \ifsepindexes%
3215 \vbox to\textheight{%
3216 \vfil%
3217 \unvbox\SB@idxtitlebox%
3218 \vskip1em\relax%
3219 \hbox to\hsize{\hfil[Index not yet generated.]\hfil}%
3220 \vskip\z@\@plus2fil\relax%
3221 }%
3222 \else%
3223 \unvbox\SB@idxtitlebox%
3224 \hbox to\hsize{\hfil[Index not yet generated.]\hfil}%
3225 \fi%
3226 }%
3227 \ifsepindexes\clearpage\fi%
3228 \endgroup\fi%
3229 }

```

`\SB@lgindex` Typeset a large-style index. We begin by typesetting the entire index into a box.

```

3230 \newcommand\SB@lgindex[2]{%
3231 \global\setbox\SB@box\vbox{%
3232 \renewenvironment{idxblock}[1]%
3233 {\begin{SB@lgidx}{##1}}{\end{SB@lgidx}}%
3234 \let\SB@idxitemsep%
3235 \SB@idxsetup{#1}%
3236 \input{\csname SB@idxfilename@#2\endcsname.sbx}%
3237 \unskip%
3238 }%

```

Next, we split the box into columns and pages until the last page is reached.

```

3239 \SB@toks{\SB@makeidxcolumn{#1}}%
3240 \SB@cnt#1\relax%
3241 \loop\ifnum\SB@cnt>\@ne%
3242 \SB@toks\expandafter{\the\SB@toks%
3243 \kern\columnsep\SB@makeidxcolumn{#1}}%
3244 \advance\SB@cnt\m@ne%
3245 \repeat%
3246 \edef\SB@oneidxpage{\the\SB@toks}%
3247 \unvbox\SB@idxtitlebox%
3248 \vskip.2in\relax%
3249 \nointerlineskip%
3250 \null%
3251 \nointerlineskip%
3252 \SB@cnt\vbaddness\vbaddness\@M%
3253 \SB@dimenii\vfuzz\vfuzz\maxdimen%
3254 \loop%
3255 \SB@dimen\textheight%
3256 \ifinner\else\kern\z@advance\SB@dimen-\pagetotal\fi%
3257 \global\setbox\SB@boxii\copy\SB@box%
3258 \global\setbox\SB@boxiii\hbox{\SB@oneidxpage}%
3259 \ifdim\ht\SB@box>\z@%
3260 \box\SB@boxiii%
3261 \vfil\break%
3262 \repeat%

```

The final page of the index should have all equal-height columns instead of a few full columns followed by some short or empty columns at the end. To achieve this, we re-typeset the final page, trying different column heights until we find one that causes the material to span an equal percentage of all the columns on the page.

```

3263 \SB@dimenii\ht\SB@boxii%
3264 \divide\SB@dimenii#1\relax%
3265 \SB@maxmin\SB@dimen>\SB@dimenii%
3266 \loop%
3267 \global\setbox\SB@box\copy\SB@boxii%
3268 \global\setbox\SB@boxiii\hbox{\SB@oneidxpage}%
3269 \ifdim\ht\SB@box>\z@%
3270 \advance\SB@dimen\p@%
3271 \repeat%

```

```

3272 \box\SB@boxiii%
3273 \global\setbox\SB@boxii\box\voidb@x%
3274 \vbadness\SB@cnt\vfuzz\SB@dimenii%
3275 }

```

**\showindex** Create an index with title  $\langle arg2 \rangle$  based on the data associated with index identifier  $\langle arg3 \rangle$  (which was passed to **\newindex**). Optional argument  $\langle arg1 \rangle$  specifies the number of columns. This macro calls the appropriate index-creation macro depending on the type of index that  $\langle arg3 \rangle$  was declared to be.

```

3276 \newcommand\showindex[3][0]{%
3277 \ifundefined{SB@idxsel@#3}{\SB@errnoidx{#3}}{%
3278 \expandafter\let\expandafter\SB@temp\csname SB@idxsel@#3\endcsname%
3279 \SB@cnt#1\relax%
3280 \ifnum\SB@cnt<\@ne\SB@cnt\SB@temp232\relax\fi%
3281 \expandafter\SB@temp%
3282 \expandafter\SB@maketitleindex%
3283 \expandafter\SB@makescripindex%
3284 \expandafter\SB@makeauthorindex%
3285 \expandafter{\the\SB@cnt}%
3286 {#2}{#3}%
3287 }%
3288 }

```

**\SB@maketitleindex** Create a song title index.  $\langle arg1 \rangle$  is a column count,  $\langle arg2 \rangle$  is the title, and  $\langle arg3 \rangle$  is the index identifier (which was passed to **\newindex**).

```

3289 \newcommand\SB@maketitleindex{%
3290 \renewenvironment{SB@lgidx}[1]{
3291 \hbox{\SB@colorbox\idxbgcolor{\vbox{%
3292 \hbox to\idxheadwidth{\{\idxheadfont\relax##1\}\hfil}%
3293 }}}%
3294 \nobreak\vskip3\p@ \@plus2\p@ \@minus2\p@ \nointerlineskip%
3295 }{\penalty-50\vskip5\p@ \@plus5\p@ \@minus4\p@}%
3296 \renewenvironment{SB@smidx}[1]{\{\}%
3297 \renewcommand\idxentry[2]{%
3298 \SB@ellipsread{\idxtitlefont\relax\ignorespaces##1\unskip}%
3299 {\{\idxrefsfont\relax##2\}}%
3300 }%
3301 \renewcommand\idxaltentry[2]{%
3302 \SB@ellipsread{\idxlyricfont\relax\ignorespaces##1\unskip}%
3303 {\{\idxrefsfont\relax##2\}}%
3304 }%
3305 \SB@displayindex%
3306 }

```

**\SB@idxcolhead** In a scripture index, this macro remembers the current book of the bible we're in so that new columns can be headed with "Bookname (continued)".

```

3307 \newcommand\SB@idxcolhead{}

```



`\SB@idxheadsep` Add vertical space following the header line that begins (or continues) a section of a scripture index.

```
3308 \newcommand\SB@idxheadsep{%
3309 \SB@dimen4\p@%
3310 \advance\SB@dimen-\prevdepth%
3311 \SB@maxmin\SB@dimen<\z@%
3312 \SB@dimenii\SB@dimen%
3313 \SB@maxmin\SB@dimenii>\p@%
3314 \vskip\SB@dimen\@plus\p@\@minus\SB@dimenii%
3315 }
```

`\SB@idxcont` Typeset the “Bookname (continued)” line that continues a scripture index section when it spans a column break.

```
3316 \newcommand\SB@idxcont[1]{%
3317 \hbox to\hsize{\idxcont{#1}}\hfil}%
3318 \nobreak%
3319 \SB@idxheadsep\nointerlineskip%
3320 }
```

`\SB@makescripindex` Create a scripture index.  $\langle arg1 \rangle$  is a column count,  $\langle arg1 \rangle$  is the title, and  $\langle arg2 \rangle$  is the index identifier (which was passed to `\newscripindex`).

```
3321 \newcommand\SB@makescripindex{%
3322 \renewenvironment{SB@lidx}[1]{%
3323 \gdef\SB@idxcolhead{##1}%
3324 \hbox to\hsize{\idxbook{##1}}\hfil}%
3325 \nobreak%
3326 \SB@idxheadsep\nointerlineskip%
3327 }{%
3328 \mark{\noexpand\relax}%
3329 \penalty-20\vskip3\p@\@plus3\p@\relax%
3330 }%
3331 \renewenvironment{SB@smidx}[1]
3332 {\begin{SB@lidx}{##1}}{\end{SB@lidx}}%
3333 \renewcommand\idxentry[2]{%
3334 \SB@ellipsread{\hskip.25cm\idxscripfont\relax##1}%
3335 {\idxrefsfont\relax##2}}%
3336 \SB@toks\expandafter{\SB@idxcolhead}%
3337 \mark{\noexpand\SB@idxcont{the\SB@toks}}%
3338 }%
3339 \renewcommand\idxaltentry[2]{\SB@erridx{a scripture}}%
3340 \SB@displayindex%
3341 }
```

`\SB@makeauthorindex` Create an author index.  $\langle arg1 \rangle$  is a column count,  $\langle arg2 \rangle$  is the title, and  $\langle arg2 \rangle$  is the index identifier (which was passed to `\newauthindex`).

```
3342 \newcommand\SB@makeauthorindex{%
3343 \renewenvironment{SB@lidx}[1]{\relax}%
3344 \renewenvironment{SB@smidx}[1]{\relax}%
3345 \renewcommand\idxentry[2]{\relax}}
```

```

3346 \SB@ellipsread{{\idxauthfont\relax\sffcode'\@m##1}}%
3347 {{\idxrefsfont##2}}}%
3348 }%
3349 \renewcommand\idxaltentry[2]{\SB@erridx{an author}}%
3350 \SB@displayindex%
3351 }

```

## 16.16 Error Messages

We break error messages out into separate macros here in order to reduce the length (in tokens) of the more frequently used macros that do actual work. This can result in a small speed improvement on slower machines.

```

\SB@Error All errors and warnings will be reported as coming from package "songs".
\SB@Warn 3352 \newcommand\SB@Error{\PackageError{songs}}
3353 \newcommand\SB@Warn{\PackageWarning{songs}}

\SB@errspos
3354 \newcommand\SB@errspos{%
3355 \SB@Error{Illegal \protect\songspos\space argument}{The argue%
3356 nt to \protect\songspos\space must be a number from 0 to 3.}%
3357 }

\SB@errnse
3358 \newcommand\SB@errnse{%
3359 \SB@Error{Nested songs environments are not supported}{End th%
3360 e previous songs environment before beginning the next one.}%
3361 }

\SB@errpl
3362 \newcommand\SB@errpl{%
3363 \SB@Error{\protect\includeonlysongs\space not permitted with%
3364 in a songs environment}{\protect\includeonlysongs\space can o%
3365 nly be used in the document preamble or between songs environ%
3366 ments in the document body.}%
3367 }

\SB@errrtopt
3368 \newcommand\SB@errrtopt{%
3369 \SB@Error{Cannot display chords in a rawtext dump}{You have u%
3370 sed the rawtext option in the \protect\usepackage\space lin%
3371 e and have either used the chorded option as well or have use%
3372 d the \protect\chordson\space macro subsequently.}%
3373 }

\SB@warnrc
3374 \newcommand\SB@warnrc{%
3375 \SB@Warn{The \protect\repchoruses\space feature will not wor%
3376 k when the number of columns is set to zero}%
3377 }

```

\SB@errboo

```
3378 \newcommand\SB@errboo{%
3379 \SB@Error{Encountered \protect\beginsong\space without seein%
3380 g an \protect\endsong\space for the previous song}%
3381 {Song \thesongnum\space might be missing a%
3382 n \protect\endsong\space line.}%
3383 }
```

\SB@errbor

```
3384 \newcommand\SB@errbor{%
3385 \SB@Error{Encountered \protect\beginsong\space without seein%
3386 g an \protect\endscripture\space for the preceding scriptur%
3387 e quotation}{A scripture quotation appearing after son%
3388 g \thesongnum\space might be missing a%
3389 n \protect\endscripture\space line.}%
3390 }
```

\SB@erreov

```
3391 \newcommand\SB@erreov{%
3392 \SB@Error{Encountered \protect\endsong\space without seein%
3393 g an \protect\endverse\space for the preceding verse}{Son%
3394 g \thesongnum\space has a \protect\beginverse\space%
3395 line with no matching \protect\endverse\space line.}%
3396 }
```

\SB@erreoc

```
3397 \newcommand\SB@erreoc{%
3398 \SB@Error{Encountered \protect\endsong\space without seein%
3399 g an \protect\endchorus\space for the preceding chorus}{Son%
3400 g \thesongnum\space has a \protect\beginchorus\space%
3401 line with no matching \protect\endchorus\space line.}%
3402 }
```

\SB@erreor

```
3403 \newcommand\SB@erreor{%
3404 \SB@Error{Encountered \protect\endsong\space without seein%
3405 g an \protect\endscripture for the preceding scripture quot%
3406 e}{A scripture quote appearing before song \thesongnum\space%
3407 ended with \protect\endsong\space instead of wit%
3408 h \protect\endscripture.}%
3409 }
```

\SB@erreot

```
3410 \newcommand\SB@erreot{%
3411 \SB@Error{Encountered \protect\endsong\space with no matchin%
3412 g \protect\beginsong}{Before song \thesongnum\space there wa%
3413 s an \protect\endsong\space with no matchin%
3414 g \protect\beginsong.}%
3415 }
```

\SB@errbv

```
3416 \newcommand\SB@errbv{%
3417 \SB@Error{Encountered \protect\beginverse\space without seein%
3418 g an \protect\endverse\space for the preceding verse}{Son%
3419 g \thesongnum\space might have a verse that has n%
3420 o \protect\endendverse\space line.}%
3421 }
```

\SB@errbvc

```
3422 \newcommand\SB@errbvc{%
3423 \SB@Error{Encountered \protect\beginverse\space without seein%
3424 g an \protect\endchorus\space for the preceding chorus}{Son%
3425 g \thesongnum\space might have a chorus that has n%
3426 o \protect\endchorus\space line.}%
3427 }
```

\SB@errbvt

```
3428 \newcommand\SB@errbvt{%
3429 \SB@Error{Encountered \protect\beginverse\space without firs%
3430 t seeing a \protect\beginsong\space line}{Before son%
3431 g \thesongnum, there is a \protect\beginverse\space line no%
3432 t contained in any song.}%
3433 }
```

\SB@errevc

```
3434 \newcommand\SB@errevc{%
3435 \SB@Error{Encountered \protect\endverse\space while process%
3436 ing a chorus}{Song \thesongnum\space might hav%
3437 e a \protect\beginchorus\space concluded by a%
3438 n \protect\endverse\space instead of an \protect\endchorus.}%
3439 }
```

\SB@errevo

```
3440 \newcommand\SB@errevo{%
3441 \SB@Error{Encountered \protect\endverse\space without firs%
3442 t seeing a \protect\beginverse}{Song \thesongnum\space m%
3443 ight have an \protect\endverse\space with no matchin%
3444 g \protect\beginverse.}%
3445 }
```

\SB@errevt

```
3446 \newcommand\SB@errevt{%
3447 \SB@Error{Encountered an \protect\endverse\space outside o%
3448 f any song}{Before song \thesongnum, there is a%
3449 n \protect\endverse\space line not preceded b%
3450 y a \protect\beginsong\space line.}%
3451 }
```

```

\SB@erretex
3452 \newcommand\SB@erretex{%
3453 \SB@Error{The \protect\repchoruses\space feature requires e-%
3454 TeX compatibility}{Your version of LaTeX2e does not appear t%
3455 o be e-TeX compatible. Find a distribution that includes e-T%
3456 eX support in order to use this feature.}%
3457 }

\SB@errbcv
3458 \newcommand\SB@errbcv{%
3459 \SB@Error{Encountered \protect\beginchorus\space without see%
3460 ing an \protect\endverse\space for the preceding verse}{Son%
3461 g \thesongnum\space might hav%
3462 e a \protect\beginverse\space with no match%
3463 ing \protect\endverse.}%
3464 }

\SB@errbcc
3465 \newcommand\SB@errbcc{%
3466 \SB@Error{Encountered \protect\beginchorus\space without see%
3467 ing an \protect\endchorus\space for the preceding chorus}%
3468 {Song \thesongnum\space might have a \protect\beginchorus%
3469 \space with no matching \protect\endchorus.}%
3470 }

\SB@errbct
3471 \newcommand\SB@errbct{%
3472 \SB@Error{Encountered \protect\beginchorus\space without see%
3473 ing a \protect\beginsong\space line first}{After son%
3474 g \thesongnum\space there is a \protect\beginchorus\space%
3475 line outside of any song.}%
3476 }

\SB@errecv
3477 \newcommand\SB@errecv{%
3478 \SB@Error{Encountered an \protect\endchorus\space while proc%
3479 essing a verse}{Song \thesongnum\space might hav%
3480 e a \protect\beginverse\space concluded by \protect\endchorus%
3481 \space instead of \protect\endverse.}%
3482 }

\SB@erreco
3483 \newcommand\SB@erreco{%
3484 \SB@Error{Encountered \protect\endchorus\space without firs%
3485 t seeing a \protect\beginchorus}{Song \thesongnum\space m%
3486 ight have an \protect\endchorus\space with no match%
3487 ing \protect\beginchorus.}%
3488 }

```

\SB@errect

```
3489 \newcommand\SB@errect{%
3490 \SB@Error{Encountered an \protect\endchorus\space outside o%
3491 f any song}{Before song \thesongnum, there is a%
3492 n \protect\endchorus\space line not preceded b%
3493 y a \protect\beginsong\space line.}%
3494 }
```

\SB@errbro

```
3495 \newcommand\SB@errbro{%
3496 \SB@Error{Missing \protect\endsong}%
3497 {Nested song and intersong environments are not supported%
3498 . Song \thesongnum\space might be missing a%
3499 n \protect\endsong\space line.}%
3500 }
```

\SB@errbrr

```
3501 \newcommand\SB@errbrr{%
3502 \SB@Error{Nested intersong environments are not supported}%
3503 {A scripture quote or other intersong environment before s%
3504 ong \thesongnum\space is missing its ending line.}%
3505 }
```

\SB@errero

```
3506 \newcommand\SB@errero{%
3507 \SB@Error{Encountered an \protect\endscripture\space whil%
3508 e processing a song}{Song \thesongnum\space ends wit%
3509 h \protect\endscripture\space when it should end wit%
3510 h \protect\endsong.}%
3511 }
```

\SB@errert

```
3512 \newcommand\SB@errert{%
3513 \SB@Error{Encountered an \protect\endscripture\space with%
3514 out first seeing a \protect\beginscripture}{Before son%
3515 g \thesongnum, there is an \protect\endscripture\space w%
3516 ith no matching \protect\beginscripture.}%
3517 }
```

\SB@errscrip

```
3518 \newcommand\SB@errscrip[1]{%
3519 \SB@Error{Encountered a \protect#1\space outside a scriptu%
3520 re quote}{\protect#1\space can only appear betwee%
3521 n \protect\beginscripture\space an%
3522 d \protect\endscripture\space lines.}%
3523 }
```

```

\SB@errchord
3524 \newcommand\SB@errchord{%
3525 \SB@Error{Song \thesongnum\space seems to have chord%
3526 s that appear outside of any verse or chorus}{All chords a%
3527 nd lyrics should appear between \protect\beginverse\space%
3528 and \protect\endverse, or between \protect\beginchorus\space%
3529 and \protect\endchorus.}%
3530 }

\SB@errreplay
3531 \newcommand\SB@errreplay{%
3532 \SB@Error{Replayed chord has no matching chord}{Son%
3533 g \thesongnum\space uses \protect^ more times than the%
3534 re are chords in the previously memorized verse.}%
3535 }

\SB@errreg
3536 \newcommand\SB@errreg[1]{%
3537 \SB@Error{Unknown chord-replay register name: #1}{Chord-re%
3538 play registers must be declared with \protect\newchords.}%
3539 }

\SB@errdup
3540 \newcommand\SB@errdup[1]{%
3541 \SB@Error{Duplicate definition of chord-replay register%
3542 : #1}{\protect\newchords\space was used to declare the sa%
3543 me chord-replay register twice.}%
3544 }

\SB@errmbar
3545 \newcommand\SB@errmbar{%
3546 \SB@Error{Song \thesongnum\space seems to have measur%
3547 e bars that appear outside of any verse or chorus}{All mea%
3548 sure bars (produced with \protect\mbar\space or |) must ap%
3549 pear between \protect\beginverse\space an%
3550 d \protect\endverse, or between \protect\beginchorus\space%
3551 and \protect\endchorus.}%
3552 }

\SB@errtab
3553 \newcommand\SB@errtab{%
3554 \SB@Error{Invalid argument to \protect\gtab\space macro. R%
3555 eplacing it with \protect\0.}{Valid arguments consist onl%
3556 y of: X, 0, 0, 1, 2, 3, or 4.}%
3557 }

\SB@errnoidx
3558 \newcommand\SB@errnoidx[1]{%
3559 \SB@Error{Unknown index identifier: #1}{This index identifie%
3560 r was never declared using \protect\newindex.}%
3561 }

```

\SB@erridx

```
3562 \newcommand\SB@erridx[1]{%
3563 \SB@Error{\protect\idxaltentry\space not allowed in #1 index}%
3564 {This error should not occur. The index generation routines ha%
3565 ve malfunctioned. Try deleting all temporary files and then re%
3566 compiling.}%
3567 }
```

## 16.17 Option Processing

\ifchorded Reserve conditionals for all of the various option settings. We wait to define these since if any are used earlier than this, it is an error in the package code, and we'd rather get an error than continue.

```
\ifmeasures 3568 \newif\ifchorded
\ifpartiallist 3569 \newif\iflyric\lyrictrue
\ifrepchorus 3570 \newif\ifslides
\iftranscapos 3571 \newif\ifmeasures
\ifnolyrics 3572 \newif\ifpartiallist
\ifrawtext 3573 \newif\ifrepchorus
\ifpdfindex 3574 \newif\iftranscapos
\ifsongindexes 3575 \newif\ifnolyrics
\ifsepindexes 3576 \newif\ifrawtext
\ifsepindexes 3577 \newif\ifpdfindex\pdfindextrue
\ifSB@colorboxes 3578 \newif\ifsongindexes\songindexestru
\ifSB@omitscrip 3579 \newif\ifsepindexes\sepindexestru
3580 \newif\ifSB@colorboxes\SB@colorboxestru
3581 \newif\ifSB@omitscrip
```

\nolyrics The \nolyrics macro is just shorthand for \nolyricstrue.

```
3582 \newcommand\nolyrics{}
3583 \let\nolyrics\nolyricstrue
```

Finally we're ready to process all of the package options. This is delayed until near the end because the option processing code needs to execute various macros found in the previous sections.

```
3584 \SB@chordson
3585 \ProcessOptions\relax
```

If we're not generating a pdf, then don't generate the pdf index.

```
3586 \ifSB@pdf\else\pdfindexfalse\fi
```

\SB@colorbox Include the colors package and define colors, if requested.

```
3587 \ifSB@colorboxes
3588 \RequirePackage{color}
3589 \definecolor{SongbookShade}{gray}{.80}
3590 \newcommand\SB@colorbox[2]{%
3591 \ifx\@empty#1%
3592 \vbox{%
3593 \kern3\p@%
```



```

3594 \hbox{\kern3\p@{#2}\kern3\p@}%
3595 \kern3\p@%
3596 }%
3597 \else%
3598 \colorbox{#1}{#2}%
3599 \fi%
3600 }
3601 \else
3602 \newcommand\SB@colorbox[2]{\vbox{%
3603 \kern3\p@%
3604 \hbox{\kern3\p@{#2}\kern3\p@}%
3605 \kern3\p@%
3606 }}
3607 \fi

```

## 16.18 Rawtext Mode

If generating raw text, most of what has been defined previously is ignored in favor of some very specialized macros that write all the song lyrics to a text file.

```

3608 \ifrawtext
3609 \newwrite\SB@txtout
3610 \immediate\openout\SB@txtout=\jobname.txt
3611 \newif\ifSB@doEOL
3612 {\catcode'\~M12 %
3613 \catcode'\~J12 %
3614 \gdef\SB@printEOL{\ifSB@doEOL~M~J\fi}}
3615 {\catcode'#12\gdef\SB@hash{#}}
3616 {\catcode'&12\gdef\SB@amp{&}}
3617 \renewcommand\SB@@@beginsong{%
3618 \begingroup%
3619 \def'{}\def\{\}\def\v{}\def\u{}\def={}\def\~{}\%
3620 \def\.\{}\def\H{}\def\~{}\def\"{}\def\t{}\%
3621 \def\copyright{(c)}%
3622 \let~\space%
3623 \let\par\SB@printEOL%
3624 \let\#\SB@hash%
3625 \let\&\SB@amp%
3626 \catcode'!9 %
3627 \catcode'*9 %
3628 \catcode'^9 %
3629 \def\[##1]{}%
3630 \resettitles%
3631 \immediate\write\SB@txtout{\thesongnum. \songtitle}%
3632 \nexttitle%
3633 \foreachtitle{\immediate\write\SB@txtout{(\songtitle)}}%
3634 \ifx\songauthors@empty\else%
3635 \immediate\write\SB@txtout{\songauthors}%
3636 \fi%
3637 \ifx\SB@rawrefs@empty\else%

```

```

3638 \immediate\write\SB@txtout{\SB@rawrefs}%
3639 \fi%
3640 \immediate\write\SB@txtout{}%
3641 \SB@doEOLfalse%
3642 \obeylines%
3643 }
3644 \renewcommand\SB@endsong{%
3645 \SB@doEOLtrue%
3646 \immediate\write\SB@txtout{\songcopyright\space%
3647 \songlicense\SB@printEOL}%
3648 \endgroup%
3649 \SB@insongfalse%
3650 \stepcounter{songnum}%
3651 }
3652 \def\SB@parsesrefs#1{\def\songrefs{#1}}
3653 \long\def\beginverse#1#2\endverse{%
3654 \SB@doEOLtrue\begingroup%
3655 \def\textnote##1{##1}%
3656 \def\SB@temp{#1}%
3657 \def\SB@star{*}%
3658 \ifx\SB@temp\SB@star%
3659 \immediate\write\SB@txtout{\@gobble#2}%
3660 \else%
3661 \immediate\write\SB@txtout{#2}%
3662 \fi%
3663 \endgroup\SB@doEOLfalse}
3664 \long\def\beginchorus#1\endchorus{%
3665 \SB@doEOLtrue\begingroup%
3666 \def\textnote##1{##1}%
3667 \immediate\write\SB@txtout{Chorus:#1}%
3668 \endgroup\SB@doEOLfalse}
3669 \long\def\beginscripture#1\endscripture{}
3670 \def\musicnote#1{}
3671 \def\textnote#1{%
3672 \SB@doEOLtrue%
3673 \immediate\write\SB@txtout{#1\SB@printEOL}%
3674 \SB@doEOLfalse}
3675 \def\brk{}
3676 \def\rep#1{(x#1)}
3677 \def\echo#1{(#1)}
3678 \def\mbar#1#2{}
3679 \def\lrep{}
3680 \def\rrep{}
3681 \def\nolyrics{}
3682 \renewcommand\memorize[1][{}]{ }
3683 \renewcommand\replay[1][{}]{ }
3684 \fi

```