

MusiX_{TEX}[©]

Using _{TEX} to write polyphonic or instrumental music

Version T.114 – June, 2006

Daniel TAUPIN

Laboratoire de Physique des Solides

(associé au CNRS)

bâtiment 510, Centre Universitaire, F-91405 ORSAY Cedex

Ross MITCHELL

CSIRO Division of Atmospheric Research,

Private Bag No.1, Mordialloc, Victoria 3195,

Australia

Andreas EGLER

(Ruhr-uni-Bochum)

Ursulastr. 32

D-44793 Bochum

*If you are not familiar with T_EX at all
I would recommend to find another software
package to do musical typesetting.
Setting up T_EX and MusiX_TE_X
on your machine and mastering it
is an awesome job which gobbles up
a lot of your time and disk space.*

But, once you master it...

Hans KUYKENS

MusiX_TE_X may be freely copied, duplicated and used in conformance to the GNU General Public License (Version 2, 1991, see included file `copying`)¹.

You may take it or parts of it to include in other packages, but no packages called MusiX_TE_X without specific suffix may be distributed under the name MusiX_TE_X if different from the original distribution (except obvious bug corrections).

Adaptations for specific implementations (e.g. fonts) should be provided as separate additional T_EX or L^AT_EX files which override original definitions.

¹Thanks to Free Software Foundation for advising us. See <http://www.gnu.org>

Preface to Version T.113

The main author of MusiX_{TEX}, Daniel Taupin, died all too early in a 2003 climbing accident. The MusiX_{TEX} community was shocked by this tragic and unexpected event. You may read some tributes to Daniel Taupin which are archived at the [Werner Icking Music Archive](#).

Now, almost two years after his death, we would like to help keep his excellent work alive and current by assembling a new release. This new version corrects various minor bugs without adding any new functionality. At the same time it incorporates directly into the distribution some additional packages:

- Type 1 (postscript) versions of all the MusiX_{TEX} fonts, created by Takanori Uchiyama;
- Postscript Slur Package K (Ver 0.92, 12 May 02) by Stanislav Kneifl;
- musixlyr (Ver 2.1c, 12 June 03) a MusiX_{TEX} extension package for lyrics handling, by Rainer Dunker.

We would like to thank the authors of these packages for generously agreeing to allow their contributions to be included in the main MusiX_{TEX} package.

We have left the remainder of this manual largely as Daniel Taupin left it, except we have updated various references and provided dynamic links to archived versions where possible.

This documentation is rather technical and is probably not the best way to begin typesetting music. If you are a beginner, you should visit the software section of the [Werner Icking Music Archive](#). In particular, we recommend [Cornelius Noack's tutorial](#). It contains helpful information for getting started with MusiX_{TEX}, as well as a tutorial for **PMX**, a preprocessor for MusiX_{TEX} with a much simpler input language, and a brief introduction to **M-Tx**, a preprocessor for **PMX** which eases the inclusion of lyrics.

In the name of the community, Olivier Vogel (oliviervogel@freesurf.ch), July 30, 2005

Preface to Version T.114

In order to expedite the release of version T.113 of MusiX_{TEX}, only the most essential modifications were made to the manual. With version T.114 the manual has been extensively edited from cover to cover. Some material has been rearranged, and references to Music_{TEX}, the immediate predecessor to MusiX_{TEX}, have been removed. The software itself remains very stable. Only the slightest changes have been made since version T.113.

It remains true that this is the definitive reference to all features of MusiX_{TEX}, but also that it is not the best place for a novice user to start. The [Werner Icking Music Archive](#) contains excellent and detailed instructions for installing MusiX_{TEX} and the strongly recommended preprocessors **PMX** (for instrumental music) and **M-Tx** (for vocal) under [Linux/UNIX](#) or [Windows 2000](#). Once the software is installed, most common music typesetting tasks can be accomplished entirely by using one of these preprocessors to generate the MusiX_{TEX} input file, relieving the user of learning any of the commands or syntax of MusiX_{TEX} itself. It is only for out-of-the-ordinary constructions that one must learn these details, so he may insert the necessary MusiX_{TEX} commands into the preprocessor's input file as so called inline _{TEX}. [Cornelius Noack's tutorial](#) is an important resource which, in addition to gently introducing the novice to **PMX** and **M-Tx**, gives further details on installing Postscript slur facilities.

Don Simons, Andre van Ryckeghem, Cornelius Noack, August 30, 2005

Contents

1	Introduction to MusiX_TE_X	1
1.1	Primary features of MusiX _T E _X	1
1.1.1	Music typesetting is two-dimensional	1
1.1.2	Horizontal spacing	2
1.1.3	Music tokens	2
1.1.4	Beams	3
1.1.5	Setting anything on the score	3
1.2	A simple example	3
1.3	The three pass system	4
1.3.1	External executable <code>musixflx</code>	7
1.3.2	Unrecorded spaces: the novice's bugaboo	7
1.4	Further highlights	8
1.4.1	Key signatures	8
1.4.2	Transposition	8
1.4.3	Extracting parts from a score	9
1.4.4	Staff and note sizes	9
1.4.5	Add-in macro libraries	9
1.5	Where to get the software and help using it	10
1.6	A very brief history of MusiX _T E _X	10
2	Elements of MusiX_TE_X	11
2.1	Setting up the input file	11
2.1.1	What makes a <code>T_EX</code> file a MusiX _T E _X file?	11
2.1.2	Cautions for the non <code>T_EX</code> pert	11
2.1.3	Usual setup commands	12
2.1.4	Groupings of instruments	13
2.2	Preparing to enter notes	15
2.2.1	After the setup, what next?	15
2.2.2	Horizontal spacing commands	15
2.2.3	Moving from one staff or instrument to another	17
2.3	Note pitch specification	17
2.4	Writing notes	17
2.4.1	Normal (unbeamed) spacing notes	17
2.4.2	Non-spacing note heads	18
2.4.3	Shifted non-spacing note heads	19
2.4.4	Non-spacing notes	19
2.4.5	Spacing note heads	19
2.4.6	Dotted notes	20
2.4.7	Sequences of equally spaced notes; collective coding	20
2.5	Beams	20
2.5.1	Starting a beam	20

2.5.2	Adding notes to a beam	21
2.5.3	Ending a beam	22
2.5.4	Changing multiplicity after the beam starts	22
2.5.5	Shorthand beam notations for repeated notes	24
2.5.6	Beams that cross line breaks	25
2.5.7	Beams with notes on several different staves	25
2.6	Rests	27
2.6.1	Ordinary rests	27
2.6.2	Raising rests	27
2.6.3	Bar centered rests	27
2.7	Skipping spaces and shifting symbols	28
2.8	Accidentals	29
2.9	Transposition and octavation	30
2.9.1	Logical transposition and octavation	30
2.9.2	Behavior of accidentals under logical transposition	30
2.9.3	Octavation lines	31
2.10	Slurs and ties	33
2.10.1	Font-based slurs	33
2.10.2	Type K postscript slurs	39
2.11	Bars Lines	42
2.11.1	Single, double, and invisible bar lines	42
2.11.2	Simple discontinuous bar lines	43
2.11.3	Elementary asynchronous bar lines	43
2.11.4	Dotted, dashed, and more general asynchronous and discontinuous bar lines	44
2.12	Bar numbering	46
2.12.1	Periodic bar numbering	46
2.12.2	System bar numbering	47
2.13	Managing the layout of your score	48
2.13.1	Line and page breaking	48
2.13.2	Page layout	48
2.13.3	Page numbering, headers and footers	48
2.13.4	Controlling the total number of systems and pages	49
2.14	Changing clefs, key signatures, and meters	49
2.14.1	Introduction	49
2.14.2	Key Signatures	50
2.14.3	Clefs	51
2.14.4	Meter changes	54
2.15	Repeats	55
2.15.1	First and second endings (Voltas)	55
2.15.2	Special symbols for repeating long sections	56
2.15.3	Repeating a single bar	57
2.16	Font selection and text placement	57
2.16.1	Predefined text fonts	57
2.16.2	User-defined text fonts	58
2.16.3	Text placement	59
2.16.4	Rehearsal marks	60
2.17	Miscellaneous other notations	60
2.17.1	Metronomic indications	60
2.17.2	Accents	60
2.17.3	Numbers and brackets for xtuplets	61
2.17.4	Ornaments	62

2.17.5	Alphabetic dynamic marks	64
2.17.6	Hairpins (crescendos and decrescendos)	64
2.17.7	Length of note stems	66
2.17.8	Brackets, parentheses, and oblique lines	66
2.17.9	Forcing activity at the beginning of systems	67
2.18	Smaller notes in normal-sized staves	68
2.18.1	Arbitrary sequences of notes	68
2.18.2	Grace notes	69
2.18.3	Ossia	69
2.19	Staff size	70
2.20	Layout parameters	72
2.20.1	List of layout parameters	72
2.20.2	A convenient macro for changing layout parameters in mid-score	73
2.20.3	Changing the number of lines per staff	73
2.20.4	Resetting normal layout parameters	73
2.21	Lyrics	74
2.21.1	Native lyrics method: placing single words	74
2.21.2	Musixlyr	75
2.21.3	Getting enough vertical space for lyrics	77
2.21.4	Fine tuning the placement of the lyrics	77
2.22	Embedding musical excerpts in text documents	79
2.22.1	Directly embedding excerpts in L ^A T _E X documents	79
2.22.2	Embedding musical excerpts as encapsulated postscript files	80
2.22.3	Issues concerning <code>\catcodes</code>	81
2.23	Extension Library	81
2.23.1	<code>curly</code>	81
2.23.2	<code>musixadd</code>	81
2.23.3	<code>musixbm</code>	81
2.23.4	<code>musixbbm</code>	81
2.23.5	<code>musixcho</code>	82
2.23.6	<code>musixcpt</code>	82
2.23.7	<code>musixdat</code>	82
2.23.8	<code>musixdbr</code>	82
2.23.9	<code>musixdia</code>	82
2.23.10	<code>musixeng</code>	83
2.23.11	<code>musixext</code>	84
2.23.12	<code>musixfll</code>	84
2.23.13	<code>musixgre</code>	84
2.23.14	<code>musixgui</code>	91
2.23.15	<code>musixlit</code>	92
2.23.16	<code>musixlyr</code>	93
2.23.17	<code>musixmad</code>	93
2.23.18	<code>musixper</code>	93
2.23.19	<code>musixpoi</code>	96
2.23.20	<code>musixps</code>	96
2.23.21	<code>musixstr</code>	96
2.23.22	<code>musixsty</code>	96
2.23.23	<code>musixtmr</code>	97
2.23.24	<code>musixtri</code>	97
2.23.25	<code>tuplet</code>	97

3	Acquiring, Installing, and Using MusiX_{TE}X	98
3.1	Where to get MusiX _{TE} X: the Werner Icking Archive	98
3.2	Installing MusiX _{TE} X	98
3.2.1	Installing on a UNIX system	99
3.2.2	Installing on a MS Windows system with Mik _{TE} X	102
3.3	Running Musix _{TE} X in a MS Windows system	105
3.3.1	Batch commands for making DVI and postscript	105
4	MusiX_{TE}X examples	107
4.1	Small examples	107
4.2	Full examples	107
4.2.1	Examples mentioned in the manual	107
4.2.2	Other examples, provided by the authors of MusiX _{TE} X	108
4.3	Compiling musixdoc.tex	108
5	Summary of denotations	109
	Index	110

Chapter 1

Introduction to MusiX_{TEX}

This chapter is not a tutorial on the use of MusiX_{TEX}, but instead serves as an overview of some of its capabilities, quirks, and history.

MusiX_{TEX} is a set of _{TEX} macros which enables the typesetting of music with systems of up to twelve staves. It requires as a prerequisite a working installation of _{TEX}¹. MusiX_{TEX} might be regarded as the digital equivalent of a box of type. It contains symbols for staves, notes, chords, beams, slurs and ornaments, ready to be arranged to form a sheet of music. But it must be told how to position those symbols on the page. This could be done by the typesetter himself, if he elects to proceed by entering MusiX_{TEX} commands manually into an input file. However most users will find it far less taxing to let such decisions be made largely by the preprocessor [PMX](#), which in addition uses a much simpler input language than MusiX_{TEX}.

Lyrics can also be handled by MusiX_{TEX}. There is a set of primitive commands for this which are described later. But there is also a far more adaptable set of macros contained in the extension file `musiclyr.tex`, and there is the preprocessor [M-Tx](#) which provides easy, transparent access to these macros.

Most users of _{TEX} are familiar with \LaTeX , a set of _{TEX} macros which eases document layout. In fact many may only use \LaTeX . Until recently, \LaTeX and MusiX_{TEX} coexisted only grudgingly, owing primarily to the limited availability of storage registers. But with modern versions of _{TEX} and with the use of $e\LaTeX$, only a modest increase in complexity is incurred with the addition of musical excerpts to a \LaTeX document. Still, fortunately, for typesetting a musical score there is rarely if ever any advantage to using \LaTeX . Only if one wanted to create a text document with embedded musical examples would there be much use for it. Even in that case there is a perfectly fine way to avoid using MusiX_{TEX} directly in the document file, namely, by using MusiX_{TEX} to create `eps` files for each of the examples, then embedding references to those in the file for the book. But for anyone who still wants to use both together, there is no better example than this manual, as generated with the files `musixdoc.tex` and `musixdoc.sty`. A few further details about such nonstandard applications are given in section [2.22.1](#).

1.1 Primary features of MusiX_{TEX}

1.1.1 Music typesetting is two-dimensional

Written music is not usually a linear sequence of symbols like a literary text. Rather, except for unaccompanied single-note instruments like clarinets, trumpets and human voices, it has the form of a two-dimensional matrix. Thus, a logical way of coding music consists in horizontally accumulating a set of *vertical combs* with *horizontal teeth* as depicted in Table [1.1](#). Accordingly,

¹See section [3](#) for guidance on installing _{TEX}.

note sequence one	note seq. four	note seq. seven	note seq. ten
note sequence two	note seq. five	note seq. eight	note seq. eleven
note sequence three	note seq. six	note seq. nine	note seq. twelve

Table 1.1: A logical way of coding music

in MusiX_{TEX} the fundamental macro used to represent one of those vertical combs (or one of the columns in Table 1.1) is of the form

```
\notes ... & ... & ... \enotes2
```

where the character `&` is used to separate the notes to be typeset on respective staves of the various instruments, starting from the bottom.

In the case of an instrument whose score has to be written with several staves, they are separated by the character `|`. Thus, a score written for a keyboard instrument and a single-line instrument (e.g., piano and violin) will be coded as follows:

```
\notes ... | ... & ... \enotes
```

for each column of simultaneous *groups of notes*. Each of those groups, represented by a single box in Table 1.1 and by a sequence of three dots in the the two example macros above, may contain not only chord notes to be played simultaneously, but short sequences of consecutive notes or chords. As we'll soon see, this implies the need for two fundamentally different kinds of elemental macros in MusiX_{TEX}, those that are automatically followed by some amount of space (called *spacing macros*, and those that are not. The former type, for example, would be used to represent all the notes and rests in a single-line score. The latter would be used for example for chord notes and ornaments.

1.1.2 Horizontal spacing

Deciding upon the proper horizontal spacing of notes is a very complicated matter that we will not address in any detail here. Obviously short-duration notes should be closer together than longer ones. Almost as obviously, the spacing cannot be linearly proportional to the duration; otherwise for example a whole note would occupy 32 times as much horizontal space as a thirty-second note. And in polyphonic scores the spacing in one staff is often influenced by the notes in another. This is a decision that the typesetter or preprocessor must make. Once the decision is made, MusiX_{TEX} can provide the desired spacing. The main mechanism is through a set of macros described in section 2.2.2. At this point we shall only mention that to control spacing, one of those macros will be selected to replace the symbol `\notes` in the two examples above, and it will imply that whenever a spacing macro is encountered within a group of notes, a certain specific amount of horizontal space will be inserted.

1.1.3 Music tokens

The tokens provided by MusiX_{TEX} include

- note symbols without stems;
- note symbols with stems, and flags for eighth notes and shorter;
- beam beginnings and endings;

²The abbreviation `\en` can be used in place of `\enotes`.

- beginnings and endings of ties and slurs;
- accidentals;
- ornaments: arpeggios, trills, mordents, turns, staccatos, pizzicatos, fermatas, etc.;
- bar lines;
- meters, key signatures, clefs.

Thus for example, `\wh a` produces a whole note at nominal frequency 222.5 Hz, `\wh h` produces one an octave higher, `\qu c` produces an up-stemmed quarter note C (250 Hz), and `\cl J` produces a down-stemmed C eighth note an octave lower.

To generate chords with solid note heads, the macro `\zq` can be used. It produces a solid note head at the specified pitch, the vertical position of which is memorized and recalled whenever the next stemmed note (possibly with a flag) is coded. The stem length is automatically adjusted to link all simultaneous notes. Thus, the C-major chord



is coded `\zq c\zq e\zq g\qu j` or more concisely, `\zq{ceg}\qu j`. Here the `u` in the spacing note macro `\qu` is what causes the upstem.

1.1.4 Beams

Each beam is generated by a pair of macros. The first defines the beginning horizontal position (implicitly the current position), altitude, direction (upper or lower), multiplicity (number of lateral bars), slope and reference number. This latter feature is needed so independent beams can overlap. The second macro of the pair specifies the termination location (again implicitly) and the reference number.

1.1.5 Setting anything on the score

A general macro (`\zcharnote`) provides a means of putting any sequence of symbols (possibly contained in an `\hbox{...}`) at any pitch of any staff of any instrument. This allows any symbol defined in a font (letters, math symbols, etc.) to be placed in the score at a position keyed to the music both in time (horizontally) and pitch (vertically) on the staff,

1.2 A simple example

Before going into more detail, we give below an example of the two first bars of the sonata in C-major KV545 by MOZART:

Musical notation for the first two bars of the sonata in C-major KV545 by Mozart. The notation is for the piano part, showing two staves (treble and bass clefs) with a brace on the left labeled 'Piano'. The time signature is 4/4. The first bar contains a half note C4 in the bass staff and a half note G4 in the treble staff. The second bar contains a quarter note C4 in the bass staff and a quarter note G4 in the treble staff, followed by a quarter note E4 in the treble staff and a quarter note C4 in the bass staff.

The coding is as follows:

```

\begin{music}
\parindent10mm
\instrumentnumber{1}      % a single instrument
\setname1{Piano}         % whose name is Piano
\setstaves1{2}           % with two staves
\generalmeter{\meterfrac44}% 4/4 meter chosen
\startextract            % starting real score
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\ql 1\sk\ql n\en
\bar
\Notes\ibu0f0\qb0{dgf}|\qlp i\en
\notes\tbu0\qb0g|\ibb1j3\qb1j\tbl1\qb1k\en
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
\endextract              % terminate excerpt
\end{music}

```

- `\ibu0f0` begins an upper beam, aligned on the f , reference number 0, slope 0
- `\tbu0` terminates this beam before writing the second g by means of `\qb0g`
- `\qb.` indicates a note belonging to a beam.
- `\sk` sets a space between the two quarters in the right hand, so that the second one is aligned with the third eighth in the left hand.
- `\qlp` is a dotted quarter note.
- `\ibb1j3` begins a double beam, aligned on the C (j at this pitch) with slope 15%.

1.3 The three pass system

$\text{T}_{\text{E}}\text{X}$'s line-breaking procedure implicitly assumes that a normal line of text will contain many words, so that inter-word glue need not stretch or shrink too much to justify the line. One might at first consider extending this to music, treating each bar like a word with no stretchable internal space. But typically this would lead to unsightly gaps before each bar rule, simply because the number of bars per line is normally many fewer than the number of words in a line of text. MusiX $\text{T}_{\text{E}}\text{X}$ needs a more sophisticated horizontal spacing algorithm than is used in $\text{T}_{\text{E}}\text{X}$.

To understand how MusiX $\text{T}_{\text{E}}\text{X}$ solves this problem, we have to recognize two different kinds of horizontal space, *hard* and *scalable*. Hard space is fixed and always represents the same physical distance. Examples of hard space are the widths of bar rules, clefs, and key signatures. Scalable space can be stretched as needed. It is what is normally used for the space after notes or rests. At the outset it is only defined in a relative sense. In other words, scalable spaces are defined as multiples of `\elemskip`, an initially undefined elemental spacing unit. For example, in **PMX** all sixteenth notes are typically assigned a scalable width of 1.41\elemskip . One main job of MusiX $\text{T}_{\text{E}}\text{X}$ is to compute the physical value of `\elemskip`, often expressed in points (72nds of an inch). The correct value is that which makes all the scalable space on a line just fill up what's not occupied by hard space. Obviously it may vary from line to line.

To this end a three pass system was developed. To start the first pass on the file `jobname.tex`, you would enter `tex jobname`. Information about each bar is written to an external file named `jobname.mx1`. This file begins with a header containing parameters such as line width and paragraph indentation. Then the hard and scalable space is listed for each bar.

The second pass, which is started with `musixflx jobname`, determines optimal values of the elemental spacing unit `\elemskip` for each line, so as to properly fill each line, and to spread the

piece nicely over an integral number of full lines. This routine was written in FORTRAN and now converted to C rather than T_EX, the main reason being the lack of an array handling capability in T_EX.

`musixflx` reads in the file `jobname.mx1`, and writes its output to `jobname.mx2`. The latter file contains a single entry for each line of music in the reformatted output. The key piece of information is the revised value of `\elemskip` for each line.

Next, the file is re-T_EX-ed. On this third pass, the `jobname.mx2` file is read in, and the information is used to physically define the final score and embed the page descriptions into a `dvi` file.

Here's an example: By assigning an arbitrary single value to `\elemskip`, after the first pass you might get the following output:



Note that the space after every quarter note is the same, and that lines are not justified. After running `musixflx` and T_EX-ing the second time you'll get:



Now M_us_iX_T_EX has determined a number of lines (which is different from the original number), the lines are justified, and if you look carefully you can see that the space after quarters in the first line is smaller than in the second. This example was coded as:

```
\hsize=100mm
\generalmeter{\meterfrac24}%
\parindent Opt
\generalsignature{-3}
\startpiece\bigaccid
\Notes\qu{ce}\en\bar
\Notes\qu{gh}\en\bar
\Notes\qu{=b}\en
\Notes\ds\cu g\en\bar
\Notes\qu{^f=f}\en\bar
\Notes\qu{=e}\itied0e\qu{_e}\en\bar
\Notes\ttie0\qqbu ed{_d}c\en\bar
\Notes\ibu0b{-2}\qb0{=b}\enotes
\notes\nbbu0\qb0{=a}\tqh0N\enotes
\Notes\Dqbu cf\en\bar
```

```
\Notes\upertext{\it tr}\qu e\upertext{\it tr}\qu d\en\bar
\Notes\qu c\qp\en\Endpiece
```

One benefit of using this 3-pass system is the very easy and fast alteration to the layout which is possible, especially of a long masterpiece, by changing only one parameter, namely `\mulooseness`. This value acts analogously to T_EX's `\looseness` command. For non-T_EX-perts: if you state `\looseness=-1` somewhere inside any paragraph, then T_EX will try to make the paragraph one line shorter than it normally would.

With `\mulooseness`, MusiX_TE_X does the same, but for *systems* and *sections* rather than paragraphs. A system is just a group of staves treated as a unit, and in this discussion is analogous to a line of text. What is a *section*? It's any chunk of coding not containing a forced system break, System breaks can be forced with `\stoppiece`, `\endpiece`, `\zstoppiece`, `\Stoppiece`, `\Endpiece`, `\alaligne` `\zalaligne`, `\alapage` or `\zalapage`. If none of these is present, the section comprises the whole piece. Somewhere³ before the end of the section, you can change the value of `\mulooseness` to something different from the default of 0, and MusiX_TE_X will typeset that section with a different number of systems.

To give an easy example, changing the last line in the previous example to:

```
\Notes\qu c\qp\en\mulooseness=1\Endpiece
```

yields:



On the other hand,

```
\Notes\qu c\qp\en\mulooseness=-1\Endpiece
```

yields



which is tighter than you would ever want, but serves to further demonstrate the use of `\mulooseness`.

If you want to build up a MusiX_TE_X input file manually (which very few users will ever need to do, considering the availability of **PMX**), here is a roadmap for one way to proceed:

1. Enter the data one `\notes` group at a time, taking care to select the horizontal spacing for each group (via the macros discussed in section 2.2.2) so as to maintain a consistent relationship between scalable space and note durations. This will be discussed in a great deal more detail in Chapter 2.

³Advisably, at the beginning or at the end of the section, for the sake of clarity.

2. $\text{T}_{\text{E}}\text{X} \implies \text{musixflx} \implies \text{T}_{\text{E}}\text{X}$.
3. Look at the output and decide if you want to have more or fewer systems, e.g. to fill the page or to get an even number of pages.
4. If you want to change the number of systems in a section, adjust `\mulooseness` accordingly. Keep in mind that each section cannot have fewer systems than bars.
5. Delete `jobname.mx2` and repeat the process
 $\text{T}_{\text{E}}\text{X} \implies \text{musixflx} \implies \text{T}_{\text{E}}\text{X}$.

There is an alternate way to proceed if you know at the outset how many systems you want in a section. You can specify it directly by assigning a positive number to `\linegoal` somewhere within the section (`\linegoal` requires version 0.83 or later of `musixflx`). `\mulooseness` must be zero for `\linegoal` to work. Both are automatically reset to zero after processing the end of a section e.g. as defined by `\stoppiece`.

Finally, for large scores (more than 4 pages or so), having only one section and an overall value of `\mulooseness` becomes impractical since one wants not only to have nicely spaced systems, but also completely filled pages without empty top and bottom margins on the last page. It is then wise to force the total number of pages and possibly the line breaks in each page, which can be done using `\alapage` and/or `\alaligne`⁴.

There is another advantage to MusiX $\text{T}_{\text{E}}\text{X}$'s way of using scalable space and the three pass system. In $\text{T}_{\text{E}}\text{X}$ nicol terms it eliminates the need for *glue*, and enables every horizontal position in a line to be computed. This in turn enables certain variable length symbols such as slurs to be specified by macros entered at their beginning and ending points, rather than having to estimate the length of the symbol and enter it at the starting point.

1.3.1 External executable `musixflx`

One issue that compromises portability between computers is the need for the executable `musixflx`. To address this, the C source and compiled versions for various OS's are included in `musixtex.zip` and are available from the [Werner Icking Music Archive](#).

On most computers, the executable is invoked by typing the name of the program and the name of the file to be acted upon. *i.e.*

```
musixflx jobname.mx1
```

Optionally, you can add a letter to indicate one of the debug modes, which are:

- `d` for debug information to screen
- `f` for debug information to file `jobname.mx1`
- `s` to get the computed lines immediately on screen

To allow for ease of use with a batch file, `musixflx` can either be fed with `jobname.mx1`, `jobname.tex` or only `jobname`, any one of which will open `jobname.mx1`.

1.3.2 Unrecorded spaces: the novice's bugaboo

Because of the way MusiX $\text{T}_{\text{E}}\text{X}$ accounts for hard and scalable space and avoids using glue, it is absolutely essential that every horizontal space be properly entered into the input file. The most common error in this regard is including a blank space in the midst *or at the end* of an input line. Such a blank space, or for that matter any stray character not entered with an appropriate MusiX $\text{T}_{\text{E}}\text{X}$ macro, will not be properly recognized and recorded by MusiX $\text{T}_{\text{E}}\text{X}$, but it will still be treated like ordinary text by $\text{T}_{\text{E}}\text{X}$. The symptoms of such a transgression will be an **Overfull**

⁴This is the technique always used by `PMX` in constructing a MusiX $\text{T}_{\text{E}}\text{X}$ input file.

hbox warning during the third pass, and the appearance of either excessive blank horizontal space or thick black vertical lines in the page image.

Considerable discipline is needed to avoid this problem!

The best technique for avoiding such unpleasantness is to avoid entering any nonessential blanks within input lines, and to end every input line with either `%` or `\relax`, unless it already ends with a control sequence ending in a letter.

There are other ways to enter unrecorded space which should be avoided. Between `\startpiece` and `\stoppiece` or `\endpiece`, never use `\hskip` or `\kern` except within zero-boxes like `\rlap`, `\llap`, `\zcharnote`, `\uptext`, etc., and never assign hard values to scalable dimensions like `\noteskip`, `\elemskip`, `\afterruleskip` and `\beforeruleskip`⁵.

Here is a checklist of tips related to spacing issues. Because the foregoing several paragraphs are so important, some of their content is repeated in the list.

1. Between `\startpiece` and `\stoppiece` or `\endpiece`, end every input line with a either `%` or a command with no arguments (including `\relax`).
2. `\off` must only be used with scalable values, e.g. `\noteskip`, `\elemskip`, `\afterruleskip`, and `\beforeruleskip`.
3. Remember that `\qsk` and `\hqsk` are scalable, i.e. `\qsk` doesn't necessarily mean exactly one note head width (it depends on `\elemskip`).
4. Lyrics and any other non-MusiX_{TEX} text must be put in zero boxes such as `\zcharnote`, `\zchar`, `\rlap`, `\lrlap`, `\llap`, `\uptext`, or `\zsong`. Additional specific capabilities for entering lyrics are provided by `\hardlyrics`, `\hsong`, and—most significantly—the text-emplacement macros defined in `musixlyr.tex` (see 2.21, p. 74).
5. Between `\startpiece` and `\stoppiece` or `\endpiece`, don't assign hard values to `\noteskip`, `\beforeruleskip`, or `\afterruleskip`.

1.4 Further highlights

1.4.1 Key signatures

A single key signature can be assigned to all 3 instruments, for example by `\generalsignature{-2}` which sets two flats on each staff. `\generalsignature` can be overridden for selected instruments, for example by `\setsign2{1}` which puts one sharp on each staff of instrument number 2. Of course, the current signature as well as meters and clefs may change at any time.

1.4.2 Transposition

With some extra attention, a score can be input in such a way that it is fairly easily transposed. There is an internal register called `\transpose`, the default value of which is zero, but which may be set to any reasonable positive or negative value. It defines a number of pitch steps (lines or spaces on a staff) by which all pitched symbols will be offset, provided they have been entered with letter values to represent their pitch. However, it will neither change the local accidentals nor the key signature.

⁵Note that `\hardspace` does not fall in this category; it is specifically designed to enter hard space in a way the MusiX_{TEX} can properly record it

For example, suppose a piece were originally input in F major, and it contained a B natural, and you wanted to transpose it to G. If you simply set `\transpose` to 1 with no other special considerations, then the key signature would not change, and the B natural would appear as a C natural, whereas it should be a C sharp. So first you must explicitly change the key signature. Then, to solve the problem with accidentals, you should declare `\relativeaccid`, which will cause the actual appearance of any accidental to depend on the pitch of the accidental as well as the current key signature. But the use of this facility requires the typesetter to have entered the original set of accidentals according to a nonstandard convention wherein an accidental does not specifically refer to the black or white keys on a piano, but to the amount by which the pitch is altered up or down from what it would naturally have been, taking the key signature into account. This is discussed in more detail in section 2.9.2.

There is also an issue with stem and beam directions. Normally a typesetter would want full control over them, and would exercise that control by entering them with macros that explicitly assign the direction. Naturally the assigned directions would persist even after changing `\transpose`. With respect to stems of unbeamed notes, this matter can also be addressed at the input level, by using special macros for notes that leave the decision about stem directions up to MusiX_{TEX}. These macros do the right thing in the face of transposition. They are discussed in section 2.4. Unfortunately there is no corresponding such facility for beams, so the typesetter will have to edit the transposed score to adjust beam directions as required⁶.

1.4.3 Extracting parts from a score

Another question is: “*Can I write a full score and then extract separate scores for each individual instrument?*”

The answer is yes, but only with a great deal of special attention—so much, in fact, that we shall strongly recommend that if you want to do this, you should use **PMX**, which makes the process very easy. If for some reason you choose to ignore this advice, you may refer to the details that were provided in the prior version of this manual, [mxdoc112.pdf](#).

1.4.4 Staff and note sizes

Although the standard staff size is 20pt, MusiX_{TEX} allows scores with sizes of 16, 24, or 29pt. Furthermore, any instrument may be assigned its own special staff size (usually smaller than the overall staff size), and there are special macros (e.g. `\smallnotesize`, `\tinynotesize`) that cause notes, beams, and accidentals all to take a different size.

1.4.5 Add-in macro libraries

During the early stages of MusiX_{TEX}’s development, common versions of \TeX itself were very limited in capacity, especially in terms of the numbers of registers that could be defined for use in macros. For this historical reason, many important enhancements to MusiX_{TEX} are available only via add-in libraries. The user can thus pick and choose which to include for any particular compilation. Most of these are included in `musixtex.zip`, and their uses are discussed in this manual. The libraries have names like `blabla.tex`, and are activated by including a line like `\input blabla` within the the input file. The most common such files are `musixadd.tex` and `musixmad.tex` which respectively increase the number of instruments from the default 6 to 9 or 12 as well as increasing available numbers of other features; `musixps.tex` which enables Type K postscript slurs; and `musixlyr.tex` which greatly eases typesetting lyrics. In fact the latter two,

⁶**PMX** will automatically adjust both stem and beam directions when transposing. However if a piece is to be transposed, the typesetter must still explicitly activate relative accidentals and to enter accidentals according to the relative accidental convention.

while now included in `musixtex.zip`, are not documented in this manual but in separate files inside `musixtex.zip`, namely `musixps.tex` itself and `mxlyrdoc.pdf` respectively.

1.5 Where to get the software and help using it

The home base for all matters related to MusiX_TE_X is the Werner Icking Music Archive, at <http://icking-music-archive.org>. The most up-to-date versions of MusiX_TE_X and friends are located in the [software](#) section of the archive. Assuming you already have T_EX and only want to install MusiX_TE_X, the file you need to download will be named [musixtex.zip](#). Further details about the installation process are given in section 3.

The Werner Icking Archive also hosts the [T_EX-music mailing list](#), where you will always find someone willing to answer questions and help solve problems.

1.6 A very brief history of MusiX_TE_X

The idea of using T_EX to typeset music appears to have originated around 1987 with the master's thesis of Andrea STEINBACH and Angelika SCHOFER⁷. They called their package M^uT_EX. It was limited to a single staff. It introduced two key concepts: (1) using a large number of font characters to construct beams and slurs, and (2) using T_EX glue to help control horizontal spacing and justification.

The next major step came around 1991 when Daniel TAUPIN created MusicT_EX. Its major enhancement was to allow multiple staves. But this came at a price: some flexibility was lost in controlling horizontal spacing and a great deal of trial and error became necessary to avoid excessive or insufficient gaps before and after bar lines.

MusicT_EX was a single-pass system. To remedy its shortcomings it became clear that a multi-pass system would be required. Around 1997 Dr. Taupin along with Ross MITCHELL and Andreas EGLER created the first version of MusiX_TE_X. At last a fully automatic procedure was coded so as to provide pleasing horizontal spacing in multi-staff scores.

Significant enhancements to MusiX_TE_X, which have already been mentioned, have been provided by Stanislav KNEIFL (Type K postscript slurs) and Rainer DUNKER (Lyrics handling via `musixlyr.tex`).

Since Dr. Taupin passed away in 2003, the primary maintainer of MusiX_TE_X has been Olivier VOGEL, with substantial assistance from Andre VAN RYCKEGHEM, Cornelius NOACK, and Don SIMONS.

No discussion of the history of MusiX_TE_X would be complete without mentioning the contributions of Werner ICKING. From the early days of M^uT_EX until his untimely death in 2001, he served this line of software as its most prominent proponent, beta tester, web site and mailing list editor, consultant, problem solver, and inspiration for many third-party enhancements including **PMX**. In fact he founded the mailing list and the archive that now is named in his honor. The web site is currently edited by Christian MONDRUP, the software page by Don SIMONS, and the mailing list by Maurizio CODOGNO.

⁷Steinbach A. & Schofer A., *Automatisierter Notensatz mit T_EX*, master's thesis, Rheinische Friedrich-Wilhelms Universität, Bonn, Germany, 1987

Chapter 2

Elements of MusiX_{TEX}

2.1 Setting up the input file

2.1.1 What makes a _{TEX} file a MusiX_{TEX} file?

A MusiX_{TEX} input file is a special kind of _{TEX} input file. What makes it special is that it must contain the command `\input musixtex` before any reference to MusiX_{TEX} macros. After that might follow `\input musixadd` or `\input musixmad` if you want to have respectively more than six or nine instruments or simultaneous beams, ties, or slurs. Since it is still a _{TEX} file, after that, if you wished to, you could write a whole non-musical book using normal _{TEX} commands provided that you did not use `&` as a tab character like in plain _{TEX} (In _{TEX} lingo, its `\catcode` has been changed).

2.1.2 Cautions for the non _{TEX}pert

When _{TEX} reads anything, it inputs one *token* at a time. A token may be either a *command* or a character. A command (or *macro*, or *control sequence*) is a *backslash* (“\”) immediately followed by sequence of letters with no intervening spaces. For practical purposes, any single symbol (letter, digit, special character, or space) that is not part of a command counts as a character and therefore as a token.

Each command expects a specific number of parameters. The tokens “{” and “}” are very special, in that (1) they must occur in matched pairs, and (2) any matched pair together with the stuff inside counts as a single parameter. If the first parameter expected is a single letter, it must either be separated from the command by a space or enclosed in braces (otherwise it would be interpreted as part of the command). For example the command `\ibu` expects three parameters, so the following are all OK: `\ibu123`, `\ibu1A3`, `\ibu1{‘A}3`, `\ibu{1}{2}{3}`, `\ibu1{-2}3`, or `\ibu1{23}4`, but `\ibu1234` is not OK; the first three digits are taken as parameters, leaving the “4” with no purpose other than to cause some of the dreaded unrecorded space that we have already mentioned.

In the rest of this manual, when describing commands we will write things like `\qb{n}{p}`. It should be understood that when *n* and *p* are replaced by their literal values, the braces may or may not be necessary. In particular, if both are single digits, no braces are needed; but if *n* has two digits, or if *p* has more than one character, they must be surrounded by braces.

Spaces (blank characters) in the input file must be handled very carefully. They are ignored at the beginning of a line, enabling logical indentation schemes to help make the file human-readable. There are also a few other places within lines where blank spaces are OK (such as mentioned in the prior paragraph), but in general is it safest to avoid any unnecessary blanks between the beginning and end of an input line. At the end of a line, the truth is that a command with no

parameters, such as `\bar` or `\enotes` will cause no trouble. However if a command with one or more parameters is the last item in an input line, it will cause unrecorded space. The way around this is to end the line with either “%” or `\relax`.

2.1.3 Usual setup commands

The first decision is what size type to use. MusiX_{TEX} offers four sizes: “small” (16-pt-high staves), “normal” (20pt), “large” (24pt), and “Large” (29pt). The default is `\normalmusicsize`. If you want a different size, then you have to enter `\smallmusicsize`, `\largemusicsize`, or `\Largemusicsize`. Each of these commands defines not only the desired staff size but many other related sizes such as note heads, ornaments, stem lengths, etc.

The command `\instrumentnumber{n}` defines the number of instruments to be n . If not entered, the default is 1. This number is used in loops that build staves, set key signatures, set meters, etc., so if it differs from 1 it must be explicitly defined before any further commands.

An instrument may have one or more staves (e.g. a piano would normally have 2 staves). The differences between one instrument of several staves and several instruments with one staff each are as follows:

- Different instruments may have different *key signatures*, while different staves of an instrument will all have the same key signature.
- A *beam* may include notes in different staves of the same instrument.
- A *chord* may extend across several staves of the same instrument.
- If an instrument has more than one staff, they will be linked together with a big, curly brace at the beginning of each line.

The default number of staves per instrument is 1. If it is different, then it must be specified by `\setstaves{n}{p}` where p is the number of staves and n is the number of the instrument. In MusiX_{TEX}, instruments are numbered *starting with the lowest*. So for example `setstaves32` assigns two staves to the third instrument from the bottom.

The default clef for every staff is the *treble* clef. To assign any other clef, the command is `\setclef{n}{s1s2s3s4}` where n is the number of the instrument, s_1 is a digit specifying the clef for the first (lowest) staff, s_2 for the second staff, and so forth. Note that like instruments, staves of a given instrument are numbered starting with the lowest. The parameters s_2 , s_3 and s_4 can be omitted, in which case any unspecified staves will be assigned a treble clef.

The digits s can range from 0 to 9, with the following meanings: $s = 0$ signifies treble or G clef. $s = 1$ to 4 mean C-clef, respectively on the first (lowest) through fourth staff line. 1 is also called *soprano*, 3 *alto* and 4 *tenor*. $s = 5$ to $s = 7$ mean F-clef, respectively on the third through fifth staff line. 5 is also called *baritone* and 6 is the normal *bass*. $s = 8$ is not used. $s = 9$ represents a G clef on the first line, also called *French violin* clef.

The three tokens `\treble`, `\alto`, and `\bass` can be used instead of a digit for s , but only if there would have been but one digit in the string. So for example the clefs for a standard piano score could be specified by `\setclef1{\bass}`.

To set a common key signature for all instruments, use `\generalsignature{s}`, where $s > 0$ is the number of *sharps* in the signature and $s < 0$ the number of *flats*¹. To override the common key signature for instrument n , use `\setsign{n}{s}`. Note that differing key signatures cannot be assigned to different staves of the same instrument.

A common *meter* for all staves can be specified by `\generalmeter{m}`, where m describes the appearance of the meter indication, and can take several different forms. If the meter is a *fraction*

¹We once saw a score in G-minor where the signature consisted of two flats (B and E) plus one sharp (F). This is not directly supported by MusiX_{TEX}.

(e.g. 3/4) the command is `\generalmeter{\meterfrac{3}{4}}`. Other possible tokens m are `\meterC`, `\allabreve`, `\reverseC`, `\reverseallabreve` and `\meterplus`. These are illustrated in the following example:



which was coded as:

```
\generalmeter\meterC
\nostartrule
\parindent0pt\startpiece
\NOTes\qa{cegj}\enotes
\generalmeter\allabreve\changecontext
\NOTes\ha{ce}\enotes
\generalmeter\reverseC\changecontext
\NOTEs\zbreve g\enotes
\generalmeter\reverseallabreve\changecontext
\NOTEs\zwq g\enotes
\generalmeter{\meterfrac{3\meterplus2\meterplus3}{8}}\changecontext
\Notes\Tqbu ceg\Dqbl jg\Tqbu gec\enotes
\endpiece
```

To override the common meter for any staff, use `\setmeter{n}{m_1}{m_2}{m_3}{m_4}`. This works just like `\setclef`. For example, `setmeter3{\meterfrac{12}{8}\allabreve}` sets the meter to 12/8 for the first staff of the third instrument, and *alla breve* for the second staff.

To insert extra space before the meter is written, use `\meterskipd` where d is any hard TeX dimension². The assignment must occur outside `\startpiece... \endpiece` and will be reset to zero after first meter is posted.

To set an *instrument name*, use `\setname{n}{name of the instrument}`. This will place the name in the space to the left of the first staff or group of staves for instrument n . To specify the amount of space available, use `\parindentd` where d is any hard TeX dimension.

2.1.4 Groupings of instruments

By default, all staves in a system will be joined at the left by a thin, vertical rule. In addition, if an instrument has more than one staff, they will be joined by a big, curly brace. Now we introduce a way to delineate groups of instruments or choirs with a square brace containing two parallel vertical rules, the left one thick and the right one thin. This is commonly used to group together the voices in a choir.

If there is only one choir, this can be done with

```
\songtop{n}
\songbottom{m}
```

where m and n are the instrument numbers of the first and last voices. An example is shown in section 2.23.5.

If there is more than one choir to be set off with square braces, each one can be specified with

```
\grouptop{g}{n}
\groupbottom{g}{m}
```

²`\meterskip` is not a macro but a dimension register. Whatever follows it must be a TeX dimension and *it must not be enclosed in braces*.

where m and n are the instrument numbers of the first and last voices of group number g . MusiXTEX allows up to three groups, numbered from 1 to 3. The command `\songtop` is equivalent to `\grouptop 1`; `\songbottom` is equivalent to `\groupbottom 1`.

With `musixadd.tex`, the allowable number of groups is increased to four.

If any of the instruments grouped this way has more than one staff, the heavy curly brace will be shifted to the left of the square brace.

Previously defined square braces can be removed by declaring `\songtop` less than `\songbottom`. The same applies to `\grouptop` and `\groupbottom` for the same group number.

An alternate command allows you to specify all choirs at once:

```
\akkoladen{{lower_1}{upper_1}{lower_2}{upper_2}{lower_3}{upper_3}}
```

where $lower_n$ and $upper_n$ are instrument numbers that denote the span of bracket number n . For setting fewer than three brackets, just omit all unneeded $\{lower_n\}\{upper_n\}$ pairs. For example, `\instrumentnumber{5}\akkoladen{{1}{2}{3}{5}}` yields the first example below, with five single-staff instruments divided into two groups.

The second example has 2 instruments, the first (lower) with two staves and the second with three. Each instrument is set off by default with a curly bracket.

If for some reason you want more than one *instrument* grouped within a curly bracket, then you can use the extension file [curly.tex](#)³, which defines the command

```
\curlybrackets{{lower_1}{upper_1}{lower_2}{upper_2}...
```

to be used as illustrated in the third example below.



is coded as:

```
\sepbarrules
\smallmusicsize
\setsize1\smallvalue
\setsize2\smallvalue
\setsize3\smallvalue
\setsize4\smallvalue
\setsize5\smallvalue
\instrumentnumber{5}
\akkoladen{{1}{2}{3}{5}}
\startextract
\notes\en\bar\notes\en
\endextract
```



is coded as:

```
\sepbarrules
\smallmusicsize
\setsize1\smallvalue
\setsize2\smallvalue
\instrumentnumber2
\setstaves12
\setstaves23
\startextract
\notes\en\bar\notes\en
\endextract
```



is coded as:

```
\input curly
\sepbarrules
\instrumentnumber5
\smallmusicsize
\setsize1\smallvalue
\setsize2\smallvalue
\setsize3\smallvalue
\setsize4\smallvalue
\setsize5\smallvalue
\curlybrackets{1235}
\startextract
\notes\en\bar\notes\en
\endextract
```

³Submitted by Mthimkhulu MOLEKWA to the mutex list

2.2 Preparing to enter notes

2.2.1 After the setup, what next?










The command `\startmuflex` initiates the serious business of MusiX \TeX . On the first \TeX pass it opens `jobname.mx1` for writing bar-by-bar tabulations of all hard and scalable space to be fed to `musixflx` on the second pass. `musixflx` generates `jobname.mx2` which defines the number of bars in each system and the factors relating scalable space to hard space in each system. On the third pass both files will be opened and read to define the final spacing. These files should be closed before leaving \TeX , preferably before `\bye` or `\end`, with `\endmuflex`. Normally \TeX closes all open files on its own when terminating the program, but it is still cleaner to do this explicitly.

After `\startmuflex`, the command `\startpiece` will initiate the first system, containing all instruments you have previously defined. The indentation will be `\parindent`, so if you want nonzero indentation, this register should be set to the desired hard dimension before issuing `\startpiece`.

2.2.2 Horizontal spacing commands

2.2.2.1 Basic note spacing

MusiX \TeX provides a set of macros each of which defines a particular increment of scalable spacing. The default set is tabulated below:

usage	spacing	suggested use for
<code>\znotes ... & ... & ... \enotes</code>	(non spacing)	specials
<code>\notes ... & ... & ... \enotes</code>	<code>2\elemskip</code>	 16th
<code>\notesp ... & ... & ... \enotes</code>	<code>2.5\elemskip</code>	 dotted 16th, 8th triplet
<code>\Notes ... & ... & ... \enotes</code>	<code>3\elemskip</code>	 8th
<code>\Notesp ... & ... & ... \enotes</code>	<code>3.5\elemskip</code>	 dotted 8th, quarter triplet
<code>\NOTes ... & ... & ... \enotes</code>	<code>4\elemskip</code>	 quarter
<code>\NOTesp ... & ... & ... \enotes</code>	<code>4.5\elemskip</code>	 dotted quarter, half triplet
<code>\NOTes ... & ... & ... \enotes</code>	<code>5\elemskip</code>	 half
<code>\NOTesp ... & ... & ... \enotes</code>	<code>5.5\elemskip</code>	 dotted half
<code>\NOTEs ... & ... & ... \enotes</code>	<code>6\elemskip</code>	 whole

What each of these macros actually does is to set an internal dimension register `\noteskip` to the given multiple of the fundamental spacing unit `\elemskip` (which has dimensions of length, usually given in points). Normally, every *spacing note* (e.g., `\qu`, `\qb`, `\h1`) will then be followed by a spacing of width `\noteskip`. By selecting a particular note spacing macro from the above table, the typesetter can thus control the relative spacing between notes.

The actual spacing will therefore be determined by the value of `\elemskip`. On the first pass, \TeX will set a default value for `\elemskip` based on the declared music size, or the user can set it to any hard dimension he chooses. However, the value on the first pass doesn't matter as much as you might think (more about that later). On the second pass, `musixflx` determines where the system breaks will come, and then computes the final value of `\elemskip` for each system.

If the arithmetic progression of note spacings in the above table does not meet your wishes, you may activate an alternate set with the command `\geometricskipsscale`. As implied by the name,

this is a geometric progression, where `\Notes` is $\sqrt{2}$ times wider than `\notes`, `\N0tes` is $\sqrt{2}$ times wider than `\Notes`, and so forth. Then the factors in the middle column of the above table will be replaced by the sequence 2.00, 2.38, 2.83, 3.36, 4.00, 4.76, 5.66, 6.72, and 8.00. Two additional macros, `\NOTEsp` and `\NOTES`, will be defined corresponding to factors 9.52 and 11.32. The original arithmetic progression can be restored by `\arithmeticsskip`.

If neither of the predefined progressions satisfies you, you may define your own, using the more general macro `\vnotes` in the same manner that MusiXTEX uses it for the predefined progressions. So for example `\def\N0tes{\vnotes5.34\elemskip}` will redefine `\N0tes` in the obvious way, and the extension to the other spacing macros should likewise be obvious.

In addition, inside any pair `\notes... \enotes` there are two equivalent ways to locally redefine `\noteskip` to another scalable value, namely by issuing a command like `\noteskip=2.4\noteskip` or `\multnoteskip{2.4}`, which have the expected effect until the notes group is terminated or `\noteskip` is further redefined.

Finally, by issuing a command like `\scale{2.4}` outside any notes group, you can scale all subsequent `\noteskips` by any desired factor.

These facilities may be useful, for example, to control spacing when there are three equal duration notes in one staff against two in another.

2.2.2.2 `\elemskip`, `\beforeruleskip` and `\afterruleskip`

We've just seen how `\elemskip` is used to scale the spacings between notes. There are two other spacing units that share some behavior with `\elemskip`. `\beforeruleskip` is the horizontal space that is automatically inserted *before* every bar line, while `\afterruleskip` goes *after* every bar line. (In practice `\beforeruleskip` is almost always set to 0pt because there will typically already be a space of `1\noteskip` before every barline.) On the first pass, just as with `\elemskip`, MusiXTEX assigns them default values according to the following table:

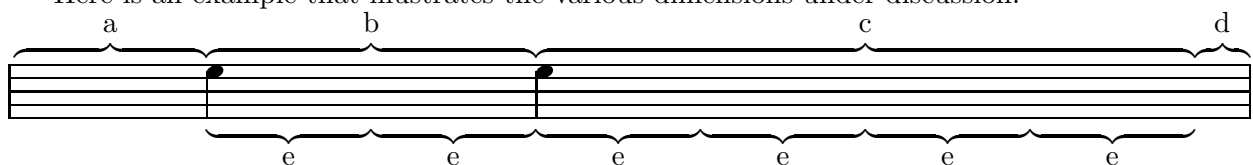
<i>using</i>	<code>\elemskip</code>	<code>\afterruleskip</code>	<code>\beforeruleskip</code>
<code>\normalmusicsize</code>	6pt	8pt	0pt
<code>\smallmusicsize</code>	4.8pt	6pt	0pt

In the second pass, `musixflx` assigns new values to each of these dimensions, a different set for each line or system. It does this in such a way that available scalable horizontal space in each system is exactly filled up.

The values that are assigned to these dimensions on the first pass, whether by default or explicitly by the user or in some combination, only matter insofar as their relative sizes. That's why we earlier stated that the first-pass value of `\elemskip` didn't matter as much as you might think. For both music sizes in the table above, it appears that by default `\afterruleskip` is $1.3333\elemskip$ ⁴.

Note that if you do want to change any of these values, you have to do so *after* setting the music size and before `\startpiece`.

Here is an example that illustrates the various dimensions under discussion:



⁴Editor's note: It is a mystery why the authors of MusiXTEX didn't simply define `\beforeruleskip` and `\afterruleskip` as specific multiples of `\elemskip`


```

a → \afterruleskip
b → \notes = \vnotes 2\elemskip
c → \N0tes = \vnotes 4\elemskip
d → \beforeruleskip
e → \elemskip

```

2.2.3 Moving from one staff or instrument to another

When entering notes inside `\notes ... \enotes`, the usual way to suspend input for one instrument and start the next (higher) is with the character “&”. If the instrument has more than one staff, to switch to the next (higher) one you can use the character “|”.

There are some alternate navigation commands that may be useful in special situations. Due to “catcode problems” (see section 2.22.3) it may sometimes be necessary to use the more explicit commands `\nextinstrument` and `\nextstaff`, which have the same meanings as “&” and “|” respectively. To switch to the previous (next lower) staff of the same instrument, use `\prevstaff`. This might be useful if a beam starts in a higher staff than where it ends. More generally, to switch to an arbitrary instrument n , use `\selectinstrument{n}`, and to switch to an arbitrary staff n of the current instrument, use `\selectstaff{n}`. In the latter case if n exceeds the number of staves defined for the instrument, you will receive an error message. You can enter part of a successive voice on *same* staff by using `\selectstaff{n}` with n for the *current* staff.

2.3 Note pitch specification

Note pitches can be specified either by letters or numbers. If no transposition or octavation is in effect, letters ranging from `a` to `z` represent notes starting with the A below middle C. Upper case letters from `A` to `N` represent pitches two octaves lower than their lower case counterparts. Any letter can be used in any clef, but some users may prefer to use the lower case letters in treble clef, and the upper case ones in bass clef.

Alternatively, a one- or two-digit, positive or negative integer can always be used. The number represents the vertical position on the staff, with 0 for the lowest line and 1 for the space right above, *regardless of the clef*. Unlike with letters, the associated pitch will depend on the clef, and notes entered this way are immune to transposition and octavation.

Notes lower than `A` and higher than `z` can be entered, with either numbers as just described, or with octavation as will be explained in section 2.9.

2.4 Writing notes

There are two major kinds of note macros, those that include a space (of length `\noteskip`) after the printed symbol, and those that don’t cause any space. A single-line melody would be written using the first type. All notes of a chord except the last would use the second.

Another distinction concerns stemmed notes. Some macros explicitly set the stem direction with either “u” or “l” contained in the name of the macro. On the other hand, an “a” in the macro’s name usually signifies *automatic* stem direction selection. In this case notes below the middle staff line will get up stems, otherwise down.

2.4.1 Normal (unbeamed) spacing notes

In the following, $\{p\}$ signifies a pitch specification as described in sections 2.3 and 2.9. However it is understood that if the pitch is a single character, the brackets are not necessary, provided that if it is a letter, a space separates the macro from the letter.

`\breve{p}` : breve (≡) .
`\longa{p}` : longa (≡) .
`\longaa{p}` : longa with automatic stem direction⁵ .
`\zmaxima{p}` : maxima (≡) .
`\wq{p}` : arbitrary duration note (♩) (also used as alternate representation of a *breve*).
`\wqq{p}` : long arbitrary duration note (♩) (also used as alternate representation of a *longa*).
`\wh{p}` : whole note.
`\hu{p}` : half note with stem up.
`\hl{p}` : half note with stem down.
`\ha{p}` : half note with automatic stem direction
`\qu{p}` : quarter note with stem up.
`\ql{p}` : quarter note with stem down.
`\qa{p}` : quarter note with automatic stem direction.
`\cu{p}` : eighth note⁶ with stem up.
`\cl{p}` : eighth note with stem down.
`\ca{p}` : eighth note with automatic stem direction.
`\ccu{p}` : sixteenth note with stem up.
`\ccl{p}` : sixteenth note with stem down.
`\cca{p}` : sixteenth note with automatic stem direction.
`\cccu{p}` : 32nd note with stem up.
`\cccl{p}` : 32nd note with stem down.
`\ccca{p}` : 32nd note with automatic stem direction.
`\ccccu{p}` : 64nd note with stem up.
`\ccccl{p}` : 64nd note with stem down.
`\ccccl{p}` : 64nd note with automatic stem direction.

As an example, the sequence



was coded as

```

\Notes\cu c\cl j\enotes\bar
\Notes\ccu c\ccl j\enotes\bar
\Notes\cccu c\ccccl j\enotes\bar
\Notes\ccccu c\ccccl j\enotes

```

2.4.2 Non-spacing note heads

These macros are used to create chords. Any number of them can be entered in sequence, followed by a spacing note. All of the note heads will be joined to the spacing note and the stem length will automatically be adjusted as needed.

`\zq{p}` : quarter (or shorter) note head.
`\zh{p}` : half note head.

⁵Editor's note: Evidently there is no explicit up-stemmed longa

⁶The "c" within this macro name stands for the equivalent British term "crotchet"

2.4.3 Shifted non-spacing note heads

These symbols are used mainly in chords containing an interval of a *second*. They provide note heads shifted either to the left or right of the default position by the width of one note head.

`\rw{p}` : whole note head shifted right.
`\lw{p}` : whole note head shifted left.
`\rh{p}` : half note head shifted right⁷.
`\lh{p}` : half note head shifted left.
`\rq{p}` : quarter note head shifted right.
`\lq{p}` : quarter note head shifted left.

2.4.4 Non-spacing notes

These macros provide normal notes, with stems if applicable, but without any following space.

`\zhu{p}` : half note with stem up but no spacing. It acts like `\hu` for chord building, i.e., it will join together any immediately preceding non-spacing note heads.
`\zhl{p}` : half note with stem down but no spacing. It acts like `\hl` for chord building.
`\zqu{p}` : quarter note with stem up but no spacing. It acts like `\qu` for chord building.
`\zql{p}` : quarter note with stem down but no spacing. It acts like `\ql` for chord building.
`\zcu{p}`, `\zccu`, `\zcccu`, `\zccccu` : eighth, ..., note with stem up but no spacing. They act like `\cu` for chord building.
`\zcl{p}`, `\zcc1`, `\zccc1`, `\zcccc1` : eighth, ..., note with stem down but no spacing. They act like `\cl` for chord building.
`\zqb{p}` : note belonging to a beam but no spacing. (DAS: put this later!)
`\rhu{p}`, `\rhl`, `\rqu`, `\rql`, `\rcu`, `\rc1` : `\rhu` acts like `\zhu`, but the note is shifted one note width to the right; others analogous.
`\lhu{p}`, `\lhl`, `\lqu`, `\lql`, `\lcu`, `\lc1` : same as above, but the note is shifted one note width to the left.
`\zw{p}` : whole note with no following space.
`\zwq{p}` : arbitrary duration note (♩) with no following space.
`\zbreve{p}` : breve (≡) with no following space.
`\zlonga{p}` : longa (≡) with no following space.
`\zmaxima{p}` : maxima (≡) with no following space.

2.4.5 Spacing note heads

Although not needed in normal music scores, these may be useful in very special cases.

`\nh{p}` : spacing half note head.
`\nq{p}` : spacing quarter note head.

As an example, the sequence



was coded as

⁷Some may not have realized that half and whole note heads have different shapes

```

\notes\nq c\nq j\enotes\barre
\Notes\nh c\nh j\enotes\barre
\notes\nq {cdef}\enotes

```

Non spacing variants are also provided, namely `\znh` and `\znq`.

2.4.6 Dotted notes

By appending one or two p’s (for “pointed”) to the name, many of the macros just introduced provide one or two dots after the notehead: `\whp{p}`, `\whpp`, `\zwp`, `\zwpp`, `\hup`, `\hupp`, `\hlp`, `\hlpp`, `\zhp`, `\zhpp`, `\qup`, `\qupp`, `\qlp`, `\qlpp`, `\zqp`, `\zqpp`, `\cup`, `\cupp`, `\clp`, `\clpp`, `\qbp` and `\qbpp`. Naturally, the ones that start with “z” are used in chords. The dot(s) will be raised if the note is on a line.

A more explicit way uses one of the macros `\pt{p}`, `\ppt`, or `\pppt` right before any note macro to place one to three dots after the normal note head at pitch p . Again they will be raised if on a line. In fact this is the only way to get a triple-dotted note. For example a quarter note with one dot could be coded `\pt h\qu h`, with two dots as `\ppt h\qu h` and with three as `\pppt h\qu h`.

Yet another method for posting a dot is to insert a *period* before the letter representing the pitch. Thus `\qu{.a}` is equivalent to either `\pt a\qu a` or `\qup a .` This may be useful when using *collective coding*, which will be discussed in the next section.

Non-spacing dotted notes can be produced using `\zhup`, `\zhlp`, `\zqup`, `\zqlp`, `\zcup`, `\zclp`, `\zqbp`, and similarly with two p’s for double-dotted notes.

As a matter of style, if two voices share one staff, the dots in the lower voice should be lowered if the note is on a line. For this you can use `\lpt{p}` and `\lppt{p}`.

2.4.7 Sequences of equally spaced notes; collective coding

It isn’t necessary to write a separate macro sequence `\notes... \enotes` for every individual column of notes. Rather, a single such macro can contain all the notes in all staves over an extended horizontal range, as long as all spacings are equal or multiples of a unique value of `\noteskip`. The notes in each staff could be entered one after another as normal spacing notes as already described in section 2.4.1. Then each spacing note will cause the insertion point to advance horizontally by the operative value of `\noteskip` defined by the choice of `\notes`, `\Notes`, `\NOtes`, etc. Of course in such sequences non-spacing chord notes can be entered right before their associated spacing note. If you need to skip forward by one `\noteskip`, for example after a quarter note when there are two eighth notes in another staff, you can use `\sk`.

If there are only spacing notes in such a sequence, a further simplification is available, called *collective coding*. For instance `\qu{cdefghij}` writes the C major scale in quarters with up stems. Similarly `\cl{abcdef^gh}` writes the A-minor scale in non-beamed eighths. (Here “^” represents a sharp). If necessary a void can be inserted in a collective coding sequence by using `*`. Not all note generating macros can be used to perform collective coding, but most of them can.

2.5 Beams

2.5.1 Starting a beam

Each beam must be declared with a macro issued before the first spacing note under the beam is coded. Two distinct kinds of macros are provided for this. The first kind initiates a “fixed-slope” beam, with an arbitrary slope and starting height chosen by the user, while the second kind, a

“semi-automatic” beam, *computes* the slope and, in addition, adjusts the starting height in some cases.

The basic form of the macros for starting fixed-slope beams is exemplified by the one for a single upper beam, `\ibu{n}{p}{s}`. Here n is the reference number of the beam, p the starting “pitch”, and s the slope. The reference number is assigned by the user. By default it must be in the range [0-5], but the maximum can be increased to 8 or 11 respectively if `musicadd` or `musicmad` has been `\input`. The beam number is needed because more than one beam may be open at a time, and it tells MusiX_{TEX} to which beam subsequent beamed notes and other beam specification commands are assigned. The “pitch” parameter is a pitch that is three staff spaces *below* the bottom of the heavy connecting bar (*above* the bar for a lower beam); in many (but not all) cases it should be input as the actual pitch of the first note. The slope s is an integer in the range [-9,9]. When multiplied by 5% it gives the actual slope of the heavy bar. Typically a slope of 2 or 3 is OK for ascending scales, and 6 to 9 for ascending arpeggios.

The full set of fixed-slope beam initiation macros is as follows:

`\ibu{n}{p}{s}` : initiates an *upper beam*.
`\ibl{n}{p}{s}` : initiates a *lower beam*.
`\ibbu{n}{p}{s}` : initiates a *double upper beam*.
`\ibbl{n}{p}{s}` : initiates a *double lower beam*.
`\ibbbu{n}{p}{s}` : initiates a *triple upper beam*.
`\ibbbbl{n}{p}{s}` : initiates a *triple lower beam*.
`\ibbbbu{n}{p}{s}` : initiates a *quadruple upper beam*.
`\ibbbbl{n}{p}{s}` : initiates a *quadruple lower beam*.

A semi-automatic beam is initiated with a command that has *four* parameters, the beam number, the first and last pitches, and the total horizontal extent in `\noteskips`, based on the value in effect at the start. For example, if you input `\Ibu2gj3`, MusiX_{TEX} will understand that you want to build an upper beam (beam number 2) horizontally extending `3\noteskip`, the first note of which is a `g` and the last note a `j`. Knowing these parameters it will choose the highest slope number that corresponds to a slope not more than $(j - g)/(3\text{noteskip})$. The nominal height of the heavy bar is offset the same as for fixed-slope beams. However, if there is no sufficiently steep beam slope available, then MusiX_{TEX} will raise (or lower) the starting point.

Eight such macros are available: `\Ibu`, `\Ibbu`, `\Ibbbu`, `\Ibbbbu`, `\Ibl`, `\Ibbl`, `\Ibbbl` and `\Ibbbbbl`.

2.5.2 Adding notes to a beam

Spacing notes belonging to beams are coded with the macro `\qb{n}{p}` where n is the beam number and p the pitch of the note. MusiX_{TEX} adjusts the length of the note stem to link to the beam.

Chord notes within a beam are entered before the main note with the non-spacing macro `\zqb{n}{p}`. Again, the stem length will be automatically adjusted as required.

There are also special macros for semi-automatic beams with two, three, or four notes: `\Dqbu`, `\Dqbl`, `\Dqbbu`, `\Dqbbbl`, `\Tqbu`, `\Tqbl`, `\Tqbbu`, `\Tqbbbl`, `\Qqbu`, `\Qqbl`, `\Qqbbu` and `\Qqbbbl`. For example `\Dqbu gh` is equivalent to `Iqbu1gh\qb1 g\tbu1\qb1 h`, except that the special macros don’t require a beam number. Their use is illustrated in the following example:



This was coded as⁸:

```
\Notes\Dqbu gh\Dqbl jh\en
\notes\Dqbbu fg\Dqbb1 hk\en\bar
\Notes\Tqbu ghi\Tqbl mmj\en
\notes\Tqbbu fgj\Tqbb1 njh\en\bar
\Notes\Qqbu ghjh\Qqbl jifh\en
\notes\Qqbbu fgge\Qqbb1 jhgi\en
```

2.5.3 Ending a beam

The termination of a given beam must be declared *before* coding the last spacing note connected to that beam. The macros for doing that are `\tbu{n}` for an upper beam and `\tbl{n}` for a lower one. These work for beams of any multiplicity. So for example an upper triple beam with 32nd notes is initiated by `\ibbbu{n}{p}{s}` but terminated by `\tbu{n}`.

Since beams usually finish with a `\qb` for the last note, the following shortcut macros have been provided:

```
\tqb{n}{p} is equivalent to \tbl{n}\qb{n}{p} .
\tqh{n}{p} is equivalent to \tbu{n}\qb{n}{p} .
\ztqb{n}{p} is equivalent to \tbl{n}\zqb{n}{p} , i.e., no spacing afterwards.
\zth{n}{p} is equivalent to \tbu{n}\zqb{n}{p} , i.e., no spacing afterwards.
```

2.5.4 Changing multiplicity after the beam starts

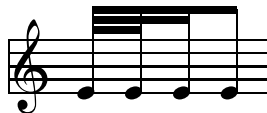
Multiplicity (the number of heavy bars) can be increased at any position after the beam starts. The commands are `\nbbu{n}` which increases the multiplicity of upper beam number n to two starting at the current position, `\nbbbu{n}` to increase it to three, and `\nbbbbu{n}` to increase to four. The commands `\nbb1{n}... \nbbbb1{n}` do the same for lower beams. Thus, the sequence



has been coded as

```
\Notes\ibu0h0\qb0e\nbbu0\qb0e\nbbbu0\qb0e\nbbbbu0\qb0e\tbu0\qb0e\enotes
```

To decrease multiplicity to one, use `\tbbu{n}` or `\tbl{n}`. To decrease to two or three use `\tbbbu{n}... \tblbb1{n}`. For example,



has been coded as

```
\startextract
\Notes\ibbbu0h0\qb0e\tbbbu0\qb0e\tbbu0\qb0e\tbu0\qb0e\enotes
\endextract
```

Although at first it may seem counterintuitive, the macros `\tbbu` and `\tbl1` and higher order counterparts may also be invoked when the multiplicity is one. In this case a second, third, or

⁸Editor's note: Most music typesetting books recommend beam slopes that are *less* than the slope between the starting and ending note; these macros cannot provide that.

fourth heavy bar will be opened one note width *before* the current stem, and immediately closed *at* the stem. Thus the following sequences



are coded `\Notes\ibu0e0\qbp0e%` `\Notes\ibu0e0\qbpp0e%`
`\tbbu0\tbu0\qb0e\en` `\tbbbu0\tbbu0\tbu0\qb0e\en`

The symmetrical pattern is also possible. For example:



has been coded as:

```
\Notes\ibbl0j0\roff{\tbb10}\qb0j\tbl0\qbp0j\enotes
```

The constructions in this section illustrate some general properties of beam initiation and termination commands: To mate properly with the expected stems, the starting position of the heavy bar(s) (for initiation commands) and the ending position (for terminations) will be at different horizontal locations depending on whether they are for upper or lower beams: The position for upper beam commands is one note head width to the right of those for lower beams. In fact this is the *only* difference between upper and lower termination commands. Both types will operate on whatever kind of beam is open and has the same beam number.

Recognizing this principle, in the example just given it was necessary to shift the double termination to the right by one note head width, using the command `\roff{...}`, which does precisely that for any MusiX_{TEX} macro.

Here is another, slightly more complicated example which also uses `\roff`:



has been coded as:

```
\notes\ibbbu0e0\roff{\tbbbu0}\qb0f\en
\notesp\tbbu0\qbp0f\en
\notes\tbu0\qb0f\en
\notesp\ibbu0f0\roff{\tbbu0}\qbp0f\en
\notes\qb0f\en
\notes\tbbbu0\tbbu0\tbu0\qb0f\en
```

Note that the first beam opening command used a pitch one step below the note. This makes the stem shorter by one pitch unit, since it is always the *closest* heavy bar that is separated from the given pitch by three staff spaces.

We close this section with an example showing how to open a beam of one sense, increase multiplicity, then terminate with opposite sense:



which has been coded as

```
\Notes\ib10p0\qb0p\nbb10\qb0p\nbbb10\qb0p\tbu0\qb0e\enotes
```

REMARK: One may save some typing by defining personalized T_EX macros to perform any oft repeated sequence of commands. For example, one could define a set of four sixteenths by the macro:

```
\def\qqh#1#2#3#4#5{\ibbl0#2#1\qb#2\qb#3\qb#4\tbl0\qb#5}
```

where the first argument is the slope and the other four arguments are the pitches of the four successive sixteenths.

2.5.5 Shorthand beam notations for repeated notes

Sometimes you may want to indicate repeated short notes with open note heads joined by a beam. Here's an example of how to do that using the `\hb` macro:



which has been coded as:

```
\Notes\ibbl0j0\hb0j\tbl0\hb0j\enotes
\Notes\ibbu0g0\hb0g\tbu0\hb0g\enotes
```

It is also possible to dispense with the stems:



which was coded as

```
\Notes\ibbl0j3\wh j\tbl0\wh l\enotes
\Notes\ibbu0g3\wh g\tbu0\wh i\enotes
```

A different look could be obtained as follows:



which was coded as:

```
\Notes\loff{\zw j}\ibbl0j3\sk\tbl0\wh l\enotes
\Notes\ibbu0g3\wh g\tbu0\roff{\wh i}\enotes\qspace
```

Yet another way to indicate repeated notes is given in the following example:



whose coding (due to Werner ICKING) is

```
\Notes\ibl0h0\qb0{hhh}\tbl0\qb0h\bsk\bsk\bsk\bsk
\ibu0j0\qb0{jjj}\tbu0\qb0j\en
\NOTes\loffset{0.5}{\ibl0j9}\roffset{0.5}{\tbl0}\zhl h%
\loffset{0.5}{\ibu0g9}\roffset{0.5}{\tbu0}\hu j\en\bar
\notes\ibbl0i0\qb0{hhh}\tbl0\qb0h\bsk\bsk\bsk\bsk
\ibbu0i0\qb0{jjj}\tbu0\qb0j%
\ibbl0i0\qb0{hhh}\tbl0\qb0h\bsk\bsk\bsk\bsk
\ibbu0i0\qb0{jjj}\tbu0\qb0j\en
\NOTes\loffset{0.5}{\ibbl0k9}\roffset{0.5}{\tbl0}\zhl h%
\loffset{0.5}{\ibbu0f9}\roffset{0.5}{\tbu0}\hu j\en
```


2.5.6 Beams that cross line breaks

Although careful typesetting can usually avoid it, occasionally a beam may need to cross a line break. If so, it must be manually terminated at the end of one line and continued in the next. This can be done by shifting beam terminations and initiations using `\roff` and/or `\loff`, or by inserting a spacing command such as `\hsk`. We give an example from GRIEG's "Hochzeit auf Troidhaugen":

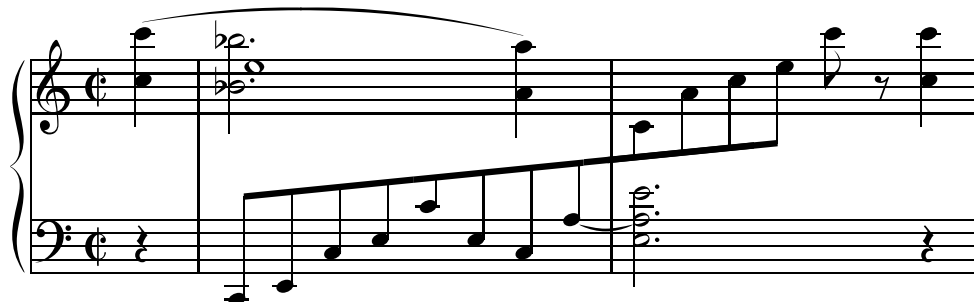


The prolongation of the two upper beam at the end is illustrated in the code fragment

```
\notes\rlap{\qs}\hsk\tbu0\|rq e\zq d\zqb1N\hsk\tbu1\en
```

2.5.7 Beams with notes on several different staves

Here's a simple example from BRAHMS's Intermezzo op. 118,1 provided by Miguel FILGUEIRAS:



The coding is

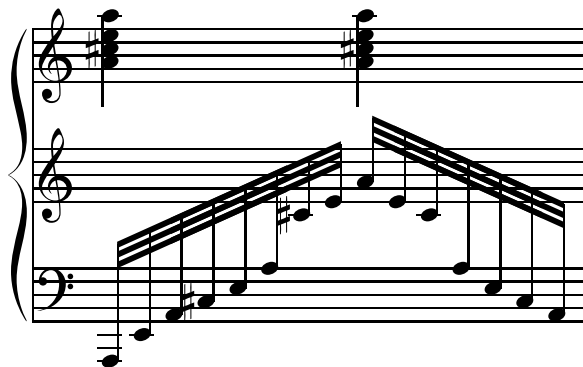
```
\interstaff{13}
\instrumentnumber{1}
\setstaves1{2}
\setclef1\bass
\generalmeter\allabreve
\startextract
\Notes\qp\nextstaff\isluru0q\zq{q}\ql{j}\enotes
\bar
\nspace
\Notes\ibu0a1\qb0{CEJLcL}%
  \nextstaff\roff{\zw{1}}\pt{p}\zh{p}\pt{i}\hl{i}\enotes
\Notes\qb0J\itied1a\qb0a\nextstaff\tslur0o\zq{o}\ql{h}\enotes
\bar
\Notes\ttie1\zh{.L.a}\hl{.e}%
  \nextstaff\qb0{chj}\tb10\qb01\cl{q}\ds\enotes
\Notes\qp\nextstaff\zq{q}\ql{j}\enotes
\endextract
```

(This example also shows that there is no problem in extending a beam across a bar line.)

The general features that enable this type of coding as well as the more complex example to follow are

- Commands like `\ibu`, `\ibl`, `\Ibu`, and `\Ibl` define beams whose initial vertical position and slope are fixed relative to the staff where they begin, but notes in other staves can still be connected to them using `\qb{n}`.
- The commands `\tbu{n}` or `\tbl{n}` terminate beam n at the specified position, but MusiX_{TEX} remembers the beam parameters until a new beam with the same number is defined. Therefore, even after beam n has been “finished” by a `\tbu` or `\tbl` command, commands like `\qb{n}{p}` will still generate notes connected to the phantom extension of this beam, *provided they are issued in a different staff*. If the command `\qb{n}{p}` were issued on the same staff as the beam after the beam had ended, an error would result.
- If the beam is initiated on one staff, notes in a lower staff can be connected to it, but only *after* the beam has been defined. This may require using the command `\prevstaff` to go back one staff, as described in section 2.2.3.

Here is an example:



which is coded as:

```

\setstaves13
\setclef1{6000}
\startextract
\notes
\nextstaff\Ibbu0Ae7\prevstaff
\qb0{AEH^JLa}\relax\nextstaff
\qb0{*****^c}\tqh0e\relax
|\zq{h^jl}\ql o\notes
\notes
\nextstaff
\Ibbu0hH6\qb0{hec}\prevstaff
\qb0{***aLJ}\tqh0H\relax\nextstaff
|\zq{h^jl}\ql o\notes \nspace
\endextract

```

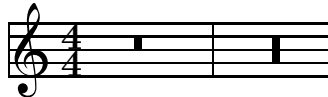
In this example we not only see multiple uses of `\nextstaff` and `\prevstaff`, but also the character `*` to make virtual beam notes (see section 2.4.7).

2.6 Rests

2.6.1 Ordinary rests

A separate macro is defined for each kind of ordinary rest. They cause a space after the symbol, just like spacing note commands, but they have no parameters. A whole rest is coded as `\pause`, dotted whole rest `\pausep`, half rest `\hpause`, dotted half rest `\hpausep`, quarter rest `\qp` or `\soupir`, eighth rest `\ds`, sixteenth rest `\qs`, 32nd rest `\hs`, and 64th rest `\qqs`.

Longer rests, normally interpreted as lasting two or four bars respectively, can be coded as `\PAuse` and `\PAUSE`, which yield:



2.6.2 Raising rests

All the previous rests except `\pausep` and `\hpausep` are *hboxes*, which means that they can be vertically offset if needed using the standard T_EX command `\raise`. For example:

```
\raise 2\Interligne\qp
\raise 3mm\qq
```

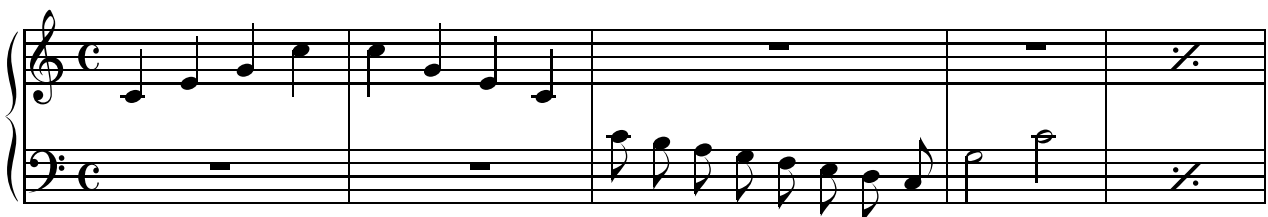
where `\Interligne` is the distance from one staff line to the next.

In addition, two macros are available to put a whole or half rest above or below the staff. The ordinary `\pause` or `\hpause` cannot be used outside the staff because a short horizontal line must be added to distinguish between the whole and the half rest. The commands, which are non-spacing⁹, are

- `\liftpause n` to get a $\overline{\text{—}}$ raised from original position by n staff line intervals,
- `\lifthpause n` to get $\overline{\text{—}}$ raised the same way.
- `\liftpausep n` to get a $\overline{\text{—}}$ raised from original position by n staff line intervals,
- `\lifthpausep n` to get $\overline{\text{—}}$ raised the same way.

2.6.3 Bar centered rests

Sometimes it is necessary to place a rest (or any other symbol) exactly in the middle of a bar. This can be done with combinations of the commands `\atnextbar`, `\centerbar`, `\centerPAUSE`, `\centerPAuse`, `\centerpause`, `\centerhpause`, as demonstrated in the following example:



with the coding

⁹Editor's note: The reason for having defined these as non-spacing is not obvious

```

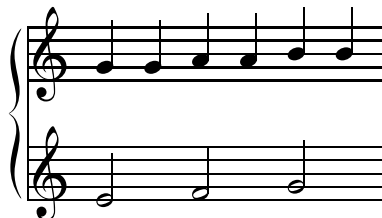
\generalmeter\meterC
\setclef1\bass
\setstaf1{2}
\parindent0pt
\startpiece\addspace\afterruleskip
\Notes|\qa{cegj}\en
\def\atnextbar{\znotes\centerpause\en}\bar
\Notes|\qa{jgec}\en
\def\atnextbar{\znotes\centerpause\en}\bar
\Notes\ca{'jihfedc}\en
\def\atnextbar{\znotes|\centerpause\en}\bar
\Notes\ha{Nc}\en
\def\atnextbar{\znotes|\centerpause\en}\bar
\addspace{10\elemskip}%
\def\atnextbar{\znotes\centerbar{\duevolte}|\centerbar{\duevolte}\en}\endpiece

```

2.7 Skipping spaces and shifting symbols

We've already mentioned that when coding a sequence of notes inside a particular pair `\notes... \enotes`, the command `\sk` can be used to skip horizontally by one `\noteskip`. This would be used for example to align the third note in one staff with the second note in another. Skipping in this manner is logically equivalent to inserting blank space; as such, the space must be recorded by MusiXTEX. This command and the others discussed here will do just that, so that `musixflx` can properly account for the added space.

To skip by one `noteskip` while in a collective coding sequence, you may simply insert an asterisk (“*”). This would have the same effect as stopping the sequence, entering `\sk`, then restarting. For example,



was coded as `\Notes\hu{e*f*g}|\qu{gghii}\en`

To skip forward by one half of a `noteskip`, use `\hsk`. To insert spacing of approximately one note head width, you can use `\qsk`, or for half of that, `\hqsk`. To skip backward by one `noteskip`, use `\bsk`. More generally, to skip an arbitrary distance, use `\off{D}` where D is any *scalable dimension*, e.g. `noteskip` or `elemskip`. Indeed, if you look in the MusiXTEX source, you will see that `\off` is the basic control sequence used to define all the other skip commands.

The foregoing commands only work *inside* a `\notes... \enotes` group. A different set of commands must be used to insert space *outside* such a group. `\nspace` produces an additional spacing of half a note head width; `\qspace`, one note head width. These are “hard” spacings. To insert an arbitrary amount of hard space outside a `\notes... \enotes` group, use `\hardspace{d}{...}` where d is any fixed dimension. The foregoing three commands are the only space-generating commands that insert hard space; all the others insert scalable spacing: `elemskip`, `beforeruleskip`, `afterruleskip`, `noteskip` and their multiples. Finally, to insert scalable spacing outside a `\notes... \enotes` group, use `\addspace{D}`. The argument may be negative, in which case the normal spacing will be reduced. For example, after `\changecontext`, many users prefer to reduce the space with a command like `addspace{-\afterruleskip}`.

There is yet another set of commands for simply shifting a note, symbol, or sequence inside `\notes... \enotes` without adding or subtracting any space. To shift by one note head width, you may write `\roff{any macro}` or `\loff{...}` for a right or left shift respectively. To shift by half of a note head width, use `\hroff{...}` or `\hloff{...}`. For example, to get



you would code:

```
\Notes\roff{\zwh g}\qu g\qu h\qu i\enotes
```

To shift notes or symbols by an arbitrary amount, use `\roffset{N}{...}` or `\loffset{N}{...}`, where N is the distance to be shifted in note head widths. For example



was coded as

```
\Notes\roffset{1.5}{\zwh g}\qu g\qu h\qu i\enotes
```

An important feature of these shift commands is that the offset, whether implicit or explicit, is *not* added to the total spacing amount, but any spacing due to the included commands is.

2.8 Accidentals

Accidentals can be introduced in two ways.

The first way, using explicit macros, consists for example in coding `\fl{p}` to put a *flat* at the pitch p , presumably right before a note at the same pitch. This is a non-spacing command and will automatically place the accidental an appropriate distance to the left of the anticipated note head. Naturals, sharps, double flats and double sharps are coded `\na{p}`, `\sh{p}`, `\df1{p}` and `\dsh{p}` respectively.

The alternate macros `\lfl`, `\lna`, `\lsh`, `\ldf1` and `\ldsh` place the same accidentals, but shifted one note head width to the left. These can be used if a note head has been shifted to the left, or to avoid collision with other accidentals in a chord. If you want to shift an accidental by some other amount for more precise positioning, you could use `\loffset` with the normal accidental macro as the second parameter.

The second way of coding accidentals is to modify the parameter of a note command. Just put the symbol `^` for a sharp, `_` for a flat, `=` for a natural, `>` for a double sharp, or `<` for a double flat, right before the letter or number representing the pitch. For example, `\qb{^g}` yields a G^\sharp . This may be used effectively in collective coding, e.g. `\qu{ac^d}`.

There are two sizes of accidentals. By default they will be large unless there is not enough space between notes, in which case they will be made small. Either size can be forced locally by coding `\bigfl`, `\bigsh`, etc., or `\smallfl`, `\smallsh`, etc. If you want all accidentals to be large, then declare `\bigaccid` near the top of the input file. For exclusively small ones use `\smallaccid`. `\varaccid` will restore variable sizes.

For editorial purposes, small accidentals can be placed *above* note heads. This is done using `\uppersh{p}`, `\upperna{p}`, or `\upperfl{p}`:



It also possible to introduce *cautionary accidentals*, i.e. small accidentals enclosed in parentheses. This done by preceding the name of the accidental keyword with “c”, e.g. `\cf1{p}` for a cautionary flat. Available cautionary accidentals are `\csh`, `\cf1`, `\cna`, `\cdf1` and `\cdsh`, which give



The distance between all notes and accidentals is controlled by `\accshift=any` *TEX* dimension, where positive values shift to the left and negative to right, with a default value of `0pt`.

2.9 Transposition and octaviation

Two different subjects are discussed in this section. First, there are commands that cause notes to be printed at different pitches than entered. We shall refer to this as *logical* transposition. Second, there are notations for octaviation that do not otherwise alter the appearance of the score, which we’ll call *octaviation lines*.

2.9.1 Logical transposition and octaviation

Logical transposition is controlled by an integer-valued *TEX* register `\transpose`. Its default value is 0. If you enter `\transpose=n`, then all subsequent pitches specified by letters will be transposed by n positions. Normally this method would be used to transpose an entire piece. Pitches specified with numbers will not be affected, so if you think you will ever want to transpose a piece, you should enter all note pitches with letters.

One way to transpose up or down by one octave would be to set `\transpose` to 7 or -7 . For example, to make a quarter note octave as a chord, you could define a macro as `\def\soqu#1{\zq{#1}{\transpose=7 \qu{#1}}}`. Note that because `\transpose` is altered inside a pair of braces, the effect of the alteration is only local and does not reach outside the braces.

Another more convenient way to transpose locally up or down by one octave makes use respectively of the characters ’ (*acute accent*) and ‘ (*grave accent*), placed immediately before the letter specifying the pitch. So for example `\qu{’ab}` is equivalent to `\qu{hi}` and `\qu{’kl}` is equivalent to `\qu{de}`. These characters have cumulative effects but in a somewhat restricted sense. They will alter the value of `\transpose`, but only until changing to a different staff or instrument or encountering `\enotes`, at which time it will be reset to the value it had before the accents were used. (That value is stored in another register called `\normaltranspose`). Thus for example `\qu{’’A’A}` and `\qu{’’A}\qu{’A}` are both equivalent to `\qu{ah}`.

At any point it is possible to reset the `\transpose` register explicitly to the value it had when entering `\notes`, by prefacing a pitch indication with “!”. Thus `\qu{!a’a}` always gives the note `a` and its upper octave `h`, shifted by the value of `\transpose` at the beginning of the current `\notes... \enotes` group, regardless of the number of grave and acute accents occurring previously within that group.

2.9.2 Behavior of accidentals under logical transposition

The above processes indeed change the vertical position of the note heads and associated symbols (e.g. stems and beams), but they don’t take care of the necessary changes of accidentals when transposing. For example, suppose an $F\sharp$ occurs in the key of C major. If the piece is transposed up three steps to the key of F, the $F\sharp$ should logically become a $B\flat$. But if all you do

is set `\transpose=3`, the note will be typeset as a B \sharp . In other words, MusiX \TeX will interpret the `\sh` or `^` to mean “print a \sharp ”.

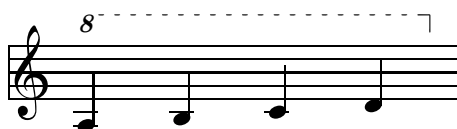
Naturally there is a solution, but it requires the typesetter to plan ahead: To force accidentals to behave well under transposition, they must be entered according to the *relative accidental convention*. To alert MusiX \TeX that you are using this convention to enter notes, you must issue the command `\relativeaccid`. Once you have done this, the meaning of accidental macros and characters (accents) in the input file is changed. Under the convention, when for example a `\sh` is entered, it indicates a note that is supposed to sound *one half step higher than what it would normally be under the current key signature*. Flats and naturals on entry similarly indicate notes one half step lower or at the same pitch as what the key signature dictates. MusiX \TeX will take account of the key signature, and print the correct symbol according to the modern notational convention, provided you have explicitly entered the transposed key signature using for example `\generalsignature`.

Many people have a difficult time understanding how this works, so here are two simple examples in great detail. Consider the case already mentioned of the F \sharp in the key of C major. With `\relativeaccid` in effect, it should still be entered as `\sh f`, and with no transposition it will still appear as F \sharp . With `\transpose=3` and `\generalsignature{-1}` it will appear (correctly) as B \natural . Conversely, suppose you want to enter a B \natural when originally in the key of F. With `\relativeaccid` in effect, it should be entered as `\sh i`. (That’s the part that people have the most trouble with: “If I want a natural, why do I have to enter a sharp?” Answer: “Go back and re-read the previous paragraph very carefully.”) With no transposition, it will be printed as B \natural . Now to transpose this to C major, set `\transpose=-3` and `\generalsignature0`, and it will appear as F \sharp .

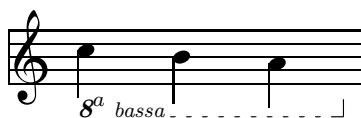
If you have invoked `\relativeaccid` and then later for some reason wish to revert to the ordinary convention, enter `\absoluteaccid`.

2.9.3 Octaviation lines

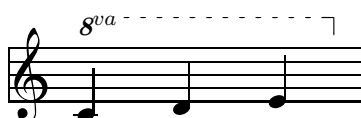
The first kind of notation for octave transposition covers a horizontal range that must be specified at the outset. The sequence



can be coded as `\NOTES\octfinup{10}{3.5}\qu a\qu b\qu c\qu d\en`. Here, the dashed line is at staff level 10 and extends 3.5\noteskip. Conversely, lower octaviation can be coded. For example



is coded as `\NOTES\octfindown{-5}{2.6}\ql j\ql i\ql h\en`. To change the text that is part of these notations, redefine one of the macros `\octnumberup` or `\octnumberdown`. The reason for the distinction between up and down is that, traditionally, upper octaviation only uses the figure “8” to denote its beginning, while lower octaviation uses a more elaborate indication such as *8^{va} bassa*. Thus



is coded

```
\NOTES\def\octnumberup{\ppffsixteen8$^{va}$}\octfinup{10}{2.5}\qu c\qu d\qu e\en
```

while



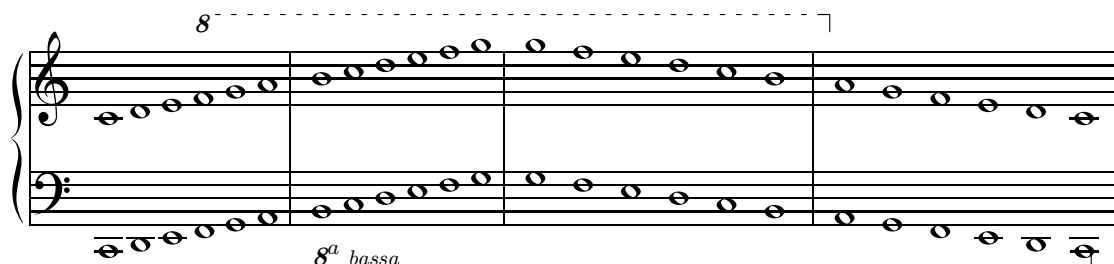
is coded as

```
\NOTES\def\octnumberdown{\ppffsixteen8$_{ba}$}%
\octfindown{-5}{2.5}\ql l\ql k\ql j\en
```

The foregoing constructions have the drawbacks that (a) the span must be indicated ahead of time and (b) they cannot extend across a line break. Both restrictions are removed with the use of the alternate macros `\Ioctfinup`, `\Ioctfindown` and `\Toctfin`.

`\Ioctfinup np` indicate an upward octave transposition line with reference number n ($0 \leq n < 6$) and with dashed line at pitch p . Usually p will be numeric and > 9 , but it can also be a letter. `\Ioctfindown np` starts a lower octave transposition line at pitch p (usually $p < -1$). Both extend until terminated with `\Toctfin`. The difference between `\Ioctfinup n` and `\Ioctfindown n` is the relative position of the figure “8” with respect to the dashed line, and the sense of the terminating hook. As shorthand, `\ioctfinup` is equivalent to `\Ioctfinup 0` and `\ioctfindown` is equivalent to `\Ioctfindown 0`.

For example,



is coded as

```
\begin{music}
\instrumentnumber{1}
\setstaves12
\setclef1{6000}
%
\startextract
\notes\wh{CDEFGH}|\wh{cde}\Ioctfinup 1p\wh{fgh}\enotes
\bar
\notes\Ioctfindown 2A\wh{IJKLMN}|\wh{ijklmn}\enotes
\bar
\Notes\wh{NMLKJI}|\wh{nmlkji}\Toctfin1\enotes
\bar
\Notes\wh{HGFED}\Toctfin2\wh C|\wh{hgfedc}\enotes
\endextract
\end{music}
```

The elevation of octaviation lines may be changed in midstream using `\Liftoctline np`, where n is the reference number of the octave line, and p a (possibly negative) number of `\internotes` (staff pitch positions) by which elevation of the dashed line should be changed. This may be useful when octaviation lines extend over several systems and the elevation needs to be changed in a systems after the one where it was initiated.

2.10 Slurs and ties

Two fundamentally different implementations of slurs, ties, and hairpins are available. (The hairpins “go along for the ride” with slurs and ties.) First, there are the original *font-based* versions. These are constructed with traditional T_EX font characters that were created with METAFONT and stored in T_EX font files. Second, it is now possible to generate these shapes directly with postscript, dispensing altogether with the font characters¹⁰. We shall first describe the font-based versions, then Type K postscript slurs, which are one of two available postscript slur options¹¹. If you plan to use Type K postscript slurs, you may skip directly to section 2.10.2.

2.10.1 Font-based slurs

Font-based slurs and ties provided by MusiX_TE_X can be divided into two categories:

- Those where the complete slur symbol is composed of a single character from one of the slur fonts, and
- those where the slur symbol is composed of three distinct characters, to form the beginning, middle and end of the slur.

The former are called *simple slurs* and the latter, *compound slurs*. In many cases the distinction between the two is invisible to the user, in that many of the macros described below will automatically select between the two types. However, there are other macros that allow simple slurs to be forced.

The next few subsections describe the usual method of slur coding, where the choice between simple or compound slurs is made automatically. In this case, slurs are initiated and terminated by separate macros, similar to beams.

2.10.1.1 Font-based slur initiation

A slur must be initiated *before* the spacing note on which the slur begins, and terminated *before* the note on which it ends. The basic slur initiation macro is `\isluru{n}{p}`, which initiates an upper slur, with reference number n , beginning on a note at pitch p . The starting point of the slur is centered above a virtual quarter note head at pitch p ¹². As with beams, the reference number n can take values from 0 to 5, or up to 8 or 11 respectively if `musixadd.tex` or `musixmad.tex` is included. Similarly, `\islurd{n}{p}` initiates a lower slur. These slurs are terminated by coding `\tslur{n}{p}` where n is the reference number and p is the termination pitch. To illustrate with an elementary example, the following passage



was coded as:

```
\N0tes\islurd0g\qu g\tslur0{'c}\qu c\en
\Notes\isluru0{'e}\ibl0e{-2}\qb0{edc}\tslur0b\tqb0b\en
\bar
```

¹⁰Please do not be confused by the availability of postscript versions of the font-based slur fonts (along with all other MusiX_TE_X fonts). Once installed in a T_EX system, their function and use are 100% transparently identical with bitmapped versions of the slur fonts. On the other hand, postscript slurs are functionally distinct from font-based slurs, and only share some of the same syntax.

¹¹An alternate approach to postscript slurs, called *Type M* after its developer Hiroaki MORIMOTO, is available from the [Icking Music Archive](#).

¹²The slur will start in the same place regardless of whether there is *actually* a note at pitch p .

```
\NOTes\islurd0{'a}\qu a\tslur0{'f}\qu f\en
\NOTes\hu g\en
```

Other macros are provided to change the starting and ending point of the slur in relation to the initial and final notes. Thus, `\issluru{n}{p}` initiates a “short” upper slur suitable for linking notes involved in chords. The starting point is shifted to the right, and is vertically aligned with the center of a virtual quarter note head at pitch p . If a lower short slur is wanted, one should use `\isslurd{n}{p}`.

Sometimes, busy scores call for slurs which are vertically aligned with the ends of note stems rather than note heads. These “beam” slurs—so called because the slur is written at usual beam height—are provided by the macros `\ibsluru{n}{p}` and `\ibslurd{n}{p}`. These macros initiate slurs raised or lowered by the current stem height to accommodate stems or beams above or below.

2.10.1.2 Font-based slur termination

Font-based slurs that are not forced to be simple must be terminated by an explicit command right before the last note under the slur. There are termination commands analogous to each of the initiation commands already presented. They are summarized in the following table:

Initiation	Termination
<code>\isluru</code> , <code>\islurd</code>	<code>\tslur</code>
<code>\issluru</code> , <code>\isslurd</code>	<code>\tsslur</code>
<code>\ibsluru</code>	<code>\tbsluru</code>
<code>\ibslurd</code>	<code>\tbslurd</code>

All of these command have two parameters, n and p .

These specific termination macros are not restricted to being used with their initiation counterpart. A slur started in one sense can be terminated in another. For example, a slur beginning as a “beam” slur may be terminated as a normal slur. This would be achieved using the macro pair `\ibslur... \tslur`.

2.10.1.3 Font-based ties

Font-based ties will have the same shapes as ordinary font-based slurs of the same length between notes of equal pitch, but there are two important distinctions: (1) There cannot be any pitch difference between start and end, and (2) the positions of both the beginning and end of a tie relative to the note heads are slightly different from those of an ordinary slur¹³. Upper ties are initiated by `\itieu{n}{p}`, which starts an upper tie of reference number n at pitch p . Lower ties are initiated by `\itied{n}{p}`, which starts an lower tie of reference number n at pitch p . The starting position of the tie is the same as `\issluru` and `\isslurd` respectively. The tie is terminated by coding `\ttie{n}`. Note that no pitch parameter is required.

There are also *short ties*, which bear the same relation to ordinary ties as short slurs to ordinary slurs. They are intended to be used between chords. They are initiated with `\itenu{n}{p}` or `\itenl{n}{p}`¹⁴, and terminated with `\tten{n}`.

The following example illustrates the differences in positioning of the various slur and tie options:



¹³Editor’s note: In fact, it appears that the default positioning of the ends of ties is exactly the same as that of short slurs.

¹⁴Editor’s note: It is not clear why this command uses “1” when all other similar ones use “d”.

It was coded as

```
\NOTes\islurd0g\qu g\tslur0g\qu g\isslurd0g\qu g\tssslur0g\qu g%
\ibsluru0g\qu g\tbsluru0g\qu g\itied0g\qu g\ttie0\qu g%
\itenl0g\qu g\tten0\qu g\en
```

Here are some more general examples of font-based slurs and ties discussed so far:



This was coded as:

```
\NOTes\isluru0g\hl g\tslur0h\hl h\en
\NOTes\islurd0c\issluru1g\zh{ce}\hu g\tslur0d\tssslur1h\zh{df}\hu h\en
\NOTes\ibsluru0g\islurd1g\hu g\tubslur0h\hu h\en
\NOTes\itieu0k\hl k\ttie0\tdbslur1f\hl k\en
```

2.10.1.4 Dotted slurs

Any font-based slur may be made dotted by specifying `\dotted` just before it is initiated:



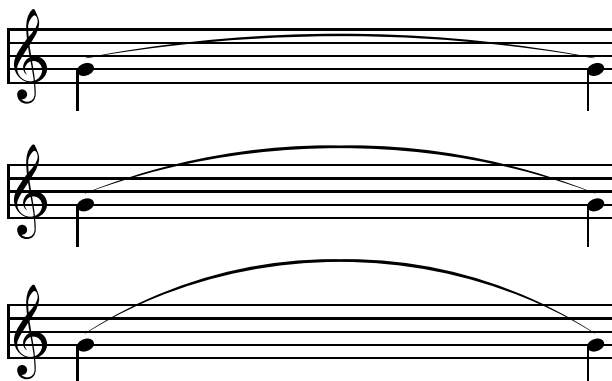
This was coded as:

```
\NOTes\dotted\islurd0g\qu g\tslur0{'c}\qu c\en
\Notes\dotted\isluru0{'e}\ibl0e{-2}\qb0{edc}\tslur0b\tqb0b\en\bar
\NOTes\dotted\slur{'a}{'f}d1\qu{'a'f}\en
\NOTes\hu g\en
```

2.10.1.5 Modifying font-based slur properties

Several macros are provided to modify the shape of slurs already initiated. These macros must be coded right before the slur *termination*. Invoking any of the macros described in this section will force the slur to be compound.

By default, the midpoint of a font-based slur is three `\internotes` above or below a line between its ends. This can be changed using the macro `\midslur h` where `h` is the revised vertical displacement. For example, `\midslur6` coded right before `\tslur` causes an upper slur to rise to a maximum height of 6 `\internote` above the starting position. For example,



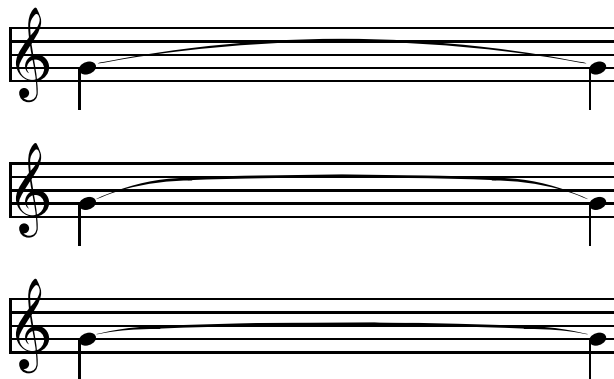
was coded as

```

\Notes\multnoteskip8\isluru0g\ql g\en
\notes\tslur0g\ql g\en
\Notes\multnoteskip8\isluru0g\ql g\en
\notes\midslur7\tslur0g\ql g\en
\Notes\multnoteskip8\isluru2g\ql g\en
\notes\midslur{11}\tslur2g\ql g\en

```

The macro `\curve hij` allows more precise control over the shape of a slur or tie. The first parameter h is the vertical deviation and it works exactly like the sole parameter of `\midslur` described above. The second and third parameters i and j set the initial and final gradient respectively. They are defined as the horizontal distance required to attain maximum vertical deviation. Thus smaller numbers for i and j lead to more extreme gradients. The default setting is `\curve344`. Hence, coding `\curve322` doubles the initial and final gradient relative to the default. As with `\midslur`, `\curve` must be coded *immediately* before the slur termination. The example below illustrates the use of `\curve`.



This was coded as

```

\Notes\multnoteskip8\itieu0g\ql g\en
\notes\ttie0\ql g\en
\endextract
\startextract
\Notes\multnoteskip8\itieu1g\ql g\en
\notes\curve 322\ttie1\ql g\en
\endextract
\startextract
\Notes\multnoteskip8\itieu2g\ql g\en
\notes\curve 111\ttie2\ql g\en

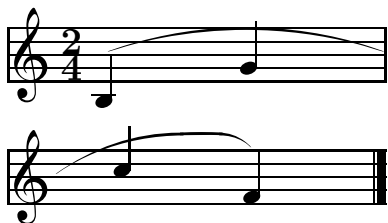
```

Two macros are provided to control the behaviour of slurs which extend across line breaks. Normally, the part of the slur before the line break is treated as a tie. This can be changed using `\breakslur n { p }`, which sets the termination height of the broken slur at the line break to pitch p , for slur number n .

After the line break, the slur is normally resumed at the initial pitch reference, i.e., the one coded in `\islur`. To change this, the macro `\Liftslur n { h }` may be used. Here n is again the slur reference number and h is the change in height relative to the initialization height. This macro is normally used following line breaks, in which case it is best coded using the `\atnextline` macro. For example, coding `\def\atnextline{\Liftslur06}` raises the continuation of slur zero by $6\backslashinternote$ relative to its initialization height.

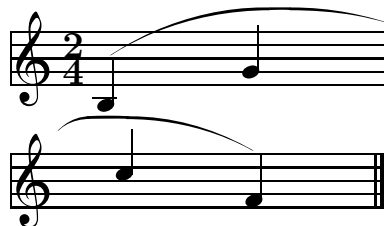
The following example illustrates the use of the macros for broken slurs:

Default, without adjustments:



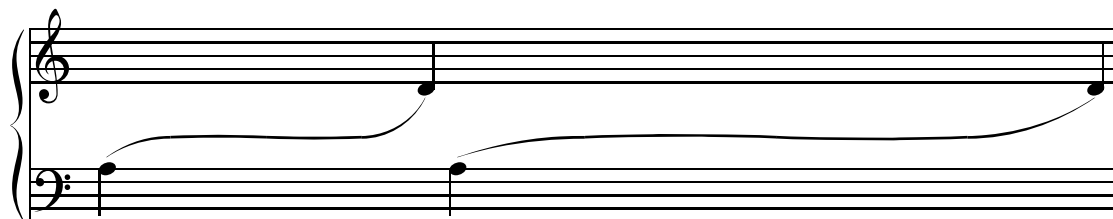
```
\Notes\ibsluru1b\qu b\qu g\en\bar
\Notes\qu{'c!}\tslur1f\qu f\en
```

With `\Liftslur` and `\breakslur`:



```
\def\atnextline{\Liftslur17}%
\Notes\ibsluru1b\qu b\qu g%
\breakslur1g\en\bar
\Notes\qu{'c!}\tslur1f\qu f\en
```

Occasionally in keyboard works one needs to begin a slur in one staff but end it in another. This can be done using the macro `\invertslur{n}` which is best described by reference to the example shown below.



This was coded as

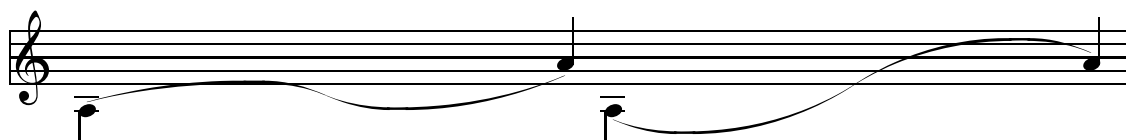
```
\Notes\multnoteskip5\isluru0a\ql a\en
\notes\invertslur0\curve311\tslur0g|\qu d\en
\Notes\multnoteskip{10}\isluru0a\ql a\en
\notes\invertslur0\curve333\tslur0g|\qu d\en
```

Slur inversion as just described takes effect where the slope is zero; therefore it only works with ascending slurs that were started with `\isluru`, and with descending slurs started with `\islurd`. Otherwise no horizontal place can be found and the result is erratic.

A different approach removes this restriction. The idea is to stop the slur at a the desired inversion point and restart it in the other sense at the same place. The commands to do this are as follows:

- `\Tslurbreak{n}{p}` stops slur number n exactly at pitch p , not above or below the virtual note head.
- `\Islurubreak{n}{p}` restarts an upper slur at the same position, not above a virtual note head.
- `\Islurdbreak{n}{p}` restarts a lower slur at the same position, not below a virtual note head.

The vertical position may have to be adjusted to minimize any discontinuity in the slope. For example, the following pattern



was coded as

```
\begin{music}
```

```

\NOTes\multnoteskip 3\isluru0a\ql a\en
\NOTes\multnoteskip 3\Tslurbreak0d\Islurdbreak0d\sk\en
\Notes\tslur0h\qu h\en
\NOTes\multnoteskip 3\islurd0a\ql a\en
\NOTes\multnoteskip 3\Tslurbreak0d\Islurubreak0d\sk\en
\Notes\tslur0h\qu h\en
\end{music}

```

Simple slurs and ties have advantages in some cases: (1) They will always have the best possible shape, and (2) if `\noteskip` doesn't change from start to finish, they are easier to code. But they have drawbacks as well: (1) They are limited in length to 68pt for slurs and 220pt for ties, (2) their maximum vertical extent is `8\internote`, and (3) they may not extend across a line break. Despite all these limitations, simple slurs are extremely useful in many applications where the slurs are short and contained within a bar.

Simple slurs must be coded *before* the note on which the slur begins. The primary macro is `\slur{p1}{p2}sl` where p_1 and p_2 are respectively the initial and final pitches, s is the sense (either “u” or “d”), and l is the length, in `noteskips`. Thus, thirds slured in pairs can be coded

```

\NOTes\slur ced1\qu{ce}\en
\NOTes\slur dfd1\qu{df}\en
\NOTes\slur egd1\qu{eg}\en
\NOTes\slur{'e}cu1\ql{ec}\en
\NOTes\slur{'d}bu1\ql{db}\en
\NOTes\slur{'c}au1\ql{ca}\en

```

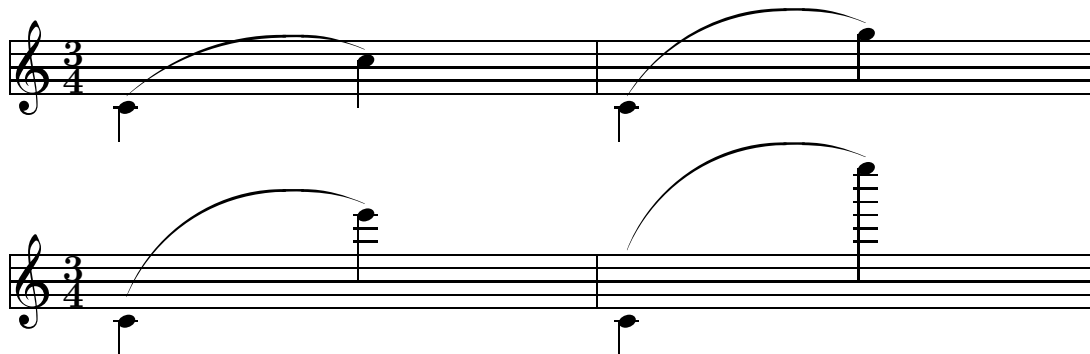
which yields



There are similar commands to force simple versions of the other variants of font-based slurs and ties. Simple ties may be set using `\tie{p}sl` (only one pitch is needed). Simple short slurs and short ties can be forced with the macros `\sslur{p1}{p2}sl` and `\stie{p}sl` respectively. Finally, simple beam slurs can be forced with `\bslurs{p1}{p2}l`.

2.10.1.6 Limitations of font-based slurs

The change in altitude between slur initiation and slur termination is limited to `16\Internote`. Thus unexpected vertical gaps can appear, as seen in



whose coding was

```

\NOTes\multnoteskip3\isluru0c\ql c\tslur0j\ql j\enotes
\bar\NOTes\multnoteskip3\isluru0c\ql c\tslur0n\ql n\enotes

```

```

\NOTes\multnoteskip3\isluru0c\ql c\tslur0s\ql s\enotes
\bar\NOTes\multnoteskip3\isluru0c\ql c\tslur0z\ql z\enotes
\end{music}

```

Furthermore, if the slope becomes too steep, even worse results can occur, such as



Another limitation of font-based slurs crops up when one attempts to generate bitmapped versions for high-resolution printers: the characters for long ties can exceed METAFONT’s maximum capacity.

2.10.2 Type K postscript slurs

All of the aforementioned limitations of font-based slurs can be circumvented by using Type K postscript slurs¹⁵. As well as slurs, the package includes ties and crescendos (see section 2.17.6). Its use is very similar to font-based slurs, and in fact identical if only the elementary slur and tie initiation and termination macros are used.

In order to use Type K postscript slurs, you must first place `musixps.tex` anywhere \TeX can find it. You must also place `pslurs.pro` somewhere that `dvips` can find it.

The `mxsk` font is required for “half ties,” which are special symbols that are used by default for the second portion of a tie that crosses a line break. If you like this treatment you must install the font in your \TeX system. However, perfectly acceptable line-breaking ties will appear if you invoke `\nohalfties`, and then you will not have to install this font.

Once the software mentioned in the prior two paragraphs is emplaced and the \TeX file-name database is refreshed, the Type K package can be invoked by including the command `\input musixps` near the beginning of your source file. The resulting dvi file should then be converted into postscript using `dvips`. If desired, a PDF file can then be generated with `ps2pdf`, `ghostscript`, **Adobe Acrobat** (see chapter 3 for more information on this).

Two minor inconveniences with Type K postscript slurs are that (1) they won’t appear in any standard dvi previewer, and (2) they won’t appear in PDF files generated with `pdftex`. The former limitation can be circumvented by using a postscript viewer such as **GSview**. The latter simply requires that you create an intermediate postscript file with `dvips`, then make the PDF with any of the software mentioned above.

2.10.2.1 Initiating and terminating type K postscript slurs

Basic usage of type K slurs is the same as for font-based slurs. To initiate one, use for example `\isluru0g` to start an upper slur with ID 0 above a virtual note at pitch level `g`. To terminate one, use a command like `\tslur0i` which terminates the slur with ID 0 on a virtual note at pitch level `i`. Both types of commands are non-spacing and must precede the first or last note under the slur.

You can shift the starting or ending point slightly to the left or right by substituting one of the commands `\ilsluru`, `\ilslurd`, `\irsluru`, `\irslurd`, `\trslur` or `\tislur`.

You can control the shape of Type K slurs with variants of the termination command. To make the slur a bit flatter than default use `\tflslur0f`; a bit higher, `\thslur0f`; higher still, `\tHslur0f`;

¹⁵ “K” stands for Stanislav Kneifl, the developer of the Type K postscript slur package.

or even higher, `\trHHslur0f`. These commands have an effect like `\midslur` does for font-based slurs.

All combinations of the shifting and curvature variants are allowed, e.g. `\trHHslur`.

The following examples demonstrate how much better the type K slurs perform in the extreme situations of the prior two typeset examples. The coding is exactly the same as above except that `\input musixps` has been added:

```
\begin{music}
\input musixps
\startextract\NOTes\multnoteskip3\isluru0c\ql c\tslur0s\ql s\enotes
....
```



For maximal control over type K slurs, you can use one of the commands `\iSlur npvh` and `\tSlur npvhca`, where the characters in *npvhca* respectively stand for ID number, height, vertical offset, horizontal offset, curvature, and angularity. All offsets are in *internote*, and the slur direction is determined by the sign of the vertical offset. See the comments in `musixps.tex` for precise definitions of the other parameters. Examples of permissible forms for these commands are `iSlur0c11` and `tSlur0{!d}11{.2}0`.

The next example shows how you can use `iSlur` in difficult circumstances:



with this coding:

```
\NOTes\irslurd0{'b}\zhl b\sk\zq{!e}\qu{'c}\en\bar
\NOTes\zh{'d}\zhu{'f}\off{7.2pt}\tSlur0{'b}\ql b\ql b\en\bar
%
\NOTes\irslurd0{'b}\zhl b\sk\zq{!e}\qu{'c}\en\bar
\NOTes\zh{'d}\zhu{'f}\off{7.2pt}\tlfslur0{'b}\ql b\ql b\en\bar
%
\NOTes\iSlur0{'b}{-.5}3\zhl b\sk\zq{!e}\qu{'c}\en\bar
\NOTes\zh{'d}\zhu{'f}\off{7.2pt}\tSlur0{'b}{.5}{-1}{.3}0\ql b\ql b\en
```

The ID number for a slur, tie or crescendo should normally range from 0 to 9. If it is bigger than nine but less than 15, the object can cross a line break but not a page break. If bigger than 14 but less than 2^{31} , it can't be broken at all, nor can a slur termination be positioned at a beam with e.g. `\tblsluru{17684}{16}`; however `\ibsluru{152867}{16}` is OK.

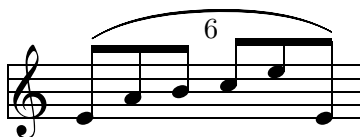
It's also OK to have opened simultaneously a slur, tie and crescendo all with the same ID, or a slur, tie and decrescendo, but not a crescendo and decrescendo.

2.10.2.2 Type K postscript beam slurs

Type K beam slurs are defined differently than the font-based ones: They require as parameters both a slur ID number n and a beam ID number m , but that's all. The commands are `\iBsluru{n}{m}`, `\iBslurd{n}{m}`, and `\tBslur{n}{m}`. They must be placed *after* the beam initiation or termination command. Type K slurs may start on one beam and end on another. For example,

```
\Notes\ibu0i0\iBsluru00\qb0{eh}\tbu0\qb0i\ibu0j0\qb0{j1}\tbu0%
\slurtext{6}\tBslur00\qb0e\en
```

produces



The above example also illustrates the use of the macro `\slurtext`. It has just one parameter—some text to be printed—and it centers it just above or below the midpoint of the next slur that is closed. This works only for non-breaking slurs; if the slur is broken, the text disappears¹⁶.

2.10.2.3 Type K postscript ties

All of the foregoing Type K slur commands have counterparts for ties. Simply replace “`slur`” with “`tie`”, and for terminations omit the pitch parameter. Type K ties not only are positioned differently by default, but they also have different shapes than slurs.

2.10.2.4 Dotted type K slurs and ties.

A slur or tie can be made dotted simply by entering `\dotted` anywhere before the beginning of the slur or tie. Only the first slur or tie following this command will be affected. On the other hand, if you enter `\Dotted`, then ALL slurs and ties from this point forward will be dotted until you say `\Solid`. Furthermore, inside `\Dotted... \Solid` you can make any individual slur or tie solid saying `\solid` before its beginning.

2.10.2.5 Avoiding collisions of Type K slurs and ties with staff lines.

In postscript it is possible to do some computations which would be very hard to implement directly in TeX. Type K slurs can use this facility to check whether the curve of a slur or tie is anywhere nearly tangent to any staff line, and if so, to adjust the altitude of the curve to avoid the collision. By default this feature is turned on. You can disable it either globally (`\Nosluradjust`, `\Notieadjust`) or locally (`\nosluradjust`, `\notieadjust`), and you can also turn it back on globally (`\Sluradjust`, `\Tieadjust`) or locally (`\sluradjust`, `\tieadjust`). Here “locally” means that the command will only affect the next slur or tie to be opened.

¹⁶If you insist on viewing files with a dvi viewer despite the fact that Type K slurs will not be visible, you may also find that figures emplaced with `\slurtext` will appear at the end of the slur rather than the middle.

2.10.2.6 Line breaking slurs and ties

Type K slurs and ties (and crescendos) going across line breaks are handled automatically. In fact they can go over more lines than two (this is true also for ties, though it would be somewhat strange).

There is a switch `\ifslopebrkslurs` that controls the default height of the end point of the first segment of all broken slurs. By default the height will be the same as the beginning. To have it raised by 3\internote , simply issue the command `\slopebrkslurstrue`. To revert to the default, use `\slopebrkslursfalse`.

To locally override the default height of the end of the first segment, use the command `\breakslur{n}{p}`, which sets the height for slur number n to pitch p , just like with font-based slurs.

You can raise or lower the starting point of the second segment of a broken slur with the command `\liftslur{n}{h}`, with parameters slur ID and relative offset in `\internotes` measured from the slur beginning. Its effect is the same as `\Liftslur` for font-based slurs, except it is not necessary to code it within `\atnextstaff{}`, just anywhere inside the slur.

As already mentioned, anything with $ID < 10$ is broken fully automatically, but you should be careful about slurs, ties and crescendos with $10 \leq ID < 15$. These cannot cross page breaks, although they can cross line breaks.

If the second segment of a broken tie is less than 15pt long, then by default it will have a special shape which begins horizontally. These shapes are called *half ties* and are contained in the font `mxsk`. Of course if they are to be used, the font files must be integrated into the `TEX` installation. Their use can be turned off with `\nohalfties` and back on with `\halfties`.

2.10.2.7 A few final technical details

Each `\i...` and `\t...` produces a `\special` command, which must be stored in TeX's main memory. Therefore, if too many slurs occur in one page, some memory problems could occur. The only solutions are to use Big`TEX` (if you aren't already), or to use font-based slurs.

Type K slurs need the postscript header file `psslurs.pro` to be included in the output postscript file. This is made to happen by the `TEX` command `\special{header=psslurs.pro}`, which is automatically included when you `\input musixps`. So normally this is not of concern. However if you wish to extract a subset of pages from the master `dvi` file using `dvidvi`, then you have three options: (1) include the first page in the subset, (2) manually issue the `\special` command in the `TEX` source for the first page of the subset, or (3) use the option `-h psslurs.pro` when you run `dvips`.

2.11 Bars Lines

2.11.1 Single, double, and invisible bar lines

The usual macro to typeset a single bar line is `\bar`. There is a possibility of confusion with a command in `TEX`'s math mode that has exactly the same name. However, there will generally be no problem, because inside `\startpiece... \endpiece`, `\bar` will take the musical meaning, while outside, it will have the mathematical one. If for some reason you need the math `\bar` inside, you can use `\endcatcodesmusic... \bar... \catcodesmusic`.

To typeset a double bar line with two thin rules, use `\doublebar`. You could also issue `\setdoublebar` to cause the next `\bar` (or `\stoppiece`, `\alaligne`, or `\alapage`) to be replaced by a double bar.

There is no specific command to print a thin-thick double bar line, but `\setdoubleBAR` will cause one in the same cases where `\setdoublebar` would cause a thin-thin double bar line.

To make the next bar line invisible, use `\setemptybar` before `\bar`.

2.11.2 Simple discontinuous bar lines

Normally, bars (as well as double bars, final bars and repeat bars) are drawn continuously from the bottom of the lowest staff of the lowest instrument to the top of the highest staff of the uppermost instrument. However, if desired, they can be made discontinuous between instruments by saying `\sepbarrules`. An example of this is given in `avemaria.tex` in section 2.19.

Continuous bar lines can be restored with `\stdbarrules`. In the extension library are some more types of bar rules, mainly for very old music, see section 2.23.15.

2.11.3 Elementary asynchronous bar lines

Situations may arise where the bar lines in different instruments are not aligned with one another. To set this up, first say `\sepbarrules`. Then use a combination of the following five commands:

- `\hidebarrule{n}` hides the bar rule for instrument n , until this is changed by `\showbarrule{n}`.
- `\showbarrule{n}` stops hiding the bar rule for instrument n , until this is changed by `\hidebarrule{n}`.
- `\Hidebarrule{n}` hides the bar rule for instrument n , only for the next bar.
- `\Showbarrule{n}` shows the bar rule for instrument n once only, and then resets it `Hidebarrule`.
- `\showallbarrules` resets all defined instruments to `\showbarrule{n}`. This command is automatically inserted with double bars, final bars and repeats.

Thus, this example



was obtained with the following coding:

```
\instrumentnumber3
\setmeter3{\meterfrac{3}{4}}
\setmeter2{\meterfrac{2}{4}}
\setmeter1{\meterfrac{3}{8}}
\nobarnumbers
\sepbarrules

\startextract
\N0tes\pt f\qa f&\qa f&\qa f\en
```

```

\hidebarrule2\hidebarrule3\bar
\Notes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\showbarrule2\bar
\Notes\pt f\qa f&\qa f&\qa f\en
\hidebarrule2\showbarrule3\bar
\Notes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\showbarrule2\hidebarrule3\bar
\Notes\pt f\qa f&\qa f&\qa f\en
\hidebarrule2\bar
\Notes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\setdoublebar
\bar\hidebarrule3
\Notes\pt f\qa f&\qa f&\qa f\en
\Hidebarrule2\bar
\Notes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\bar
\Notes\pt f\qa f&\qa f&\qa f\en
\message{Showbarrule3 coming}%
\Hidebarrule2\Showbarrule3\bar
\Notes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\bar
\Notes\pt f\qa f&\qa f&\qa f\en
\Hidebarrule2\bar
\Notes\multnoteskip{.333}\Tqbu fff&\qa f&\qa f\en
\setrightrepeat
\endextract

```

2.11.4 Dotted, dashed, and more general asynchronous and discontinuous bar lines

The extension package [musixdbr.tex](#) by Rainer DUNKER provides commands for dashed, dotted, and arbitrarily discontinuous bar lines. It supports individual bar line settings for each instrument, multi-staff instruments, different sizes of staves, and even different numbers of lines per staff,

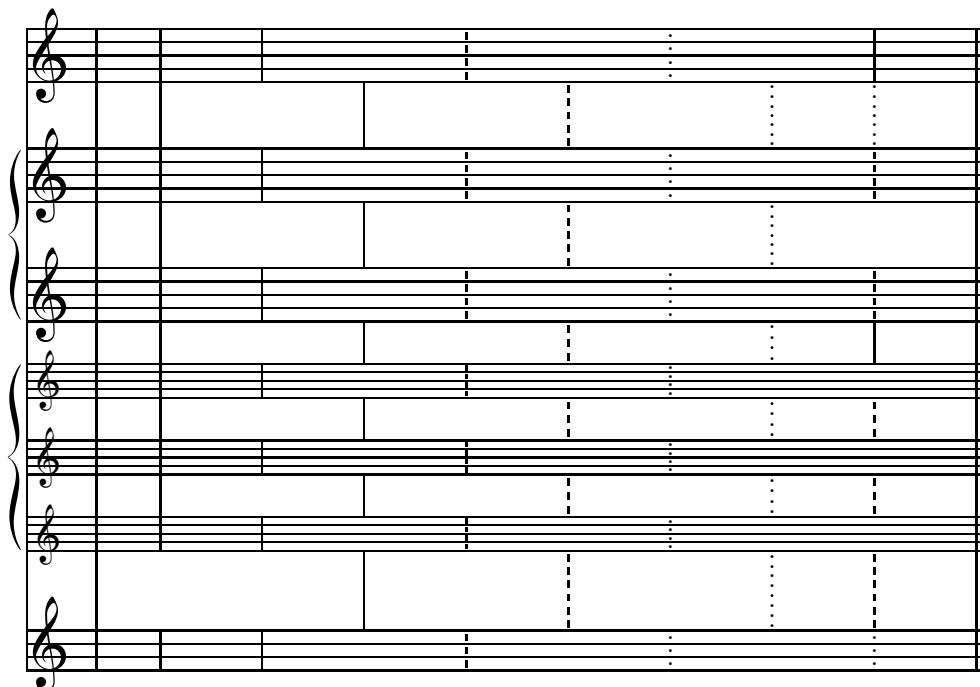
To use the package, you must `\input musixdbr` after `musixtex`, and execute the macro `\indivbarrules` which activates individual bar line processing. Then the following commands are available:

- `\sepbarrule{n}` separates bar lines of instrument n from those of instrument $n - 1$
- `\condashbarrule{n}` connects bar lines of instrument n to those of instrument $n - 1$ with a dashed line
- `\condotbarrule{n}` connects bar lines of instrument n to those of instrument $n - 1$ with a dotted line
- `\conbarrule{n}` connects bar lines of instrument n to those of instrument $n - 1$
- `\hidebarrule{n}` hides bar lines of instrument n
- `\showdashbarrule{n}` dashes bar lines of instrument n
- `\showdotbarrule{n}` dots bar lines of instrument n
- `\showbarrule{n}` shows bar lines of instrument n
- `\sepmultibarrule{n}` separates bar lines within multistaff instrument n

- `\condashmultibarrule{n}` dashes bar lines between staves of multistaff instrument n
- `\condotmultibarrule{n}` dots bar lines between staves of multistaff instrument n
- `\conmultibarrule{n}` shows bar lines between staves of multistaff instrument n
- `\allbarrules[any of the above commands, without numerical parameter]` sets bar line style for all instruments together.

Dashing and dotting style may be changed by redefining the macros `\barlinedash` or `\barlinedots` respectively (see original definitions in `musixdbr.tex`).

Here is an example of the use of these macros:



This is the code:

```
\input musixdbr
\instrumentnumber4\setstoffs23\setstoffs32\setlines14\setsize2\tinyvalue
\indivbarrules\parindent0pt\startextract
% normal barlines
\bar
% separate instrument 2 from 1
\sepbarrule2
\notes\en\bar
% barlines on staves
\allbarrules\sepbarrule
\allbarrules\sepmultibarrule
\allbarrules\showbarrule
\NOTes\en\bar
% barlines between staves
\allbarrules\conbarrule
\allbarrules\conmultibarrule
\allbarrules\hidebarrule
\NOTes\en\bar
% dashed barlines on staves
\allbarrules\sepbarrule
```

```

\allbarrules\sepmultibarrule
\allbarrules\showdashbarrule
\NOTes\en\bar
% dashed barlines between staves
\allbarrules\condashbarrule
\allbarrules\condashmultibarrule
\allbarrules\hidebarrule
\NOTes\en\bar
% dotted barlines on staves
\allbarrules\sepbarrule
\allbarrules\sepmultibarrule
\allbarrules\showdotbarrule
\NOTes\en\bar
% dotted barlines between staves
\allbarrules\condotbarrule
\allbarrules\condotmultibarrule
\allbarrules\hidebarrule
\NOTes\en\bar
% a wild mixture of all
\showdotbarrule1\hidebarrule2\showdashbarrule3\showbarrule4%
\condashbarrule2\conbarrule3\condotbarrule4%
\condashmultibarrule2\sepmultibarrule3%
\NOTes\en\bar
% conventional ending
\allbarrules\showbarrule
\allbarrules\conbarrule
\allbarrules\conmultibarrule
\NOTes\en\setdoubleBAR\endextract

```

2.12 Bar numbering

The current bar number is stored in a count register call `\barno`. When `\startpiece` is encountered, `\barno` is set equal to another count register called `\startbarno`, whose default value is one. Therefore, if you want the first bar to have a number n different from 1, you may either say `\startbarno= n` before `\startpiece`, or say `\barno= n` afterwards, but before the first bar line. You may also alter the bar number at any time, either by explicitly resetting `\barno`, or by incrementing it with a command like `\advance\barno-1`.

MusiX_{TEX} supports two distinct modes for printing bar numbers. In *periodic* bar numbering, the bar number is placed above the top staff with a user-selectable frequency. In *system* bar numbering, the number will appear at the beginning of each system.

2.12.1 Periodic bar numbering

In a normal piece, periodic bar number printing is turned on by default, with a frequency of one. In an extract, the default is to not print bar numbers. To turn off bar numbering say `\nobarnumbers`. To reinstate periodic bar numbering, or to initiate it in an extract, say `\barnumbers`. To change to a different frequency n , say `\def\freqbarno{ n }`.

The appearance and positioning of the bar number is controlled by the token `\writethebarno`, which by default is defined as

`\def\writethebarno{\fontbarno\the\barno\kernm\qn@width}` where the font is defined as `\def\fontbarno{\it}`. You can change either of these as desired, for example



which was coded as

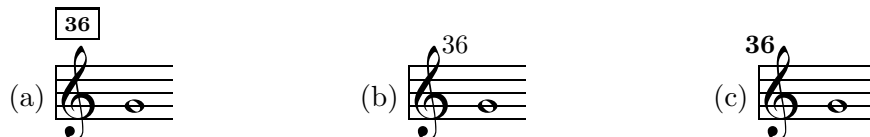
```
\barnumbers
\Notes\Dqbu gh\Dqbl jh\en
\notes\Dqbbu fg\Dqbb l hk\en\bar
\Notes\Tqbu ghi\Tqbl mmj\en
\def\fontbarno{\bf}%
\notes\Tqbbu fgj\Tqbb l njh\en\bar
\Notes\Qqbu ghjh\Qqbl jifh\en\bar
\notes\Qqbbu fgge\Qqbb l jhgi\en
```

2.12.2 System bar numbering

To have a bar number printed just above the beginning of each system, use `\systemnumbers`. The distance above the staff is controlled by `\raisebarno`, which by default is `4\internote` (to fit above a treble clef). This can be redefined with the command `\def\raisebarno{any TEX dimension}`. The horizontal position similarly is defined by `\shiftbarno` which by default is `0pt`.

The number normally is enclosed in a box. If you don't like that, you may redefine the macro `\writebarno` which by default is defined as `\def\writebarno{\boxit{\eightbf\the\barno\barnoadd}}`. This uses the utility MusiX_TE_X macro `\boxit` which will enclose any text string in a box.

Here are some possible alternate formats for system bar numbers:



This was coded as

- (a) (default)
- (b) `\def\writebarno{\tenrm\the\barno\barnoadd}%`
`\def\raisebarno{2\internote}%`
`\def\shiftbarno{2.5\Interligne}%`
- (c) `\def\writebarno{\llap{\tenbf\the\barno\barnoadd}}%`
`\def\raisebarno{2\internote}%`
`\def\shiftbarno{1.3\Interligne}%`

If the previous line does not stop with a bar rule, then the next printed system bar number will immediately be followed by the contents of the token `\writezbarno`, whose default setting is the lower case character 'a'. You might want to change this to '+', in which case you should say `\def\writezbarno{+}`.

2.13 Managing the layout of your score

2.13.1 Line and page breaking

If every bar ends with `\bar` and no other line- or page-breaking commands are used, then the external program `musixflx` will decide where to insert line and page breaks. Line breaks will only come at bar lines. The total number of lines will depend on the initial value of `\elemskip`, which by default is 6 pt in `\normalmusicsize`.

You can force a line or page break by replacing `\bar` with `\alaligne` or `\alapage` respectively. On the other hand, to forbid line-breaking at a particular bar, replace `\bar` with `\xbar`. To force a line or page break where there is no bar line, use `\zalaligne` or `\zalapage`. To mark any mid-bar location as an optional line-breaking point, use `\zbar`.

The heavy final double bar of a piece is one of the consequences of `\Endpiece` or `\Stoppiece`. If you just want to terminate the text with a simple bar, say `\stoppiece` or `\endpiece`. To terminate it with no bar line at all, code `\zstoppiece`.

Once you have stopped the score by any of these means, you may want to restart it, and there are several ways to do so. If you don't need to change the key signature, meter, or clef, you can use `\contpiece` for no indentation, or `\Contpiece` to indent by `\parindent`. If you do want to change some score attribute that takes up space, for example with `\generalsignature` after `\stoppiece`, then to restart you must use `\startpiece`. However, if you don't want `\barno` reset to 1, you must save its new starting value to `\startbarno`. You may also wish to modify instrument names or `\parindent` before restarting.

Recall that thin-thin or thin-thick double bars or blank bar lines can be inserted without stopping by using the commands described in section 2.11.1. Those commands can also be used before `\stoppiece`, `\alaligne`, or `\alapage`

2.13.2 Page layout

Blank space above and below each staff is controlled by the dimension registers `\stafftopmarg` and `\staffbotmarg`. For more info see section 2.20.

The macro `\raggedbottom` will remove all vertical glue and compact everything toward the top of page. In contrast, the macro `\normalbottom` will restore default behavior, in which vertical space between the systems is distributed so that the first staff on the page is all the way at the top and the last staff all the way at the bottom.

The macro `\musicparskip` will allow the existing space between systems to increase by up to `5\Interligne`.

The following values of page layout parameters will allow the maximum material to fit on each page, provided the printer allows the margins that are implied.

A4	letter	A4 or letter
<code>\parindent= 0pt</code>		
<code>\hoffset= -15.4mm</code>		
<code>\voffset= -10mm</code>		
<code>\hsize= 190mm</code>	<code>\hsize= 196mm</code>	<code>\hsize= 190mm</code>
<code>\vsize= 260mm</code>	<code>\vsize= 247mm</code>	<code>\vsize= 247mm</code>

2.13.3 Page numbering, headers and footers

There are no special page numbering facilities in MusiX_TEX; you must rely on macros from plain T_EX. There is a count register in T_EX called `\pageno`. It is always initialized to 1 and

incremented by one at every page break. By saying `\pageno= n`, it can be reset to any value at any time, and will continue to be incremented from the new value.

By default, T_EX will place a page number on every page, centered at the bottom. Unfortunately, this is not the preferred location according to any standard practice. To suppress this default behavior, say `\nopagenumbers`.

One way to initiate page numbering in a more acceptable location is to take advantage of the facts that (a) T_EX prints the contents of the control sequences `\headline` and `\footline` at the top and bottom respectively of every page, and (b) the value of `\pageno` can be printed by saying `\folio`. Therefore, for example, the following sequence of commands, issued anywhere before the end of the first page, will cause page numbers and any desired text to be printed at the top of every page, alternating between placement of the number at the left and right margins, and alternating between the two different text strings (the capitalized text in the example):

```
\nopagenumbers
\headline={\ifodd\pageno\rightheadline\else\leftheadline\fi}%
\def\rightheadline{\tenrm\hfil RIGHT RUNNING HEAD\hfil\folio}%
\def\leftheadline{\tenrm\folio\hfil LEFT RUNNING HEAD\hfil}%
\voffset=2\baselineskip
```

2.13.4 Controlling the total number of systems and pages

Once all the notes are entered into a MusiX_TE_X score, it would be convenient if there were a simple sequence of commands to cause a specified number of systems to be uniformly distributed over a specified number of pages. Unfortunately that's not directly possible¹⁷. Rather, some trial and error will usually be required to achieve the desired final layout. To this end, some combination of the following strategies may be used:

1. Assign a value to the count register `\linegoal` representing the total number of systems. The count register `\mulooseness` must be 0 if using `\linegoal`.
2. Explicitly force line and page breaking as desired, using the macros `\alaligne`, `\alapage`, `\zalaligne` or `\zalapage`.
3. Adjust both `\mulooseness` and the initial value of `\elemskip`: increasing `\mulooseness` from its default of 0 will increase the total number of systems, while changing the initial value of `\elemskip` (use `\showthe\elemskip` to find its default value) may change the total number of systems accordingly.

2.14 Changing clefs, key signatures, and meters

2.14.1 Introduction

To define clefs, key signatures, or meters at the start of a piece, or to change one or more of these attributes¹⁸ anywhere else in a score, MusiX_TE_X requires two steps. The first step is to set the new values of the attributes. Most of the commands for this have the form `\set... .` They will be described in the following subsections.

But this alone will not cause anything to be changed or printed. The second step is to activate the change. This is done by issuing one of the following commands (outside `\notes...\en`): `\startpiece`, `\startextract`, `\contpiece`, `\Contpiece`, `\alaligne`, `\alapage`, `\zalaligne`, `\zalapage`, `\changecontext`, `\Changecontext`, `\zchangecontext`,

¹⁷It is possible in PMX.

¹⁸In this section, *attribute* will refer to any clef, key signature, or meter.

`\changesignature`, `\changeClefs`, or `\zchangeClefs`. Most of these perform other functions as well, and some may be used even when no attributes change. Features unrelated to changing attributes are detailed elsewhere. The first 11 will activate all pending new attributes. If more than one type is activated by a single command in this manner, then regardless of the order they were set, they will always appear in the following order: clef, key signature, meter. The last three commands in the above list obviously activate only the specific type of attribute referred to in the name of the command.

The macros `\changecontext`, `\Changecontext`, `\zchangecontext` will respectively insert a single, double, or invisible bar line before printing the attributes.

2.14.2 Key Signatures

We've already seen in section 2.1.3 how to set key signatures for all instruments with `\generalsignature`, or for specific instruments with `\setsign`. As just noted, these commands only prepare for the insertion of the signatures into the score; it is really `\startpiece` that puts them in place at the beginning of the score.

The commands `\generalsignature` and `\setsign` also serve to set new key signature(s) anywhere in score. The change can then be activated with one of the 11 general commands listed above, or with `\changesignature` if in the middle of a bar. While neither `\changesignature` nor `\zchangecontext` prints a bar line, the differences are that the latter increments the bar number counter and inserts a horizontal space of `\afterruleskip` after the new signature(s). All of these options will repost signatures that have not been changed.

Normally, changing a signature from flats to sharps or vice-versa, or reducing the number of sharps or flats, will produce the appropriate set of naturals to indicate what has been suppressed. This standard feature can be temporarily inhibited by the command `\ignorenats` issued right before the change-activating command.

Here is an example showing various possibilities for changing key signatures. Note the comments between the code lines.

```











\instrumentnumber2\setstaves22%
\setclef1{\bass}\generalsignature2%
\startextract
\Notes\qu K&\qu d|\qu e\en
% Signature change in a single instrument with two staves.
% Naturals appear by default, indicating cancelled sharps.
\setsign20\changesignature
\Notes\qu J&\qu d|\qu e\en
% When changing signature in the middle of a bar and no naturals
% are posted, the new signature can be confused with a simple accidental.
\setsign11\ignorenats\changesignature
\Notes\qu M&\qu d|\qu e\en
% New signatures after a double bar line
\generalsignature{-2}\Changecontext%
\Notes\qu K&\qu d|\qu e\en%
\Notes\qu K&\qu d|\qu e\en%
% New signatures after an invisible bar line. Note the
% difference in spacing compared with beat 3 of the prior measure
\generalsignature{1}\zchangecontext%
\Notes\qu K&\qu d|\qu e\en%

```



2.14.3 Clefs

Macros that define clefs have already been discussed in section 2.1.3. By way of review, here are all of the possible clefs (applied to the lowest staff):

<code>\setclef1\treble</code> <code>\setclef10</code> 	<code>\setclef11</code> 	<code>\setclef12</code> 	<code>\setclef1\alto</code> <code>\setclef13</code> 	<code>\setclef14</code> 
<code>\setclef15</code> 	<code>\setclef1\bass</code> <code>\setclef16</code> 	<code>\setclef17</code> 	<code>\setclefsymbol1\empty</code> ¹⁹ <code>\setclef18</code> 	<code>\setclef19</code> 

Just as with key signatures, these commands only *prepare* for clef changes. To *activate* them, any of the first 11 commands listed in section 2.14.1 could be used. However, one should keep in mind that according to modern conventions, a clef change at a bar line is posted before the bar line, while for example `\changecontext` would post it after the bar line. In part for this reason, we have the special command `\changeClefs`. It can be used anywhere outside `\notes... \enotes` to activate a clef change and insert an amount of horizontal space to accommodate the new clef symbol(s), without printing a bar line. Sometimes no added space is required, in which case `\zchangeClefs` should be used.

Here are some examples of clef changes:

```

\instrumentnumber2\setstaves2%
\setclef1{\bass}\generalsignature2%
\startextract
% Change in one staff only, with added space
\setclef1\treble\changeClefs%
\Notes\qu k&\qu e|\cu{.d}\ccu{e}\en%
% Combined with signature change, also no extra space needed
% twice the same clef in staff 2 - with the help of a blank clef
\setclef28\zchangeClefs\setclef2\treble%
\setclef1\bass\zchangeClefs\setsign1{-2}\setsign2{-2}%
\ignorenats\changesignature%
\Notes\qu K&\cu{de}|\qu e\en%

```

¹⁹Details of the macro `\setclefsymbol` will be discussed later

```

% clef change before barline
\setclef1\treble\zchangeClefs\bar%
\Notes\qu k&\cu{de}|\qu e\en%
% clef change after barline
\setclef1\bass\bar\changeClefs%
\Notes\qu K&\cu{de}|\qu e\en%
% clef change after barline with changecontext
\setclef1\treble\changecontext%
\Notes\cu k&\cu d|\qu e\en%
% twice the same clef
\setclef18\zchangeClefs\setclef1\treble\changeClefs%
\Notes\cu k&\cu e |\en%
\endextract

```



Of course the examples in the last two bars are contrary to accepted practice.

Clef changes initiated with the `\setclef` command have several features in common. When activated after the beginning of the piece, the printed symbol is smaller than the normal one used at the beginning of the piece. Also, MusiX_TE_X automatically adjusts vertical positions of noteheads consistent with the new clef.

There is an additional group of macros for setting new clefs which does not share either of these features. In other words, they will always print full sized symbols, and they won't change the vertical positions of noteheads from what they would have been before the new symbol was printed. We could call this process “clef symbol substitution”, because all it does is print a different symbol (or no symbol at all) in place of the underlying clef which was set in the normal way.

You'll need to use clef symbol substitution if you want to have a so-called octave treble clef or octave bass clef, i.e., one containing a numeral 8 above or below the normal symbol. The syntax for setting upper octaviation is `\setbassclefsymbol{n}\bassoct` or `\settrebleclefsymbol{n}\trebleoct`; for lower octaviation it is `\setbassclefsymbol{n}\basslowoct` or `\settrebleclefsymbol{n}\treblelowoct`. Because these sequences act to *replace* normal bass or treble clefs with a different symbol, they require that the normal clefs be set first. For example

```

\parindent 19mm
\instrumentnumber{4}
\generalmeter{\empty}
\setclef1\bass \setclef2\bass
\setclef3\treble \setclef4\treble
\setbassclefsymbol1\basslowoct
\setbassclefsymbol2\bassoct
\settrebleclefsymbol3\treblelowoct
\settrebleclefsymbol4\trebleoct
\startextract
\Notes\qu{'abcdefghi}&\qu{'abcdefghi}%
&\qu{abcdefghi}&\qu{abcdefghi}&\enotes
\endextract

```

Another application of clef symbol substitution is to cause no clef to be printed, as for example might be desired in percussion music. This can be accomplished with `\setclefsymbol{n}\empty`, which once activated would replace *all* clef symbols in the first (lowest) staff of instrument *n* with blanks.

Normal symbols for those clefs that have been substituted can be restored by `\resetclefsymbols`.

The various clef symbol substitution commands can only be used to substitute for `treble`, `alto`, or `bass` clefs.

In the following example, (1) is two normal clef changes. At (2) the clef is first changed back to treble and then the `\treblelowoct` symbol is substituted by using `\settrebleclefsymbol`. When changing the clef away from treble and then back as at (3), the substitution symbol is still in force. At (4), `\resetclefsymbols` cancels the symbol substitution. If using `\setclefsymbol` all available clefs are changed to the same symbol, as you can see in the two clefs after (5) in comparison with (2). Obviously the second clef after (5) is nonsense; `\resetclefsymbols` puts matters in order at (6) and (7).

This is the code:

```

\begin{music}
\instrumentnumber1\setclef1\bass
\startpiece
\notes\zchar{-5}{1}\qu H\en\setclef1\treble\changeclefs
\notes\qu i\en\setclef1\bass\changeclefs\bar
\notes\zchar{-5}{2}\qu J\en
\setclef1\treble\settrebleclefsymbol1\treblelowoct\changeclefs
\notes\qu i\en\setclef1\bass\changeclefs\bar
\notes\zchar{-5}{3}\qu I\en\setclef1\treble\changeclefs
\notes\qu i\en\setclef1\bass\changeclefs\bar
\notes\zchar{-5}{4}\qu J\en\resetclefsymbols\setclef1\treble\changeclefs
\notes\qu i\en\setclef1\bass\changeclefs\doublebar
\notes\zchar{-5}{5}\qu J\en
\setclef1\treble\setclefsymbol1\treblelowoct\changeclefs
\notes\qu i\en\setclef1\bass\changeclefs\bar
\notes\zchar{-5}{6}\qu I\en\setclef1\treble\changeclefs

```

```

\notes\qu i\en\resetclefsymbols\setclef1\bass\changeClefs\bar
\notes\zchar{-5}{7}\qu J\en\setclef1\treble\changeClefs
\notes\qu i\en
\endpiece

```

2.14.4 Meter changes

As mentioned in section 2.1.3, a common *meter* for all staves can be specified by `\generalmeter{m}`, where *m* denotes the meter. On the other hand, meter changes in specific staves are implemented with `\setmeter{n}{m1}{m2}{m3}{m4}`, where *n* is the number of the instrument, *m1* specifies the meter of the first (lowest) staff, *m2* the second staff, and so forth. (Only enter as many *m*'s as necessary.)

Since meter changes are meaningful only across bars, there is no special command to activate a new meter; rather, they are activated with the general commands `\changecontext`, etc., listed in section 2.14.1.

The next example shows a few methods to get a meter change, in all staves or in a single staff.

```

\instrumentnumber2\setstaves2%
\generalmeter{\meterfrac{4}{4}\meterfrac{2}{4}\meterfrac{1}{4}}%
\setclef1{\bass}\generalsignature2%
\startextract
\setmeter1{\meterfrac{2}{4}}%
\setmeter2{\lower2pt\hbox{\meterfrac{\Bigtype 2}{}}}%
{\meterfrac{3}{4}}\changecontext
\Notes\qu K&\cu{de}|\qu e\en
% bar 11
% Meters, clefs, and key signatures.
% All 3 clefs after bar (probably bad form) if no changeclefs
\setmeter1{\meterfrac{2}{8}}%
\setmeter2{\meterfrac{3}{6}}{\meterfrac{3}{8}}%
\setsign2{-1}%
% How to force showing the bass clef?
\setclef1\bass\setclef2{23}%
\Changecontext
\Notes\qu K&\cu{de}|\qu e\en
% bar 12
% Meters, clefs, and key signatures, with clef before the bar.
% Maybe not best form if signatures are involved
\setmeter1{\meterfrac{2}{4}}%
\setmeter2{\meterfrac{3}{8}}{\meterfrac{3}{6}}%
\setsign2{-1}%
\setclef1\treble\zchangeClefs\changecontext
\Notes\qu k&\cu{de}|\qu e\en

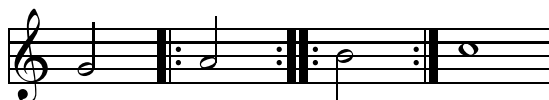
```



2.15 Repeats

To replace a bar line with a left, right, or left-right repeat, use one of the commands `\leftrepeat`, `\rightrepeat` or `\leftrightrepeat` in place of `\bar`. If a `\leftrepeat` happens to come at the end of a system, it will automatically be moved to the start of the next system. If a `\leftrightrepeat` happens to come at the end of a system, MusiX_{TEX} will automatically post a right repeat at the end of the system and a left repeat at the beginning of the next.

For example,



has been coded as:

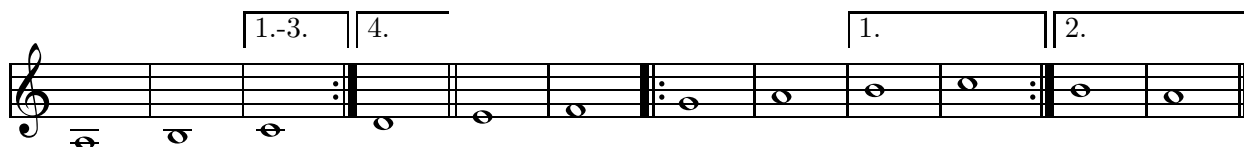
```
\NOTes\ha g\enotes
\leftrepeat
\NOTes\ha h\enotes
\leftrightrepeat
\NOTes\ha i\enotes
\rightrepeat
\NOTes\wh j\enotes
```

To insert a right repeat at a forced line break or at the end of a piece, use `\setrightrepeat` *before* `\alaligne` or `\endpiece`. In contrast, to insert a left repeat at a forced line break or at the beginning of a piece, simply use `\leftrepeat` immediately *after* `\startpiece` or `\alaligne`. To insert a left-right repeat at a forced line break, use `\setrightrepeat\alaligne\leftrepeat`.

In fact it is possible to use `\setleftrepeat`, `\setrightrepeat` or `\setleftrightrepeat` before any `\bar`, `\stoppiece` or `\changecontext`. But be aware that while `\setleftrepeat` behaves properly if the bar is at the end of a system, `\setleftrightrepeat` does not, placing the symbol only at the end of the system.

2.15.1 First and second endings (Voltas)

All volta commands must be entered right before the bar line command (or repeat, etc.) where they are to take effect. There are three commands that suffice to set all voltas. To start one, use `\Setvolta{text}`; to terminate it with or without a vertical line, use `\endvolta` or `\endvoltabox` respectively. The text by default will be followed by a period. There are also various alternate commands (e.g., `\setendvoltabox` is equivalent to `endvoltabox`). Some such alternate forms are used in the following example, but the first three mentioned above are all that are required:



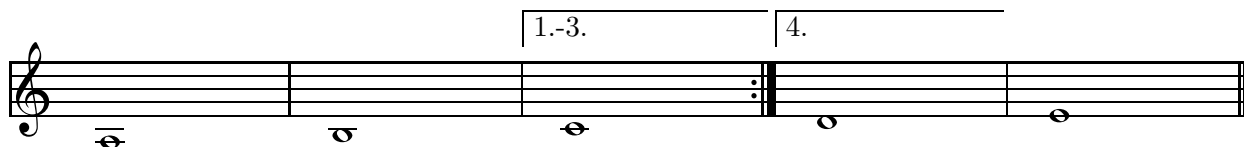
This was coded as

```

\startpiece \addspace\afterruleskip
\notes\wh a\en\bar
\notes\wh b\en\setvoltabox{1.-3}\bar
\notes\wh c\en\setvolta4\setendvolta\rightrepeat
\notes\wh d\en\doublebar
\notes\wh e\en\bar
\notes\wh f\en\leftrepeat
\notes\wh g\en\bar
\notes\wh h\en\setvolta1\bar
\notes\wh i\en\bar
\notes\wh j\en\setvolta2\setendvoltabox\rightrepeat
\notes\wh i\en\bar
\notes\wh h\en\setendvoltabox
\endpiece

```

If the volta only spans one measure and ends without a vertical segment, it can be specified simply by saying `\setvolta{text}` before the bar line command that starts it, and it will automatically terminate at the second bar line command:



which was coded as:

```

\parindent0pt \startpiece \addspace\afterruleskip
\notes\wh a\en\bar
\notes\wh b\en\setvolta{1.-3}\bar \notes\wh c\en\setvolta4\rightrepeat
\notes\wh d\en\bar
\notes\wh e\en\endpiece

```

The height above the top staff line of the horizontal line in a volta symbol is determined by the token `\raisevolta` which is `4\internote` by default. You can change this to any desired dimension.

The period after the text can be removed by saying `\def\voltadot{}` and restored by `\def\voltadot{.}`.

2.15.2 Special symbols for repeating long sections

Four special symbols and corresponding macros are available, namely `\coda p`, `\Coda p`, and `\segno p`, where `p` specifies the pitch; and `\Segno` with no argument. Their behavior is illustrated in this example:



which has been coded:


```

\Notes\segno m\enotes\bar
\Notes\coda m\enotes
\Notes\Segno\enotes\bar
\Notes\Coda m\enotes

```

2.15.3 Repeating a single bar

The special symbol for a single bar repeat is generated by `\duevolte`, as shown in the following example:



whose coding is:

```

\generalmeter\meterC
\setclef1\bass\setstoffs1{2}
\startextract
\Notes\sk\sk\pause|\qa{cegj}\en
\bar\Notes\qa{cdef}|\sk\hsk\duevolte\en
\endextract

```

This is often used with `\centerbar` to more easily center the symbol between bar lines, as shown in the example in section 2.6.3.

2.16 Font selection and text placement

2.16.1 Predefined text fonts

While any \TeX font can be used by MusiX \TeX , there are certain styles and sizes that can be selected using shortcut commands. For ordinary text the shortcuts cover fonts of eight different sizes and three styles. The sizes in points are 8, 9, 10, 12, 14, 17, 20, and 25; the styles are from the standard Computer Modern family: Roman, bold and italic. The four smallest sizes are each available in all three styles, while the larger sizes, which are intended for titles, are available only in bold style. The size selection macros from smallest to biggest are `\smalltype`, `\Smalltype`, `\normtype`, `\medtype`, `\bigtype`, `\Bigtype`, `\BIGtype` and `\BIGtype`. Following size selection, the style may be selected or changed using `\rm` (Roman), `\bf` (bold) or `\it` (italic). If no style is explicitly selected, Roman style will be used for the sizes `\medtype` or smaller. For the larger sizes, only bold style is provided and no style selection is required. Thus, for example, eight point italic is selected with `\smalltype\it`, while twelve point Roman is selected using `\medtype\rm` or simply `\medtype`. To change between styles while maintaining the same size, code `\rm`, `\it` or `\bf` as in Plain \TeX . When MusiX \TeX is started, the default font for ordinary text is ten point Roman, equivalent to `\normtype\rm`.

Another group of fonts, in bold extended italic style, is predefined in point sizes 10, 12, 14, and 17 for dynamic markings. The appropriate font for the current staff size may be selected simply by using `\ppff` as a font specification. Macros `\smalldyn`, `\normdyn`, or `\meddyn` may be used to redefine `\ppff` to represent one of the smallest three.

All predefined fonts are summarized in the following table. The second column gives an explicit control sequence that can alternatively be used locally as a font specification.

Size and style	Font specification	Example
<code>\smalltype</code>	<code>\eightrm</code>	small Roman
<code>\smalltype\bf</code>	<code>\eightbf</code>	small bold
<code>\smalltype\it</code>	<code>\eightit</code>	<i>small italic</i>
<code>\Smalltype</code>	<code>\niner</code>	Small Roman
<code>\Smalltype\bf</code>	<code>\ninebf</code>	Small bold
<code>\Smalltype\it</code>	<code>\nineit</code>	<i>Small italic</i>
<code>\normtype</code>	<code>\tenrm</code>	normal Roman
<code>\normtype\bf</code>	<code>\tenbf</code>	normal bold
<code>\normtype\it</code>	<code>\tenit</code>	<i>normal italic</i>
<code>\medtype</code>	<code>\twelver</code>	medium Roman
<code>\medtype\bf</code>	<code>\twelvebf</code>	medium bold
<code>\medtype\it</code>	<code>\twelveit</code>	<i>medium italic</i>
<code>\bigtype</code>	<code>\bigfont</code>	big bold
<code>\Bigtype</code>	<code>\Bigfont</code>	Big bold
<code>\BIGtype</code>	<code>\BIGfont</code>	BIG bold
<code>\smallldyn</code>	<code>\ppffsixteen</code>	<i>pp ff diminuendo</i>
<code>\normldyn</code>	<code>\ppffttwenty</code>	<i>pp ff crescendo</i>
<code>\medldyn</code>	<code>\ppffttwentyfour</code>	<i>pp ff crescendo</i>
	<code>\ppfftwentynine</code>	<i>pp ff diminuendo</i>

2.16.2 User-defined text fonts

Since MusiX_TEX is a superset of T_EX, you are free to use the standard T_EX machinery for defining and using any special font you desire. You must first of course ensure that (a) all the necessary font files (e.g., `bla10.tfm`, `bla10.pfb`, or equivalents) are installed in the right places in your system, (b) all configuration files (e.g., `config.ps` or equivalent) have been updated, and (c) the T_EX system has been “rehashed”. Then you can use the font just as in any T_EX document, e.g. by coding `\font blafont=bla10` and then `\zchar{10}{\blafont Text in user-defined font}`.

You might also wish to replace once and for all the typefaces invoked by the commands described in the previous section. Again, before doing this, you must follow steps (a-c) of the previous paragraph for all fonts in questions. You can use the standard bitmapped fonts which are converted to postscript by e.g. `dvips`, but you also may replace them by native postscript fonts.

As an example, you can replace the standard `musixtex` fonts by the Times series of fonts as follows:

```
% 8pt roman, bold, and italic
\font\eightrm=ptmr7t at 8pt
\font\eightbf=ptmb7t at 8pt
\font\eightit=ptmri7t at 8pt
% 9pt
\font\niner=ptmr7t at 9pt
\font\ninebf=ptmb7t at 9pt
\font\nineit=ptmri7t at 9pt
% 10pt
\font\tenrm=ptmr7t
\font\tenbf=ptmb7t
\font\tenit=ptmri7t
```

```

% 12pt
\font\twelverm=ptmr7t scaled \magstep 1
\font\twelvebf=ptmb7t scaled \magstep 1
\font\twelveit=ptmri7t scaled \magstep 1
% Large fonts for titles : normal shaped Times-Roman fonts are applied
\font\bigfont=ptmr7t scaled \magstep2 % 14pt
\font\Bigfont=ptmr7t scaled \magstep3 % 17pt
\font\BIGfont=ptmr7t scaled \magstep4 % 20pt
\font\BIGfont=ptmr7t scaled \magstep5 % 25pt
%
\normtype

```

The above definitions are made available in a file `musixtmr.tex`²⁰, to be used as

```

\input musixtex
\input musixps
\input musixtmr
...

```

Here's a comparison of some of the Computer Modern Roman and Times Roman fonts:

normal (10pt) cm Roman	normal Times Roman
normal cm bold	normal Times bold
<i>normal cm italic</i>	<i>normal Times italic</i>
cm big	Times big
cm BIG	Times BIG bold

2.16.3 Text placement

Special macros are provided to allow precise placement of any T_EX text, vertically relative to the staff, and horizontally relative to any note in the staff.

The macros in the first group will vertically position the text with the baseline at any specified pitch or staff line. They must be used inside `\notes... \enotes`. They will not insert any additional horizontal space. They have the forms `\zcharnote{p}{text}`, `\lcharnote{p}{text}`, and `\ccharnote{p}{text}`, where *p* is the pitch. With the first one, text will spill to the right from the current insertion point, with the second it will spill to the left, and with the third it will be centered horizontally.

There are similar macros `\zchar{p}{text}`, `\lchar{p}{text}`, and `\cchar{p}{text}`, which differ from the previous three in that the pitch *must* be given with a number (representing the number of staff positions up from the lowest line), and that the number need not be an integer.

To vertically position any text midway between two consecutive staves, use `\zmidstaff{text}`, `\lmidstaff{text}`, or `\cmidstaff{text}` at the appropriate point in the lower staff.

The macros `\uptext{text}` and `\Uptext{text}` are simply shorthands for `\zchar{10}{text}` and `\zchar{14}{text}` respectively.

The text items handled by all of the above macros can include any appropriate string of T_EX control sequences, including font definitions, `\hbox`'es, etc.

Material posted with any of the macros described in this section will not create any additional horizontal or vertical space within the current system, and will overwrite anything in the current system that gets in the way. It is the typesetter's responsibility to ensure there is adequate white space within the current system to accommodate any text emplaced with these macros. On the

²⁰by Hiroaki MORIMOTO

other hand, if text is emplaced far above or below a system, MusiX_TEX will usually insert additional vertical space if needed.

2.16.4 Rehearsal marks

Rehearsal marks are usually boxed or circled uppercase letters or digits. They can be defined using the macros `\boxit{text}` or `\circleit{text}`. For boxed text, the margin between the text and box is controlled by the dimension register `\boxitsep`, which can be reset to any T_EX dimension if the default value of `3pt` is unsatisfactory. To emplace the mark, use `\Uptext` or any of the other macros defined in the previous section.

2.17 Miscellaneous other notations

2.17.1 Metronomic indications

By way of example, the notations $\text{♩} = 60$ and $\text{♩} = \text{♩}$ are respectively coded as `\metron{\hup}{60}` and `\metronequiv{\qup}{\qu}`, which are normally emplaced using `\Uptext`.

2.17.2 Accents

You may use²¹

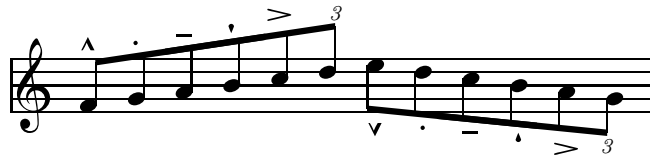
- `\upz{p}` (upper *staccato*) to put a dot above a note head at pitch p ,
- `\lpz{p}` (lower *staccato*) to put a dot below a note head at pitch p ,
- `\usf{p}` (upper *sforzando*) to put a $>$ accent above a note head at pitch p ,
- `\lsf{p}` (lower *sforzando*) to put a $>$ accent below a note head at pitch p ,
- `\ust{p}` (upper *tenuto*) to put a hyphen above a note head at pitch p ,
- `\lst{p}` (lower *tenuto*) to put a hyphen below a note head at pitch p ,
- `\uppz{p}` (upper *staccatissimo*) to put a solid vertical wedge above a note head at pitch p ,
- `\lppz{p}` (lower *staccatissimo*) to put an inverted solid vertical wedge below a note head at pitch p ,
- `\usfz{p}` (upper *forzato*) to put a “dunce cap” above a note head at pitch p ,
- `\lsfz{p}` (lower *forzato*) to put an inverted “dunce cap” below a note head at pitch p ,
- `\upzst{p}` (upper *staccato/tenuto*) to put a dot and a hyphen above a note head at pitch p ,
- `\lpzst{p}` (lower *staccato/tenuto*) to put a dot and a hyphen below a note head at pitch p ,
- `\flageolett{p}` to put a small circle above a note head at pitch p .

These marks are horizontally centered relative to solid note heads. To compensate for the fact that whole notes are wider, you should use `\wholeshift{Any nonspacing macro}` to center accents and other items (e.g. `\Fermataup`) above a whole note.

There are also variants of the most common accents²² which will be automatically positioned above or below a beam. They are spelled like the corresponding normal accent, but preceded with the letter **b**, and their argument, instead of the pitch, is the beam reference number . Thus

²¹Note from the editor: the reason the names of some of these macros don't seem to be constructed to suggest the terms used in the descriptions is that whoever originally defined the macros had in mind terms that did not agree with normal English usage.

²²Thanks to Klaus BECHERT's corrections.



was coded as

```
\startextract
\Notes\ibu0f3\busfz0\qb0f\bupz0\qb0g\bust0\qb0h%
  \buppz0\qb0i\busf0\qb0j\butext0\tqh0k\en
\Notes\Ib10lg5\blsfz0\qb0l\blpz0\qb0k\blst0\qb0j%
  \blppz0\qb0i\blsf0\qb0h\bltext0\tqb0g\en
\endextract
```

The macros `\bltext` and `\butext` are detailed in the next section, where the mystery of why they produce the number 3 is resolved.

2.17.3 Numbers and brackets for xtuplets

The following table lists all the special macros that place a number indicating an xtuplet. Some also place a bracket above or below the notes, and are intended for use with unbeamed notes. In the table, p is a pitch, k is a number, n is a beam number, w is a bracket width in `\internotes`, and s is the bracket slope as a multiple of 1.125 degrees. The macro `\txt` contains a default number and its font, which will be emplaced by the first and third through sixth macros, and is initially defined as `\def\txt{\eighttit 3}`. The macro `\tuplettxt` serves the same role for the last two macros. The first four are to be used with beamed xtuplets. As indicated in the last column, the last four produce a sloping bracket and are to be used with unbeamed xtuplets. The last two require the extension file `tuplet.tex` to be input right after `\input musixtex`.

Macro and arguments	Number printed	Where invoked	Needs <code>tuplet.tex</code> ?	Bracket
<code>\triolet{p}</code>	<code>\txt</code>	before beam	no	none
<code>\xtuplet{k}{p}</code>	k	before beam	no	none
<code>\butext{n}</code>	<code>\txt</code>	before note at number	no	none
<code>\bltext{n}</code>	<code>\txt</code>	before note at number	no	none
<code>\uptrio{p}{w}{s}</code>	<code>\txt</code>	before first note	no	solid
<code>\downtrio{p}{w}{s}</code>	<code>\txt</code>	before first note	no	solid
<code>\uptuplet{p}{w}{s}</code>	<code>\tuplettxt</code>	before first note	yes	with gap
<code>\downtuplet{p}{w}{s}</code>	<code>\tuplettxt</code>	before first note	yes	with gap

Here are some examples of the first six macros in the table:



whose coding is

```
\notesp\xtuplet6n\isluru01\ib1010\qb0{11111}\tslur01\tqb01\en\bar
\notesp\triolet n\isluru01\Ib101n2\qb0{1m}\tslur0n\tqb0n\en
\notesp\ibslurd0k\Ib10km2\qb0k\bltext0\qb01\tdbslur0m\tqb0m\en\bar
\Notesp\triolet o\isluru01\ql{1m}\tslur0n\ql n\en\bar
\Notesp\uptrio o16\ql 1\en\notesp\cl n\en
\Notesp\downtrio 016\qu e\en\notesp\cu g\en
```

Next are examples using the macros from `\tuplet.tex`. Comparison of the coding with the printed result reveals that (a) a redefinition of `\tuplettxt` inside a notes group only applies inside that group and leaves the default definition intact (bar 2), (b) the default definition is `\def\tuplettxt{\ppfffsixteen3}`, a 10-pt bold extended italic numeral 3 which may be larger and heavier than desired (first set, bar 3) (c) to get the number properly centered in the gap, you must include some extra spaces after the number in the definition of `\tuplettxt`, and (d) as you can see in the last bar, if the span becomes too small, the macros still won't provide enough room for the number in the gap; in this case it would be better to use the macros without gaps.



with coding

```
\input tuplet
\parindent0mm\generalmeter{\meterfrac24}
\startextract\addspace\afterruleskip
\notesp\triolet o\Ib10ln2\qb0{lm}\tqb0n\en
\notesp\Ib10km2\qb0k\bltext0\qb0l\tqb0m\en\bar
\notesp\def\tuplettxt{\eightit 5\//}\uptuplet o{4.1}2\c1{jklmn}\en\bar
\Notesp\uptuplet o16\ql l\en\notesp\c1 n\en
\def\tuplettxt{\eightit 3\//}
\Notesp\downtuplet O16\qu e\en\notesp\cu g\en\bar
\notesp\uptuplet o16\ccl l\en\notesp\cccl n\en
\notesp\downtuplet O16\ccu e\en\notesp\cccu g\en
\endextract
```

2.17.4 Ornaments

2.17.4.1 Arpeggios

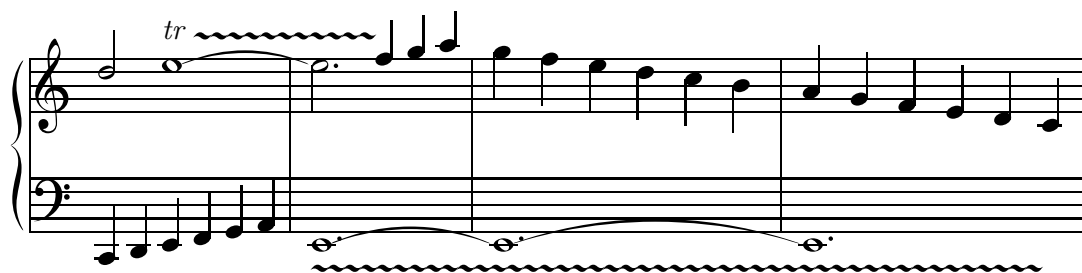
Arpeggios (i.e. $\}$) can be coded with the macro `\arpeggio{p}{m}` where p is the pitch of the base of the arpeggio symbol and m is its height in units of `\interligne`, the distance from one staff line to the next. It should be issued before the affected chord. It is automatically positioned to the left of the chord, but inserts no spacing. Its variant `\larpeggio` sets the arpeggio symbol roughly one note head width to the left of the default position, and is intended to avoid collision with single accidentals on chord notes.

2.17.4.2 Arbitrary length trills

There are two styles of arbitrary length trills, each with two variants. For a trill with preassigned length, use `\trille{p}{l}` for $\sim\sim\sim$ or `\Trille{p}{l}` for $tr \sim\sim\sim$, where p is the pitch and l the length in current `\noteskips`.

To let MusiX_{TEX} compute the length of the trill, or if it extends across a system break, you can use `\Itrillen{p}` to start a plain trill, where n is a trill reference number between 0 and 6; then `\Trillen` to terminate it. To get the tr at the beginning, use `\ITrillen{p}` to start the trill and `\Trillen` to close it.

For example:

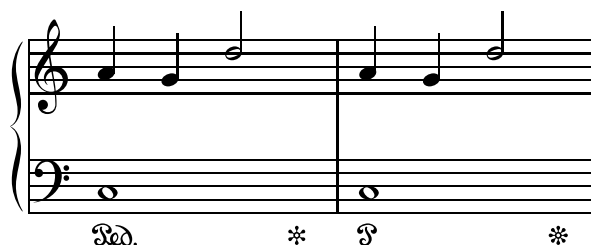


whose coding is

```
\begin{music}
\instrumentnumber{1}
\setstaves12
\setclef1{6000}
%
\startextract
\notes\qu{CDEFGH}|\hu k\sk\ITrille 1p\itenu1\wh l\enotes
\bar
\notes\ITrille 2A\itenu1\whp E|\tten1\hlp l\sk\Ttrille 1\qu {mno}\enotes
\bar
\Notes\tten1\itenu1E\whp E|\ql{nmlkji}\Toctfin1\enotes
\bar
\Notes\tten1\whp E\sk\sk\sk\sk\Ttrille2|\qu{hgfedc}\enotes
\endextract
```

2.17.4.3 Piano pedal commands

The macro `\PED` inserts a piano pedal command below the staff; `\DEP`, a pedal release. Alternate symbols, the first of which occupies less space, are invoked with `\sPED` and `\sDEP`. For example,













was coded as

```
\Notes\PED\wh J|\qu h\enotes
\Notes|\qu g\enotes
\Notes|\hu k\enotes
\Notes\DEP\enotes \bar
\Notes\sPED\wh J|\qu h\enotes
\Notes|\qu g\enotes
\Notes|\hu k\enotes
\Notes\sDEP\enotes
```





The vertical position of `\PED`, `\sPED`, `\DEP` and `\sDEP` can be globally changed by redefining its elevation, which has the default definition `\def\raiseped{-5}`. To locally change the vertical position of a pedal symbol, use one of the more fundamental macros `\Ped`, `\sPed`, `\Dep` and `\sDep` in combination with `\zchar` or `\zcharnote`. Since the `\PED` symbol is rather wide, it might collide with adjacent bass notes. To shift it horizontally, you could use `\loff{\PED}`.


2.17.4.4 Other ornaments

The argument p in the following macros for ordinary ornaments is the pitch at which the ornament itself appears. They are all nonspacing macros. You may use

- `\mordent{p}` for ,
- `\Mordent{p}` for ,
- `\shake{p}` for ,
- `\Shake{p}` for ,
- `\Shake1{p}` for ,
- `\Shakesw{p}` for ,
- `\Shakene{p}` for ,
- `\Shakenw{p}` for ,
- `\turn{p}` for ,
- `\backturn{p}` for .

In the following macros for fermatas, the argument p is the pitch of the notehead on which the fermata rests, assuming no additional vertical adjustments are needed for stems or intervening staff lines. They are all nonspacing macros. You may use

- `\fermataup{p}` for ,
- `\fermatadown{p}` for ,
- `\Fermataup{p}` for , centered over a whole note,
- `\Fermatadown{p}` for , centered under a whole note.

A *breath* mark  can be put just above the staff with `\zbreath`. This is a nonspacing macro. On the other hand, `\cbreath` will cause a space of one `\noteskip` and place the comma midway through the space.

The `\caesura` command inserts a slash 0.5\noteskip before the place it is entered, while adding no space:



2.17.5 Alphabetic dynamic marks

Conventional dynamic symbols *pppp*, *ppp*, *pp*, *p*, *mp*, *mf*, *f*, *fp*, *sf*, *ff*, *fff* and *ffff* can be posted using the macros `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\fp`, `\sF`, `\ff`, `\fff`, `\ffff` respectively as the second argument of `\zcharnote` or `\ccharnote`.

2.17.6 Hairpins (crescendos and decrescendos)

The syntax and properties of hairpins will differ depending on whether or not you have input `musixps.tex`, which activates not only type K postscript slurs and ties but hairpins as well. The font-based variety are always horizontal, are limited in length, and cannot span system breaks. Using the postscript variety removes all these restrictions, but they will not be visible in DVI previewers.

2.17.6.1 Font-based hairpins

There are two categories of font-based hairpins. The first type requires only one command, `\crescendo{l}` or `\decrescendo{l}`, where l is any T_EX dimension, either a fixed one—for example in points—or a scalable one expressed either explicitly or implicitly as some number of `\noteskips`. These should be used as arguments to `\zcharnote`, `\zchar`, `\uptext`, `\zmidstaff`, etc., to post them at the desired altitude. The longest such symbol is $\simeq 68$ mm.

The second type of font-based hairpin requires two commands, one to start it and another to end it. The starting macro is `\icresc`. It has no arguments. Only one invocation suffices to start any number and combination of crescendos and diminuendos. The ending macros are `\tcresc` or `\tdecresc`. They should be used as arguments of `\zcharnote` etc, which will set the altitude. For example,



which was coded as

```
\Notes\cmidstaff\ppp|\ca c\en
\Notes\icresc|\ca{defgh'abcde}\en
\Notes\zmidstaff{\loff\tcresc}\cmidstaff\fff|\ca{'f}\en
```

while



was coded as

```
\Notes\cmidstaff\ppp|\ca c\en
\Notes\icresc|\ca{defgh'abcde}\en
\Notes\zcharnote N{\tcresc}\cmidstaff\fff|\zcharnote q{\tcresc}\ca{'f}\en
```

2.17.6.2 Postscript hairpins

As already noted, these require that you have input `musixps.tex`. Once having done that, you should not try to use font-based hairpins because the syntax is incompatible.

Again, there are two different types. The first type is normally initiated with either `\icresc{n}` or `\idecresc{n}`, and terminated with `\tcresc{n}`, where n is a hairpin index, which is virtually unlimited but certain restrictions apply if it exceeds 14. The altitude is set by the value of `\setcrescheight`, which by default is -5 and which must be expressed numerically. Note that `\tcresc` is the same as `\tdecresc`.

You can shift the starting or ending point horizontally by replacing the foregoing macros with `\ilcresc`, `\ildecresc`, `\ircresc`, `\irdecresc`, `\tlcresc`, `\tldecresc`, `\trcresc`, `\trdecresc`, for example to make space for an alphabetic dynamic mark.

The second form of postscript hairpin macros allows individual and arbitrary specification of the altitude and horizontal offset. The syntax is `\Icresc{n}{h}{s}`, where h is the altitude—which must be numerical—and s is the horizontal offset in `\internote`. Similar syntax obtains for `\Idecresc` and `\Tcresc`.

These hairpins may span several lines. If one of them spans three systems then the height of the middle section can be adjusted with `\liftcresc{n}{h}`. The height of the first and last parts of a broken crescendo will be defined by the height parameter in `\Icresc` or `\Tcresc`.

There are numerous other nuances and shorthand macros that are described in the comments in `musixps.tex`.

As an example of a postscript hairpin,



was coded as

```
\input musixps
\generalmeter{\meterfrac{12}{8}}
\startextract
\Notes\ccharnote{-8}{\ppp}\Icresc0{-7}{6}\ca{bdegh'bde}\en
\Notes\Tcresc0{-4}{-2}\zcharnote{-5}{\fff}\ca{'f}\en
\endextract
```

2.17.7 Length of note stems

The default length of note stems is the distance of one octave, i.e. `7\internote` or `4.66\interbeam`. The default may be changed with the macro `\stemlength{b}` where b is the length in `\interbeams`.

The command `\stemcut` causes stems that extend outside the staff to be shortened depending on the pitch of the notes. It is the default behavior. To suppress this adjustment, issue the command `\nostemcut`.

Normally, down stems never end above the middle line of the staff and up stems never below that line. The command `\stdstemfalse` will inhibit this adjustment, but only for the next stem generated. There is no command to globally suppress this behavior.

2.17.8 Brackets, parentheses, and oblique lines

Several varieties of brackets, parentheses and oblique lines are provided for use within a score.

- `\lpar{p}` and `\rpar{p}` yield left and right parentheses at pitch p . They could be used to enclose notes or to build *cautionary* accidentals, although the latter are more easily obtained with predefined macros (see 2.8).

For example,

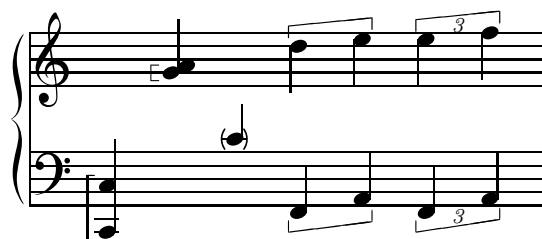
is coded as

```
\notes\lpar{g}\rpar{g}\hu{g}\sk%
\loffset{1.5}{\lpar{g}}\loffset{1.5}{\rpar{g}}%
\loffset{.4}{\sh g}\hu{g}\en
```

- `\bracket{p}{n}` posts a square bracket to the left of a chord, vertically spanning n internotes.
- `\doublethumb{p}` indicates a bracket as above spanning $2\internotes$.

- `\ovbkt{p}{n}{s}` and `\unbkt{p}{n}{s}` draw a sloped bracket starting at the current position at pitch p , with horizontal extent n noteskips and slope s in multiples of $1-1/8$ degree.
- `\uptrio{p}{n}{s}` and `\downtrio{p}{n}{s}` are like `\ovbkt` but with freely definable `\txt` centered inside.
- `\varline{h}{l}{s}` builds an oblique line starting at the current horizontal position. It must be used inside a zero box, such as for example as the second argument of `\zcharnote`. h is the height of the starting point, l is the length, and s is the slope. It is used in the definitions of some of the foregoing macros, and could be used, for example, to construct various obscure baroque ornaments made up of diagonal lines.

For example,



is coded as

```
\begin{music}
\setstaves1{2}
\setclef1{\bass}
\startextract
\Notes\bracket C8\zq C\qu J\en
\Notes|\doublethumb g\rq h\qu g\en
\Notes\lpar c\rpar c\qu c\en
\Notes\unbkt C15\qu {FH}|\ovbkt n14\ql{k1}\en
\Notes\downtrio C16\qu {FH}|\uptrio o14\ql{lm}\en
\endextract
\end{music}
```

2.17.9 Forcing activity at the beginning of systems

A macro named `\everystaff` is executed each time a new system begins. It is normally void, but it can be defined (simply by `\def\everystaff{...}`) to cause MusiX_{TEX} to post anything reasonable at the beginning of each system. For it to affect the first system in a score, it must be defined *before* `\startpiece`.

If a macro named `\atnextline` is defined at any point in a score, it will be executed just once, viz., at the next computed or forced system break. More precisely, it is executed after the break and before the next system begins. Thus it is suitable for redefining layout parameters.

In some scores, tenor parts are not coded using the *bass* clef, but using rather the *octave treble clef*, which is subscripted by a numeral 8. This clef is supported by the clef substitution command `\settrebleclefsymbol{n}\treblelowoct`, as already explained in section 2.14.3. However, if for some reason you aren't happy with the height of the "8", it can be posted on selected staves at the beginning of every system using `\everystaff` and `\zcharnote` as follows:



The coding is

```
\instrumentnumber{4}
\setclef1\bass
\def\everystaff{%
  \znotes&\zchar{-6}{\eightrm \kern -2\Interligne 8}%
  &\zchar{-6}{\eightrm \kern -2\Interligne 8}\en}%
\startextract
\NOTes\ha{HIJK}&\ha{efgh}&\ha{hijk}&\ha{hmlk}\en
\endextract
```

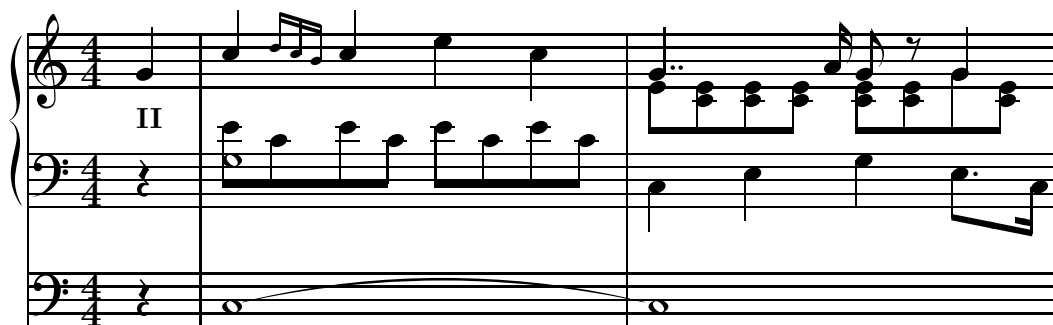
2.18 Smaller notes in normal-sized staves

Here we describe how to reduce the size of note symbols without changing the size of the staff itself. Changing overall staff size will be treated in section 2.19.

2.18.1 Arbitrary sequences of notes

Written-out ornaments and *cadenzas* are usually typeset with smaller notes and spacing than normal. The smaller *size* can be initiated anywhere inside a `\notes` group by stating `\smallnotesize` or `\tinynotesize`. Normal note size is then restored by `\normalnotesize` or simply by terminating the `notes` group and starting another. Smaller *spacing* must also be explicitly indicated, usually by redefining `\noteskip` in some way.

As an example, this excerpt, from the beginning of the Aria of the “Creation” by Joseph HAYDN),



can be coded as

```

\instrumentnumber{2}
\generalmeter{\meterfrac44}
\setstaves2{2}
\setclef2{\bass}
\setclef1{\bass}
\startbarno=0
\startextract
\NOTes\qp&\zmidstaff{\bf II}\qp|\qu g\en
% measure 1
\bar
\Notes\itieu2J\wh J&\zw N\ibl0c0\qb0e|\qu j\en
\notes&\ibl0c0\qb0c|\multnoteskip\tinyvalue\tinynotesize
  \Ibbu1ki2\qb1{kj}\tqh1i\en
\Notes&\qb0e\tbl0\qb0c|\qu j\en
\Notes&\ibl0c0\qb0{ece}\tbl0\qb0c|\ql l\sk\ql j\en
% measure 2
\bar\Notes\ttie2\wh J&\ql J\sk\ql L|\zqupp g\qb1e0%
  \zq c\qb1e\zq c\qb1e\zq c\tbl1\zqb1e\en
\notes&|\sk\ccu h\en
\Notes&\ql N\sk\ibl0L{-4}\qbp0L|\ibl1e0\zq c\zqb1e\cu g%
  \zq c\zqb1e\raise\Interligne\ds\zqu g\qb1g\en
\notes&\sk\tbb10\tbl0\qb0J|\tbl1\zq c\qb1e\en
\endextract

```

2.18.2 Grace notes

Grace notes are a special case of small and tiny notes, namely single-stemmed eighth notes with a diagonal slash through the flag. To enable this, there are the macros `\grcu{p}` and `\grcl{p}`, which by themselves would produce normal-sized eighth notes with a slash. They should be used along with the note size reduction macros and spacing reduction macros just discussed. In addition, chordal grace notes can be built as in the following example:



which was coded as

```

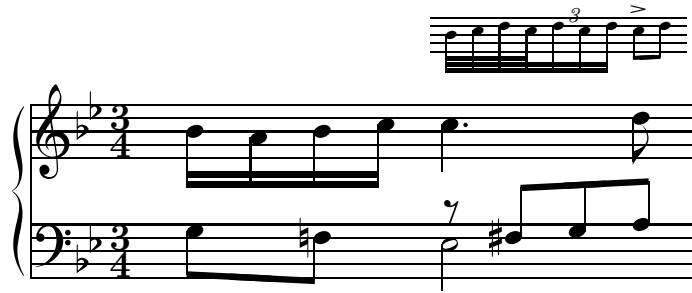
\startextract
\NOTes\hu h\enotes
\notes\multnoteskip\smallvalue\smallnotesize\grcu j\enotes
\NOTes\hu i\enotes
\bar
\notes\multnoteskip\tinyvalue\tinynotesize\zq h\grcl j\enotes
\NOTes\wh i\enotes
\endextract

```

2.18.3 Ossia²³

This clever example had been provided by Olivier Vogel:

²³(Italian O sia) Or else



The code is:

```

\hsize70mm%
\def\xnum#1#2#3{\off{#1\elemskip}\zcharnote{#2}{\smalltype\it #3}%
\off{-#1\elemskip}}%
\newbox\ornamentbox
\setbox\ornamentbox=\hbox to 0pt{\kern-4pt\vbox{\hsize=2.6cm%
\nostartrule\smallmusicsize\setsize1{\smallvalue}\setclefsymbol1\empty%
\startpiece\addspace{2pt}%
\notes\ibbb12{'c}0\qb2b\qb2c\qb2d\tbbb12\qb2c\en%
\notes\xnum{1.15}{'e}3\qb2d\qb2c\tb12\qb2d\en%
\notes\ibl12{'c}1\usf e\qb2c\en%
\notes\tb12\qb2{'d}\en%
\zstoppiece%
}\hss}
\normalmusicsize\nopagenumbers
\def\nbstruments{1}%
\setstoffs12\setclef1{60}%
\generalsignature{-2}\generalmeter{\meterfrac{3}{4}}%
\parindent 0pt%
\stafftopmarg0pt\staffbotmarg5\Interligne\interstaff{10}\relax
\startpiece\addspace\afterruleskip%
\notes\ibl1{'G}{-1}\qb1G\sk\bigna F\tb11\qb1F|%
\ibb12{'b}0\qb2b\qb2a\qb2b\tb12\qb2c\en%
\notes\hl{'E}\bsk\raise6\internote\ds\ibu3{G}1\bigsh F%
\qb3F\qb3G\tbu3\qb3{'A}|\zcharnote{10}{\copy\ornamentbox}\qlp{'c}\sk\sk%
\cl d\en%

```

2.19 Staff size

In contrast with the prior section, here we describe how to change the sizes of everything...staff, notes, and all other symbols. In section 2.1.3 we saw how to set the size for all instruments at the start of a score. Any one of the same macros—`\normalmusicsize`, `\smallmusicsize`, `\largemusicsize`, or `\Largemusicsize`—can be used to change the size of all instruments midway through a score, but in this case it must come between `\stoppiece` and `\startpiece`.

Once the overall staff size is set, you can alter the size of any desired instrument with the macro `\setsize{n}{s}`, where n is the instrument number and s is a factor by which the size is to be changed from the prevailing overall size. There are five predefined macros that should be used for the size factor s . Their names and respective values are `\normalvalue` (1.0), `\smallvalue` (0.80), `\tinyvalue` (0.64), `\largevalue` (1.2), and `\Largevalue` (1.44). MusiX_TE_X should not crash if you use an explicit number different from any of these, but the result may be less than satisfactory.

Once again, if used at the beginning of a piece, the `\setsize` macro must precede `\startpiece` (not `\contpiece`), and if used after the beginning, must be preceded by `\stoppiece`.

As an example, we give two bars of the *Ave Maria* by Charles GOUNOD, based on the first prelude of J. S. Bach's *Well Tempered Clavier*, as transcribed for organ, violin and voice by Markus VEITTES:

This example was coded as:

```

\def\oct{\advance\transpose by 7}
\def\liftqs#1{\raise#1\Interligne\qs}
\parindent0pt
\sepbarrules
\instrumentnumber{3}
\generalmeter{\meterC}
\setinterinstrument2{3\Interligne}
\setsize3\tinyvalue
\setsize2\tinyvalue
\setclef1\bass
\setstaves1{2}
\startpiece\addspace\afterruleskip
%Takt 9
\notes\zhl c\liftqs6\qupp e|\ds&\oct
  \itieu5h\hl h&\tx ~~~gra---*\itied4h\hu h\enotes
\notes|\ibbl0j3\qb0h\tqb0l\enotes
\notes|\ibbl1k0\qb1{ohl}\tqb1o\enotes
\notes\zhl c\liftqs6\qupp e|\ds&\oct
  \ttie5\ibl4c0\qb4h&\ttie4\ibu5g{-3}\qb5h\enotes
\notes|\ibbl0j3\qb0h\tqb0l&\oct\qb4a&\tx ---*\tqh5a\enotes
\notes|\ibbl1k0\qb1o\qb1h&\oct\qb4b&\tx ~ti~-*\cu b\enotes
\notes|\qb1l\tqb1o&\oct\tqb4c&\tx a*\cu c\enotes
\bar
%Takt 10
\notes\zhl c\liftqs6\qupp d|\ds&\oct
  \qlp d&\tx ~~~ple---*\ibsluru4e\qup d\enotes
\notes|\ibbu1g3\bigaccid\qb1{^f}\tqh1h\enotes
\notes|\ibbu2i0\qb2k\qb2f\enotes
\notes|\qb2h\tqh2k&\oct\cl e&\curve222\tubslur4f\cu e\enotes
\notes\zhl c\liftqs6\qupp d|\ds&\oct\ql d&\tx na,*\qu d\enotes
\notes|\ibbu1g3\qb1f\tqh1h\enotes
\notes|\ibbu2i0\qb2{kfh}\tqh2k&\qp&\qp\enotes

```

`\endpiece`

2.20 Layout parameters

Most layout parameters are set by MusiX_TE_X to reasonable default values. However, some projects will require altering one or more of them. In this section we discuss the most important parameters and how to change them.

2.20.1 List of layout parameters

In the following, the indication “(*NOT to be changed*)” does not mean that this parameter cannot be changed at all, but that it should not be modified directly, e.g. by saying something like `\Interligne=14pt`. In other words, changes in these parameters must be accomplished only by more comprehensive macros which not only revise them but at the same time perform other necessary related changes. Even though you cannot *change* these, you may *refer* to them in your coding if that proves useful.

`\Interligne` : vertical distance between the bottoms of consecutive staff lines of the current instrument, taking no account of a possible alteration by `\setsize` (*NOT to be changed*).

`\internote` : vertical spacing between notes one scale step apart in the current instrument, taking account of a possible alteration by `\setsize` (*NOT to be changed*).

`\Internote` : vertical spacing between notes one scale step apart in any instrument whose `\setsize` has the default value `\normalvalue` (1.0), equal to 0.5\Interligne (*NOT to be changed*)

`\staffbotmarg` : margin below the first (lowest) staff of the first (lowest) instrument. Changes are recognized at the next system. Default is 3\Interligne .

`\stafftopmarg` : margin above the last (uppermost) staff of last (uppermost) instrument. Changes are recognized at the next system. Default is 3\Interligne .

`\interbeam` : vertical distance between beams. (*NOT to be changed*).

`\interstaff` : a very important macro with a single numerical argument representing the factor that multiplies 2\internote to give the distance between the bottom of one staff and the bottom of the next one. In fact the macro redefines the parameter `\interfacteur`.

`\interportee` : distance between the bottom of one staff and the bottom of the next one. It is always reset to $2\text{\interfacteur}\text{\internote}$ at the next system. Therefore, trying to change `\interportee` will have no effect. Change `\interstaff` instead. Further, note that `\interstaff` applies to all the instruments, but each distinct instrument may have a different `\internote` (see 2.19).

`\interinstrument` : additional vertical distance between two consecutive instruments. This means that the distance between the lowest line of the previous instrument and the lowest line of the top staff of the current instrument is $\text{\interportee} + \text{\interinstrument}$. The default value of `\interinstrument` is zero, but sometimes you may want additional space between distinct instruments. This is a general dimension register. As usual in T_EX, it can be set using a command such as `\interinstrument=10pt` or `\interinstrument=6\internote`. Its value can be overridden for the space above any particular instrument with the macro `\setinterinstrument{n}{s}`, where *n* is the instrument and *s* is the replacement value of the space to be added. The `\setinterinstrument` macro may be useful in some vocal scores to provide vertical space for lyrics. Note that after you have used `\setinterinstrument`, you cannot reset the distances for that instrument with `\interinstrument`; you must subsequently use `\setinterinstrument` for that purpose.

`\systemheight` : distance from the bottom of the lowest staff to the top of the highest one. This is the length of any vertical lines such as repeats that span the full height of a system. (*NOT to be changed*).

In addition, when handling notes of a given staff of a given instrument, the following dimensions are available (note these are not true registers, but *equivalenced symbols* through a `\def`):

`\altplancher` : altitude of the lowest line of the lowest instrument (*NOT to be changed*).

`\altitude` : altitude of the lowest line of the lowest staff of the current instrument (*NOT to be changed*).

`\altportee` : altitude of the lowest line of the current staff (*NOT to be changed*).

`\stemfactor` : parameter defining the length of stems on half, quarter, and beamed eighth notes, in units of `\interbeam`. Normally a stem has the length of one octave, i.e. 3.5\Interligne . However, this is not correct for small or tiny note sizes. Therefore, stem length is defined as a multiple of the dimension `\interbeam`, which is chosen because it is automatically redefined as a different multiple of `\Interligne` whenever note size is changed. For example, with `\normalmusicsize` when `\setsize` is `\normalvalue`, `\interbeam` is 0.75\Interligne . This legislates a default value for `\stemfactor` of 4.66 ($=3.5/0.75$). To change stem length, it is easiest to use e.g. `\stemlength{3.5}`, which simply redefines `\stemfactor`. Subsequently, `\stemfactor` will not automatically be reset to the default, but keep in mind that if it is changed inside a notes group, the change will only be effective within that notes group.

2.20.2 A convenient macro for changing layout parameters in mid-score

Of the parameters just described that can be changed, many should only be changed between the end of one system and the beginning of the next. The command sequence `\def\atnextline{any control sequence}` may be useful for this purpose. It will cause *any control sequence* to be inserted right before the next new line is begun, provided the line break is not initiated by a `\startpiece`. Thus this will work with automatically generated line breaks, with those forced by `\alaligne`, and with those forced by explicit use of `\endpiece` or `\stoppiece` followed by `\contpiece`. (Note, however, that in the latter case it would be just as convenient to enter the parameter changes explicitly as well.) The control sequence will only be executed once, after which `\atnextline` is redefined as `\empty`.

2.20.3 Changing the number of lines per staff

Naturally, the default number of lines per staff is five. But you may want a different number in some or all staves, for example for gregorian music, percussion music, guitar tablature, or early baroque keyboard music. To do so, use the command `\setlines{n}{m}` where *n* is the instrument number and *m* is the number of lines.

2.20.4 Resetting normal layout parameters

The general size can only be changed with one of the commands `\smallmusicsize`, `\normalmusicsize`, `\largemusicsize`, or `\Largemusicsize`. Beyond that, the command `\resetlayout` will reset the following key layout parameters to their default values: `\staffbotmarg` (3\Interligne), `\stafftopmarg` (3\Interligne), `\interstaff` (9), number of lines per staff for all instruments (5); and will reset all clef symbols to standard clef symbols.

2.21 Lyrics

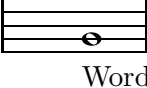
MusiX_TE_X itself doesn't manage lyrics very well. You should use `musixlyr` instead, a MusiX_TE_X extension package for lyrics handling by Rainer Dunker. The \TeX source and [documentation](#) are included in the MusiX_TE_X distribution.

But first we recall briefly on the older methods, who are still usefull when only a small number of words are involved.

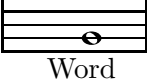
2.21.1 Native lyrics method: placing single words

2.21.1.1 Native MusiX_TE_X commands for lyrics

1. An obvious solution consists in using the command `\zcharnote` (expanded to the right), `\lcharnote` (expanded to the left), `\ccharnote` (centered), to post the text at any position (computed in `\internotes`) with respect to the lower line of the current staff. The pitch should be usually negative to have the text below the staff.

Example:  is coded by `\zcharnote{N}{Word}\wh g`.

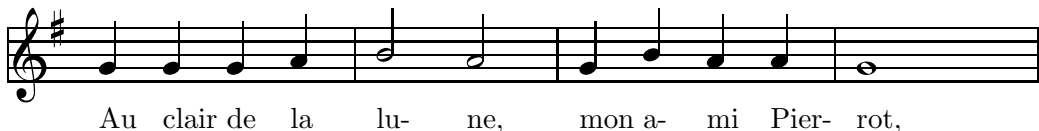
2. The vertical position can also be given with a number in the commands `\zchar` (expanded to the right), `\lchar` (expanded to the left), `\cchar` (centered). The number is internally multiplied by `\internote`.

Example:  is coded by `\cchar{-5}{Word}\wh g`.

3. Of easier use are the commands `\zsong` (right of the note), `\lsong` (left) and `\csong` (centered) which post the lyrics at the lower staff line *minus* the previous `\interinstrument` n or the `\staffbotmarg` quantity. These commands only have one argument, namely the lyrics text: `\zsong{text}` `\lsong{text}` `\csong{text}` Depending on the values of the inter-instrument spacings and margins, the resulting vertical position might be inappropriate. Then it can be changed for any specific n -th instrument until further change using

```
\setsongraise n{any  $\TeX$ -dimension}
```

As an example, the following French song

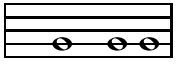


was coded as:

```
\generalsignature{1}
\startextract
\geometricsskip scale
\NOTes\zsong{Au }\qu g\zsong{clair }\qu g\en
\NOTes\zsong{de }\qu g\zsong{la }\qu h\en\bar
\NOTes\zsong{lu- }\hu i\zsong{ne, }\hu h\en\bar
\NOTes\zsong{mon }\qu g\zsong{a- }\qu i\en
\NOTes\zsong{mi }\qu h\zsong{Pier- }\qu h\en\bar
\NOTes\zsong{rot, }\wh g\sk\en
\endextract
```

2.21.1.2 Adapting note spacing for lyrics

The command `\hardlyrics{longword}` provides a spacing that is equal to the length of the text argument `{longword}`. In the same time the argument `{longword}` is saved in `\thelyrics`

As an example  is coded by: `\hardlyrics{clair}%`
`\notes\hsong{\thelyrics}\wh g\en`
`\notes\wh{gg}\en`

All notes with long lyrics need such a treatment. The commands only carry out on `\notes` (not on `\Notes`, `\Notes...`).

If you want to go back to the normal placing on an easy way, you simply can replace `'\hardlyrics'` by `'\softlyrics'`.

A complete score is given in example `glorias.tex` and in `gloriab.tex`, the latter exhibiting not only the song tune but also the organ accompaniment.

Alternate versions of `\hsong` are `\dhsong` which has a fixed length of `2\noteskip` and `\thsong` whose fixed length is `3\noteskip`. These are useful when the text is set below (or above) a collective coding of two or three notes.

2.21.2 Musixlyr

Lyrics are best handled by the `musixlyr` package by Rainer Dunker. The package can be used by inserting a file in your source code:

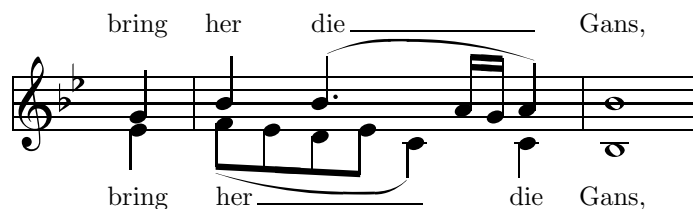
```
\input musixtex
\input musixlyr
...
```

The manual, the input file and a few examples can be downloaded as compact `musixlyr` packet [for Windows](#) or [for UNIX](#). Look at the manual for a detailed description. Here you can find an overview of the commands and an example of use.

Example	Explanation
<code>\setlyrics{sopr}{the ly_ -ric words_}</code>	defining the lyrics text
<code>\copylyrics{sopr}{alto}</code>	alto has same lyrics as soprano
<code>\appendlyrics{alto}{more words}</code>	alto lyrics is longer
<code>\assignlyrics2{sopr,alto}</code>	soprano and alto lyrics at staff 2
<code>\assignlyricsmulti{1}{2}{alto}</code>	assign alto lyrics to staff 2 of instrument 1
<code>\Notes\assignlyricshere{alto}\qa c\en</code>	assigning without staff number
<code>\auxlyr\assignlyrics{2}{sopr}</code>	assign soprano above staff 2
<code>\lyrrule\qu c...\lyrruleend\qu c</code>	make a melisma by hand
<code>\beginmel\qu c...\endmel\qu c</code>	melisma, same as word extension underline
<code>\lyr\qu c</code>	force a syllable from lyrics text at this note or rest
<code>\lyric{word}\qu c</code>	insert syllable 'word' at this note
<code>\loffset{2}{\lyric*{1.}}\qu c</code>	combine '1.' with regular syllable
<code>\lyrich{syl}\qu c</code>	same as <code>\lyric</code> , but with hyphenation
<code>\lyrich*{ }\qu c</code>	same as <code>\lyric*</code> , but with hyphenation
<code>\lyricsoff...\lyricson</code>	stop lyrics, then start again
<code>\nolyr\qu c</code>	no syllable at this note
<code>\llabel{labelname}name</code>	labelling a "go to" target in text
<code>\golyr{labelname}\qu c</code>	perform a jump, in music code
<code>\lyrpt,\qu c</code>	add a comma to the syllable under this note

<code>\lyrnop\qu c</code>	remove last character in syllable
<code>\lclyr\qu c</code>	make first character lower case
<code>\llyr\qu c</code>	left justified syllable
<code>\leftlyrtrue\qu c...</code> <code>\leftlyrfalse\qu c</code>	start and stop left justification as the default
<code>\lyroffset{-4}\qu c</code>	shift syllable 1 notehead to the left
<code>\minlyrspace{3pt}\qu c</code>	define minimum space between the words
<code>\forcelyrhyphenstrue\qu c</code>	always use a hyphen from now on
<code>\forcelyrhyphensfalse\qu c</code>	remove hyphen and make one word if necessary
<code>\showlyrshifttrue\qu c</code>	show the lyric shift
<code>\lyrraise{1}{a 2\Interligne}</code>	raise lyrics below staff 1 by 2\Interligne
<code>\lyrraisemulti{1}{2}{a 2\Interligne}</code>	raise alto lyrics above staff 2 of instrument 1
<code>\lyrraisehere{b 2\Interligne}\qu c</code>	raise lyrics below this staff by 2\Interligne
<code>\minlyrrulelength{2mm}</code>	melismas shorter than 2mm are not shown
<code>\minmulthyphens{15mm}</code>	distance between hyphens in 'hyphen melisma'
<code>\def\lyrhyphenchar{-}</code>	chose a hyphen character
<code>\setlyrics{\lyrlayout{\it}..}</code>	apply italics to all lyrics lines
<code>\verses{,\beginmel}\qu c</code>	initiate melisma at second verse
<code>\small\setlyrstrut</code>	adapt the vertical distance between lyrics lines
<code>\lyrstrutbox{10pt}</code>	(re)define the distance between the lyrics lines
<code>\lyrmodealter2</code>	attach lyrics of staff 2 to the upper voice
<code>\lyrmodealtermulti{1}{2}</code>	attach lyrics of instr. 1 staff 2 to the upper voice
<code>\lyrmodealterhere\qu c</code>	attach lyrics of this staff to the upper voice
<code>\lyrmodenormal2</code>	restore the default behaviour
<code>\lyrmodenormalmulti{1}{2}</code>	restore the default behaviour at staff 2 of instr. 1
<code>\lyrmodenormalhere\qu c</code>	restore the default behaviour of this staff
<code>\lyrlink</code>	linking 2 words with a '˘'
<code>\lowlyrlink</code>	same as <code>\lyrlink</code> but a little bit lower
<code>\resetlyrics</code>	set word pointer to the first word in all lyrics lines
<code>\enableauxlyrics</code>	don't use this anymore
<code>\setsongraise{1}{2\Interligne}</code>	same as <code>{\lyrraise}{1}{b 2\Interligne}</code>
<code>\auxsetsongraise{1}{2\Interligne}</code>	same as <code>{\lyrraisemulti}{1}{b 2\Interligne}</code>
<code>\oldlyrlinestart</code>	don't let the lyrics extent to the left margin

As a further illustration of the use of the commands, have a look at the following example²⁴:



which was coded as:

```
% define lyrics above the staff
\setlyrics{soprano}{bring her die Gans,}
% lyric beneath the staff are the same
\copylyrics{soprano}{alto}
% assign alto lyrics below staff 1 on the notes with stem down
```

²⁴The example is taken from the musixlyr manual.

```

\assignlyrics1{alto}
% assign soprano lyrics above staff 1 on the notes with stem up
\auxlyr{\assignlyrics1{soprano}}
% attach both lyrics to the upper voice
\lyrmodealter0
\generalsignature{-2}
% make place for the lyrics
\advance\stafftopmarg1\Interligne\advance\staffbotmarg2\Interligne
\startextract\addspace\afterruleskip
\NOTes\zqu g\ql e\en\bar
% start melisma in lower lyrics
\Notes\zqu i\beginmel\ibslurd0f\ibl0f{-1}\qb0{fe}\en
% start melisma in upper lyrics
\Notes\auxlyr\beginmel\ibsluruli\zqup i\qb0d\tqb0e\en
% end melisma in lower lyrics
\Notes\endmel\tbslurd0e\ql c\en
\notes\ibbu0h{-1}\qb0h\tqh0g\en
% start melisma in upper lyrics
\NOTes\auxlyr\endmel\tbsluruli\zqu h\ql c\en\bar
% The lyrics of the whole notes (without stem) must be given manually.
\NOTEs\auxlyr\lyr\zwh i\lyr\wh b\en
\endextract
\lyrmodenormal0

```

2.21.3 Getting enough vertical space for lyrics

Since songs are usually equivalent to a one-staff instrument (possibly with several voices) the recommended solution consists in adjusting the distance between instruments using either `\interinstrument=any` *TeX-dimension* to give more place below all instruments or using `\setinterinstrument` to make more space above. Note that `\setinterinstrument` defines spacing above and not below an instrument. Since lyrics are usually set below the staff, the first argument of a `\setinterinstrument` should be the song instrument number *minus one*.

In the case of a single staff tune, or if the song instrument is the lowest one, then additional place can be provided using `\staffbotmarg`.

2.21.4 Fine tuning the placement of the lyrics

When not using `\hardlyrics`, on short notes, sometimes the lyrics are shifted away from the notes or they collide with other words. This are a few approaches to get around this:

1. Making more music lines for the notes to go further apart. This could be done with `\mulooseness`.
2. Stretch a bar with short notes in it by i.e. replacing `\notes` by `\NOTes`.
3. Insert space between the notes by using `\sk`, `\hsk`, `\qsk`...
4. Stretch a bar with short notes in it by using the command `\scale`:

```
\scale{1.6}\notes..\en\scale{1}%
```

This method can be used in **PMX** but only with care, because it changes horizontal spacing in a way that **PMX** will not be aware of. It will not move bars to the next line, but will shorten the other bars on the line.

As an example, the lyrics of this music line are better placed by using `\scale` in the first bar and moving to the left the word 'mon'. Note that the hyphen is removed when there is no place for it:



The code of the second music line is (`\assignlyrics1{}` is only needed because lyrics are assigned before this in this manual):

```
\begin{music}
\input musixlyr
\resetlyrics
\setlyrics{v1}{Au clair de la lu-ne, \kernm1exmon a-mi Pier-rot,}%
\assignlyrics1{}\assignlyrics1{v1}%
\staffbotmarg2\Interligne\generalsignature{1}%
\startextract \geometricsskip scale
\scale{1.4}\notes\qu{gggh}\en\bar\scale{1}%
\Notes\hu{ih}\en\bar
\notes\qu{gihh}\en\bar
\Notes\wh g\en
\endextract
\end{music}
```

5. Moving a word in any direction

```
\setlyrics{alto}{\kernm3ex1.~~firstsyllable...}
left moving for numbering verses
\setlyrics{alto}{... \kern1exword...}
right moving a single word
\setlyrics{alto}{... \lower2pt\hbox{word}...}
lowering a single word
\setlyrics{alto}{... \raise2pt\hbox{word}...}
raising a single word
\def\strut{\vbox to 2\Interligne{}}\setlyrics{alto}{\lyrlayout{\strut}...}
controlling distance between verses
\lyrlayout{\vphantom{Mp(\lowlyrlink)}}
minimum distance between verses
\setbox\lyrstrutbox=\hbox{\vphantom{yM\lyrlink}}
redefining default lyrstrut
```

6. Placing of accents can be made easier as shows this example:

```
\catcode'\ä\active \defä{"a}
\catcode'\ö\active \defö{"o}
\catcode'\å\active \letå\aa
\setlyrics1{å ä ö} \assignlyrics1{}\assignlyrics11
\startextract
\Notes\qa{ggg}\en
\endextract
```

7. Using an 8-bit encoded characterset

This can be achieved by putting `\input plainenc\relax\inputencoding{cp850}` at the beginning of the file.

Here you must adapt the character set to your region. To do this, if necessary, replace `cp850` by `cp1250` (Czech, Croation region), `cp1251` (Russian, Bulgarian region), `cp1252` (most other European regions), `cp1253` (Greek region) or check your \TeX distribution for the right file.

This is the coding for the previous example by using an 8-bit character set:



```
\input plainenc\relax\inputencoding{cp850}
\input musixtex
\input musixlyr
\startpiece%
\znotes\zcharnote{16}{Poème naïf}\en%
\setlyrics1{á ä ö}\assignlyrics11%
\Notes\qa{ggg}\en%
\Endpiece\vfill\eject \bye
```

2.22 Embedding musical excerpts in text documents

Here we discuss the options for including music in text documents. The first decision is whether or not to use \LaTeX ²⁵. Because \LaTeX so effectively simplifies production of text-based \TeX documents, most users take that path, and most of the descriptions given here will assume that's the case²⁶.

If for some reason you choose not to use \LaTeX to emplace musical excerpts, the basic approach is simply to set off the musical parts between `\startpiece` or `\contpiece` and `\stoppiece` or `\endpiece`, or between `\startextract` and `\endextract`. But some details in what follows will also apply with no \LaTeX .

There are two basic approaches to embedding musical excerpts in \LaTeX documents. The first method is to directly include the \MusixTeX code in the \LaTeX source file. That will be the subject of the next subsection. The other is to create an EPS (encapsulated Postscript) file containing only the excerpt, and then “paste” it into the \LaTeX file. That will be covered in subsection 2.22.2.

The advantages of using the direct method are that all of the source code for all excerpts can be kept in the same file as the text, and that there is no limit on the length of the excerpt. The advantage of the EPS method is that you don't have to burden the \LaTeX source with any of the \MusixTeX paraphernalia. That in turn permits use of primitive versions of the \TeX compiler that may not have the capacity to handle the direct method (due to the number of registers consumed by \LaTeX and \MusixTeX). The disadvantages are that you must create and keep track of a separate \TeX and EPS file for every excerpt, and that the excerpt must not span any page breaks. On balance, the direct method is probably to be preferred.

2.22.1 Directly embedding excerpts in \LaTeX documents

To use the direct method, you should include “`musixtex`” as one of the arguments of the `\documentstyle` command. This will cause the file `musixtex.sty` to be loaded, so naturally you must make that file available in a place where \TeX can find it. That file simply inputs two other files, `musixtex.tex` and `musixltx.tex`, which again must obviously be available to \TeX .

Now you are in position to directly embed an excerpt by inserting code at the appropriate place in the source file. The most common type of excerpt is one that occupies less than a full

²⁵We'll assume a user wanting to embed a musical excerpt in a \LaTeX document is already familiar with the fundamentals of \LaTeX . For more information about it, see for example the manual *LaTeX: A Document Preparation System* by Leslie LAMPORT

²⁶Please do not be confused; while \LaTeX is recommended for text-based documents containing musical excerpts, its use is definitely discouraged for ordinary self-contained musical scores of any sort.

line and is to be horizontally centered. In that case, the extract should begin with the command `\begin{music}`, followed by any preliminary commands. Then, instead of `\startpiece`, use `\startextract`. Now comes the normal MusiX_TE_X coding. Finally, end the extract with `\endextract` instead of `\endpiece` or `\stoppiece`, followed by `\end{music}`.

To terminate an extract without any bar line, use `\zendextract` instead of `\endextract`.

To create a left-justified excerpt, use the sequence `\let\extractline\leftline`.

If several extracts are to be placed on the same line, you can redefine `\extractline` as demonstrated in the following example²⁷:

```
\begin{music}
\let\extractline\hbox
\hbox to \hsize{%
\hss\startextract ... \endextract\hss%
\hss\startextract ... \endextract\hss}
\end{music}
```



An even shorter type of extract is one that is embedded *within* a line of text. To insert MusiX_TE_X symbols within a line of text, you could begin by defining `\notesintext` as follows²⁸:

```
\begin{music}
\makeatletter
\def\notesintext#1{%
  {\let\extractline\relax
   \setlines10\smallmusicsize \nobarnumbers \nostartrule
   \staffbotmarg0pt \setclefsymbol1\empty \global\clef@skip0pt
   \startextract\addspace{-\afterruleskip}#1\zendextract}}
\makeatother
\end{music}
```

Then, for example, the code

```
Use \raisebox{0ex}[0ex][0ex]{\notesintext{\notes\rq11\qu2\en}}
not \raisebox{0ex}[0ex][0ex]{\notesintext{\notes\ql2\lqu1\en}}
```

would produce

Use  not . The `\raisebox` voids the vertical space that is introduced by the notes.

Finally, you may want to insert an extract containing more than one line of music. Once having mastered MusiX_TE_X, this is the easiest of all: Between `\begin{music}` and `\end{music}`, use *exactly* the same coding you would to make an ordinary score.

The best way to learn how to apply these methods is to study `musixdoc.tex`, the source file for this document²⁹.

2.22.2 Embedding musical excerpts as encapsulated postscript files

To use this method of including excerpts, you first must create a separate MusiX_TE_X input file for each excerpt. Process each such file with `TEX` and `musixflx` to generate a `.dvi` file. Generate

²⁷The macro `\extractline` is defined once and for all in `musixtex.tex` as `\centerline`. You might think that the suggested coding would permanently redefine `\extractline`, thereby upsetting the normal function of `\startextract ... \endextract` for subsequent use. But it doesn't, because any actions within `\begin{music}... \end{music}` are local, not global.

²⁸provided by Rainer DUNKER

²⁹Do note, however, that `musixdoc.tex` includes `musixdoc` but not `musixtex` as an argument of `\documentstyle`. The former performs the functions of the latter as well as numerous tasks peculiar to this particular document.

a postscript file from each `.dvi` using `dvips`. Then convert each postscript file to an `.eps` file. One way to do that is with `ghostscript` and—if you are using Windows—`GSview`. In general this is possible only for single-page postScript files.

To set up your \LaTeX document for including `.eps` files, you must post the command `\usepackage[dvips]{graphicx}` in the preamble of the document. Now, you may include each `.eps` file at the appropriate place in the \LaTeX document with a command like `\includegraphics{sample.eps}`.

2.22.3 Issues concerning `\catcodes`

MusiX \TeX uses the following symbols differently from plain \TeX : `>`, `<`, `|`, `&`, `!`, `*`, `.`, and `:`. The symbols are given their special meanings by executing the macro `\catcodesmusic`, and are restored to their plain \TeX meanings with `\endcatcodesmusic`. When setting either a self-contained score or a musical extract, you normally need not worry about this at all, because `\startpiece` or `\startextract` executes `\catcodesmusic` and `\endpiece` or `\endextract` executes `\endcatcodesmusic`. But there are some special situations where you might need to use these catcode-modifying macros explicitly. One is if you were to define a personalized macro outside `\startpiece ... \endpiece`, but which incorporated any of the symbols with their MusiX \TeX meanings. Another would be if you wished to have access to facilities enabled by alternate style files such as `french.sty` which change `\catcodes` themselves. In such cases, provided you have input `musixtex.tex`, you can always invoke `\catcodesmusic` to set the `\catcodes` at their MusiX \TeX values, and `\endcatcodesmusic` to restore them to their prior values.

2.23 Extension Library

All following files are invoked by saying `\input filename`. Most of them are fully compatible with MusiX \TeX in that they do not redefine any existing macros but rather provide additional functionality. In future versions of MusiX \TeX we may very well incorporate many of them directly into `musixtex.tex`, but for now we leave them separate.

2.23.1 curly

Allows you to group more than 1 instrument with curly brackets (see 2.1.4).

2.23.2 musixadd

Increases the number of instruments, slurs and beams from six to nine.

2.23.3 musixbm

Provides 128th notes, either with flags or with beams, via the commands `\ibbbbbbu`, `\ibbbbbb1`, `\nbbbbbu`, `\nbbbbb1`, `\tbbbbbu`, `\tbbbbb1`, `\Ibbbbbu`, `\Ibbbbb1`, `\cccccu`, `\ccccca`, `\zccccu`, `\cccccl` and `\zcccc1`.

2.23.4 musixbbm

Provides 256th notes, only with beams, via the commands `\ibbbbbbbu`, `\ibbbbbbb1`, `\nbbbbbbu`, `\nbbbbbb1`, `\tbbbbbbu`, `\tbbbbbb1`, `\Ibbbbbbu` and `\Ibbbbbb1`.

2.23.5 musixcho

Enables certain macros intended for choral music³⁰. Provides the following commands: `\biglbrace`, `\bigrbrace`, `\braceheight`, `\Dtx` and `\Drtx` for two-line text, `\Ttx` and `\Trtx` for three-line text, `\Qtx` and `\Qrtx` for four-line text. To eliminate zigzagging lyrics lines, all multiple line texts are automatically vertically justified with the macro `\ChoirStrut`, defined as `\vphantom{\^Wgjpqy}`.

The macros `\tx{text}`, `\rtx{text}` cause song text to be left-justified on the insertion point rather than centered. `\hf{m}` sets a text continuation rule of length m `\noteskip`.

Consult the source file `musixdoc.tex` to see the coding of the following example:

Sopran
Alt
Tenor
Bass

1. Oh freedom, oh freedom, oh freedom
2. No more weepin', no more weepin', no more weepin'
3. No more moanin', no more moanin', no more moanin'
4. There'll be singin', there'll be singin', there'll be singin'

} over me, over

2.23.6 musixcpt

Empowers M_usiX_TE_X to run files created under MusicT_EX, the predecessor of M_usiX_TE_X, such as some of the examples provided by Daniël TAUPIN. It is not needed for any files created under M_usiX_TE_X, and is included mainly for historical completeness.

2.23.7 musixdat

Enables the command `\today`, which sets the current date in one of several possible languages. The language is selected by an optional preparatory command `\date...`. The default is `\dateUSenglish`, but this can be changed, either at the end of `musixdat.tex` for a permanent change, or right before issuing `\today`. Available choices and sample results are summarized below:

<code>\dateUSenglish</code>	November 29, 2006
<code>\dateaustrian</code>	29. November 2006
<code>\dateenglish</code>	29th November 2006
<code>\datefrench</code>	29 novembre 2006
<code>\dategerman</code>	29. November 2006

2.23.8 musixdbr

Enables dashed and dotted bar lines (see 2.11.4).

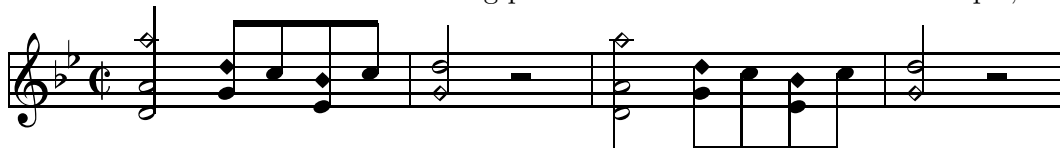
2.23.9 musixdia

Enables notes with diamond-shaped heads as follows:

³⁰Remember that we now recommend using `musixlyr` for any except the simplest lyrics. The extension `musixcho` is only for those diehards who choose to ignore this advice

- Solid note heads (◆) are obtained using the macros `\yqu`, `\yqup`, `\yqupp`, `\yql`, `\yqlp`, `\yqlpp`, `\yzq`, `\yzqp`, `\yzqpp`, `\yqb`, `\ycu`, `\yccu`, `\ycccu`, `\ycccuu`, `\ycl`, `\ycll`, `\yclcl`, `\yclccl`, `\yclccll`, `\ycup`, `\ycupp`, `\yclp`, `\yclpp`. (Think of *dy*amond). A solid diamond with no stem is obtained with `\ynq` (spacing) or `\yznq` (non-spacing).
- Open note heads (◇) are obtained using the macros `\dqu`, `\dqup`, `\dqupp`, `\dql`, `\dqlp`, `\dqlpp`, `\dzq`, `\dzqp`, `\dzqpp`, `\dqb`, `\dcu`, `\dccu`, `\dccccu`, `\dccccu`, `\dcl`, `\dccl`, `\dccccl`, `\dccccll`, `\dcup`, `\dcupp`, `\dclp`, `\dclpp`. (Think of *d*iamond). An open diamond with no stem is obtained with `\dnq` (spacing) or `\dznq` (non-spacing).

One use of these note heads is for a string part with *harmonic notes*. For example,



was coded as follows:

```
\generalsignature{-2}
\generalmeter\allabreve
\startextract
\NOTes\dzq o\zh d\hu h\enotes
\Notes\ibu0k0\zq g\yqb0k\qb0j\zq e\yqb0i\tbu0\qb0j\enotes
\bar
\NOTes\dzq g\hu k\enotes
\NOTes\hpause\enotes
\bar
\NOTes\dzq o\zh d\hl h\enotes
\Notes\ibl0e0\zq g\yqb0k\qb0j\zq e\yqb0i\tbl0\qb0j\enotes
\bar
\NOTes\dzq g\hu k\enotes
\NOTes\hpause\enotes
\endextract
```

Another use is for percussion parts. In fact the file `musixdia.tex` is automatically loaded if you input `musixper.tex` (see 2.23.18).

2.23.10 musixeng

This package is provided for music typesetters who are allergic to the default rest names, which are taken from French, German or Italian. It does not provide new features, only new command names:

<i>original</i>	<i>alternate</i>
<code>\PAUSE</code>	<code>\Qwr</code>
<code>\PAuse</code>	<code>\Dwr</code>
<code>\liftpause</code>	<code>\liftwr</code>
<code>\pausep</code>	<code>\wrp</code>
<code>\pause</code>	<code>\wr</code>
<code>\liftpause</code>	<code>\lifthr</code>
<code>\hpausep</code>	<code>\hrp</code>
<code>\hpause</code>	<code>\hr</code>
<code>\qp</code>	<code>\qr</code>
<code>\ds</code>	<code>\er</code>
<code>\qs</code>	<code>\eer</code>
<code>\hs</code>	<code>\eeer</code>
<code>\qqs</code>	<code>\eeeer</code>

2.23.11 musixext

Defines the following two specific macros:

`\slide{p}{x}{s}` , which provides a glissando starting at pitch p and extending for x `\internotes` with slope s (ranging from -8 to 8).

`\raggedstoppiece` , which inhibits right-justification of the last line of a score.

2.23.12 musixfl

Enables modification of ledger lines. Ledger lines normally exceed the width of a note head by 25 percent in each direction. If the space between notes is insufficient, the ledger lines of consecutive notes may meet, creating visual ambiguities. Therefore, MusiX_TE_X shortens the ledger lines if notes are set so close together that the ledger lines may meet. But because MusiX_TE_X does not know whether consecutive notes need ledger lines, this automatic shortening may be superfluous. The extension file `musixfl1.tex` allows this feature to be switched off and on. Upon inputting `musixfl1.tex`, the automatic shortening of ledger lines is switched off. From then on, it may be switched on again using `\autoledgerlines` and switched off again using `\longledgerlines`. Both macros have global effect.

The following example shows that narrowly set scales look better with `\autoledgerlines` (the default behavior), while single notes requiring ledger lines look better with `\longledgerlines`.



2.23.13 musixgre

Gregorian chant is often coded using four line staves (see section 2.20.3) and using special notes called *neumes* (which are described later in this section). It also requires special clefs. One way to substitute them for the modern ones is for example with commands like

```
\setaltoclefsymbol3\gregorianCclef
```

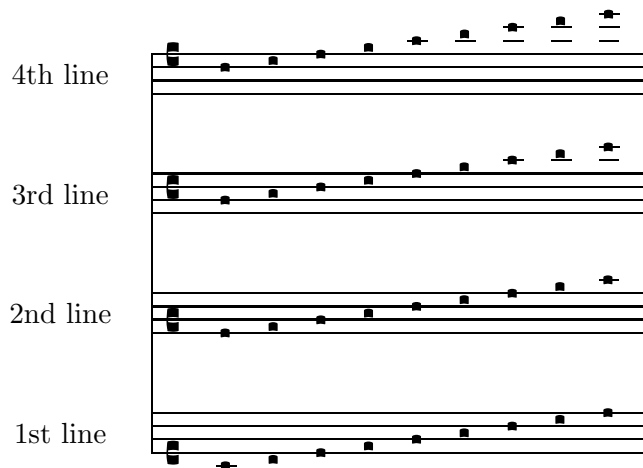
or

```
\setbassclefsymbol3\gregorianFclef ,
```

which will cause instrument number 3 to display the selected gregorian clef. The standard clefs can be restored for every instrument with `\resetclefsymbols`. Note that when using this method you must specify whether to substitute for the bass or alto clef symbol (there is no treble clef in gregorian chant). The reason is that MusiX_TE_X selects and raises the F and C clefs differently, according to the arguments of the `\setclef` command. Therefore, if one had substituted any F clef symbol while saying `\setclef1{1000}`, then an F clef would duly appear on the staff, but it would be set at the position of an alto clef, thus seriously misleading the musician.

Another method of clef substitution employs `\setclefsymbol` (see section 2.14.3), which substitutes the clef given by the second argument *for all clef symbols* of the instrument given by the first, regardless of the actual musical meaning of the new clef symbol. This method is generally appropriate only if you want to change the clef symbol(s) of an instrument for the whole of the score.

As an example, the same gregorian scale has been written with a gregorian C clef on all four lines of the staff:




The coding was:

```
\instrumentnumber{4}
\setname1{1st line} \setname2{2nd line} \setname3{3rd line} \setname4{4th line}
\setlines1{4}\setlines2{4}\setlines3{4}\setlines4{4}
\sepbarrules
\generalmeter{\empty}
\setclef1{1000} \setclef2{2000} \setclef3{3000} \setclef4{4000}
\setaltoclefsymbol1\gregorianCclef
\setaltoclefsymbol2\gregorianCclef
\setaltoclefsymbol3\gregorianCclef
\setaltoclefsymbol4\gregorianCclef
\startextract
\Notes\squ{abcdefghi}&\squ{abcdefghi}&\squ{abcdefghi}&\squ{abcdefghi}&\enotes
\endextract
```

All of the special gregorian symbols available in MusiX_TE_X are described in the following subsections.

2.23.13.1 Clefs

- Gregorian C clef:  = `\gregorianCclef`, normally activated for instrument n with the command `\setaltoclefsymbol{n}\gregorianCclef`

- Gregorian F clef: F = `\gregorianFclef`, normally activated with the command `\setbassclefsymbol{n}\gregorianFclef`

2.23.13.2 Elementary symbols

- Diamond shaped *punctum* (This has a different shape compared to the percussion diamond): \blacklozenge = `\diapunc{p}` .
- Square *punctum*: \blacksquare = `\squ{p}` or `\punctum{p}` .
- Left stemmed *virga* (not in the 1905 gregorian standard): \blacklshoe = `\lsqu{p}` .
- Right stemmed *virga*: \blackrshoe = `\rsqu{p}` or `\virga{p}` .
- *Apostropha*: \blacklozenge = `\apostropha{p}` .
- *Oriscus*: \blacklozenge = `\oriscus{p}` .
- *Quilisma*: \blacklozenge = `\quilisma{p}` .
- *Punctum auctum* (up): \blacklozenge = `\punctumauctup{p}` .
- *Punctum auctum* (down): \blacklozenge = `\punctumauctdown{p}` .
- Diamond shaped *punctum auctum* (down): \blacklozenge = `\diapunctumauctdown{p}` .
- *Punctum deminutum*: \blacklozenge = `\punctumdeminutum{p}` .
- *Apostropha aucta*: \blacklozenge = `\apostropha aucta{p}` .

All non-*liquescentes* symbols have non-spacing variants, namely `\zdiapunc`, `\zsqu`, `\zlsqu`, `\zrsqu`, `\zapostropha` and `\zoriscus`.

2.23.13.3 Plain complex neumes

Other *neumes* can be obtained by combining two or more of these symbols. Since *neumes* have a special note head width, an additional shifting macro is provided, namely `\groff`. It is similar to `\roff`, but the offset is smaller. For use with complex neumes, another shifting macro is provided, namely `\dgroff`, which causes an offset twice the offset of `\groff`.

Since most of these symbols depend on relative pitches of their components, we cannot provide all possible compact combinations as single symbols. The ones that are available in `musixgre` are described below. In the following, p_1 , p_2 , p_3 , and p_4 represent pitches specified as usual. Please refer to the source file `musixtex.tex` if you wish to see the coding of those examples for which it is not quoted here.

`\bivirga{p_1}{p_2}` , for example:



This example was coded as:

```

\instrumentnumber 1
\setstaves 1 1
\setlines 1 4
\setclef 1{3000}
\setaltoclefsymbol 1 \gregorianCclef
\startextract

```

```

\notes \bivirga ab\enotes
\notes \bivirga cc\enotes
\endextract

```

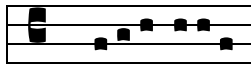
`\trivirga{p1}{p2}{p3}` , for example:



`\bistropa{p1}{p2}` , for example:



`\tristropa{p1}{p2}{p3}` , for example:



`\clivis{p1}{p2}` , for example:



`\lclivis{p1}{p2}` , for example:



`\podatus{p1}{p2}` , for example:



`\podatusinitedebilis{p1}{p2}` , for example:



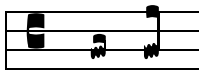
`\lpodatus{p1}{p2}` , for example:



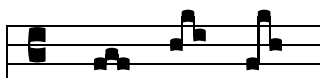
`\pesquassus{p1}{p2}` , for example:



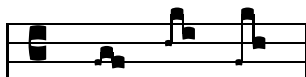
`\quilismapes{p1}{p2}` , for example:



`\torculus{p1}{p2}{p3}` , for example:



`\torculusinitedebilis{p1}{p2}{p3}` , for example:



`\Porrectus{p1}{p2}{p3}` , for example:



coded:

```
\notes \Porrectus bab\enotes
\notes \Porrectus bac\enotes
\notes \Porrectus bNd\enotes
\notes \Porrectus bMe\enotes
\notes \Porrectus bLe\enotes
```

`\Porrectus` exists in four different shapes, depending on the difference between first and second argument. The constraint is that

$$p_1 - 4 \leq p_2 \leq p_1 - 1$$

otherwise a diagnostic occurs. Note also that `\bporrectus` provides the first curved part of the `porrectus` command, if you should need it. It has two arguments, the starting pitch and the lower pitch.

`\Porrectusflexus{p1}{p2}{p3}{p4}` , for example:



coded:

```
\notes \Porrectusflexus bacN\enotes
\notes \Porrectusflexus bNdb\enotes
```



```
\notes \Porrectusflexus bMeb\notes
\notes \Porrectusflexus bLea\notes
```

`\climacus{p1}{p2}{p3}` , for example:



`\climacusresupinus{p1}{p2}{p3}{p4}` , for example:



`\lclimacus{p1}{p2}{p3}` , for example:



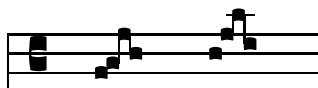
`\scandicus{p1}{p2}{p3}` , for example:



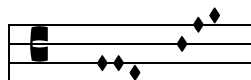
`\salicus{p1}{p2}{p3}` , for example:



`\salicusflexus{p1}{p2}{p3}{p4}` , for example:



`\trigonus{p1}{p2}{p3}` , for example:



2.23.13.4 Liquesens complex neumes

`\clivisauctup{p1}{p2}` , for example:



`\clivisauctdown{p1}{p2}` , for example:



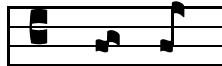
`\podatusauctup{p1}{p2}` , for example:



`\podatusauctdown{p1}{p2}` , for example:



`\pesquassusauctdown{p1}{p2}` , for example:



`\quilismapesauctdown{p1}{p2}` , for example:



`\torculus auctdown{p1}{p2}{p3}` , for example:



`\Porrectusauctdown{p1}{p2}{p3}` , for example:



`\climacusauctdown{p1}{p2}{p3}` , for example:



`\scandicus auctdown{p1}{p2}{p3}` , for example:



`\salicus auctdown{p1}{p2}{p3}` , for example:



`\clivisdeminut{p1}{p2}` , for example:



`\podatusdeminut{p1}{p2}` , for example:



`\torculusdeminut{p1}{p2}{p3}` , for example:



`\torculusdebilis{p1}{p2}{p3}` , for example:



`\Porrectusdeminut{p1}{p2}{p3}` , for example:



`\climacUSDeminut{p1}{p2}{p3}` , for example:



`\scandicusdeminut{p1}{p2}{p3}` , for example:



2.23.14 musixgui

Provides macros for typesetting modern style *guitar tablatures*. For example:

The image shows a musical score for the song "We wish you a merry Christmas" in G major, 3/4 time. The score consists of two staves of music. Above the first staff, five guitar chord diagrams are shown: G, C (with a barre at the 5th fret), e/H (with a barre at the 5th fret), A7 (with a barre at the 5th fret), and D. Above the second staff, seven guitar chord diagrams are shown: D/c, B7, e (with a barre at the 5th fret), G/d, C6, D7, and G. The lyrics are: "We wish you a merry christmas, we wish you a merry christmas, we wish you a mer-ry christ-mas and a hap-py new year."

The macro `\guitar` sets the grid, chord name, barre type, and on-off indicators for the strings. For example, the first chord in above example was coded as

```
\guitar G{o-----\gbarre3\gdot25\gdot35\gdot44
```

where the first argument is the text to be placed above the grid, the second is empty (relative barre), and the next six characters indicate if the string is played or not with either `x`, `o` or `-`. The dots are set with `\gdot sb` where the *s* is the string and *b* is the barre. The rule is set with `\gbarre b` where *b* indicates the position of the barre.

The whole symbol may be vertically shifted with `\raiseguitar{n}`, where *n* is a number in units of `\internote`. When using guitar tablatures, it might be useful to reserve additional space above the chord by advancing `\stafftopmarg` to something like `stafftopmarg=10\Interligne`.

For frequently used chords, it might be useful to define your own macros, e.g.

```
\def\Dmajor{\guitar D{x-----\gdot42\gdot53\gdot62}}%
```

2.23.15 musixlit

Provides a notation style intermediate between gregorian and baroque, for example

The image shows a musical score for the song "Il nous a signés de son sang" in G major, 8/8 time. The score consists of three staves: a vocal line at the top, a piano right-hand line in the middle, and a piano left-hand line at the bottom. The vocal line uses a notation style where notes are represented by small black squares on a staff, with stems and beams. The lyrics are: "Il nous a signés de son sang Et nous avons é - té pro-té-gés. Al - le - lu - ia !"

Il nous a signés de son sang Et nous avons é - té pro-té-gés. Al - le - lu - ia !

This package provides:

- `\oldGclef` which replaces the ordinary G clef with an old one, using (for instrument 2 as an example): `\settrebleclefsymbol12\oldGclef`
- `\cqu p` provides a square headed quarter note with stem up at pitch p .
- `\cq1 p` provides a square headed quarter note with stem down at pitch p .
- `\chu p` provides a square headed half note with stem up at pitch p .
- `\chl p` provides a square headed half note with stem down at pitch p .
- `\cnqu p` and `\cnq1 p` provide a stemless square headed quarter note at pitch p .
- `\cnhu p` and `\cnhl p` provide a stemless square headed half note at pitch p .
- `\Hpause p n` provides an arbitrary length pause at pitch p and of length n `\noteskip`. However, in the first of the above example, this feature has been used to denote an arbitrary length note rather than a rest!
- `\Hlonga p n` provides an arbitrary length note at pitch p and of length n `\noteskip`. This feature has been used to denote an arbitrary length note in the second of the above examples.
- `\shortbarrules` has been used to provide bar rules shorter than the staff vertical width.
- `\interbarrules` has been used to provide bars between the staves, rather than over them. This is an arbitrary question of taste...

2.23.16 musixlyr

Enables the recommended method for adding lyrics to a score (see 2.21.2).

2.23.17 musixmad

Increases the number of instruments, slurs and beams up to twelve. When using this extension, it is not necessary to explicitly input `musixadd.tex`.

2.23.18 musixper

Provides special symbols intended for percussion parts. Included are a *drum clef*—comprising two vertical parallel lines—and notes with various specially shaped heads. The note symbols that are available are as follows:

- The \otimes symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceded with a “dc” (think of *d*agonal *c*ross). Available are `\dcqu`, `\dcq1`, `\dcqb`, `\dczq`, `\dccu`, `\dccc`, `\dccc1` and `\dccc1`.

- The \times symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceded with a “dh” (think of *d*iagonal cross *h*alf *o*pen). Available are `\dhqu`, `\dhql`, `\dhqb`, `\dhzq`, `\dhcu`, `\dhccu`, `\dhcl` and `\dhccl`.
- The \times symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceded with a “do” (think of *d*iagonal cross *o*pen). Available are `\doqu`, `\doql`, `\doqb`, `\dozq`, `\docu`, `\doccu`, `\docl` and `\doccl`.
- The \times symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceded by “x” (e.g. for spoken text of songs). Available are `\xqu`, `\xql`, `\xqb`, `\xzq`, `\xcu`, `\xccu`, `\xcl` and `\xccl`.
- The \times symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceded by “ox” . Available are `\oxqu`, `\oxql`, `\oxqb`, `\oxzq`, `\oxcu`, `\oxccu`, `\oxcl` and `\oxccl`.
- The \blacktriangleright symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceded by “ro” (think of *r*hombus). Available are `\roqu`, `\roql`, `\roqb`, `\rozq`, `\rocu`, `\roccu`, `\rocl` and `\roccl`.
- The \blacktriangleleft symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceded by “tg” (think of *t*riangle). Available are `\tgqu`, `\tgql`, `\tgqb`, `\tgzq`, `\tgcu`, `\tgccu`, `\tgcl` and `\tgccl`.
- The $+$ symbol which is obtained using the `\qu`, `\qb`, `\cu`, etc. macros preceded by “k” . Available are `\kqu`, `\kql`, `\kqb`, `\kzq`, `\kcu`, `\kccu`, `\kcl` and `\kccl`.

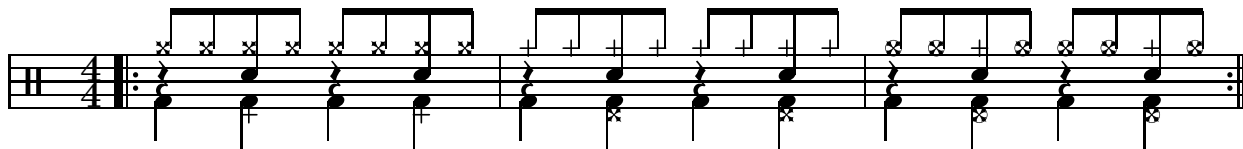
The diamond shaped noteheads described in section 2.23.9 are also available, because `musixper.tex` inputs `musixdia.tex`.

If any of the foregoing notes need to be dotted, you must use the explicit dotting macros `\pt`, `\ppt`, or `\pppt` as described in section 2.4.6.

Since the usage of these note symbols is not standardized, it would be wise to include in the score a explanation of which symbol corresponds to which specific percussion instrument.

A special *drum clef*—comprising two heavy vertical bars—can be made to replace the normal clef for the n -th intrument by saying `\setclefsymbol{n}\drumclef` . To cause this to appear at the right vertical position, the instrument should previously have been assigned a treble clef (or not explicitly assigned any clef, thereby giving it a treble clef by default).

Percussion music might be written on a staff with either one or five lines. If there are several different percussions instruments it may be useful to use a five-line staff with a drum clef, and differentiate the instruments by the type of the note heads and the apparent pitch of the note on the staff. Here is an example of the latter³¹:



Its coding is

```
\begin{music}
\instrumentnumber{1}
\generalmeter{\meterfrac{4}{4}}
\setclefsymbol1\drumclef
\parindent0pt\startpiece
\leftrepeat
\Notes\zql f\rlap\qp\ibu0m0\xqb0{nn}\enotes
\Notes\kzq d\zql f\zq j\xqb0n\tbu0\xqb0n\enotes
```

³¹provided by Agusti MARTÍN DOMINGO

```

\Notes\zql f\rlap\qp\ibu0m0\xqb0{nn}\enotes
\Notes\kzq d\zql f\zq j\xqb0n\tbu0\xqb0n\enotes
\bar
\Notes\zql f\rlap\qp\ibu0m0\kqb0{nn}\enotes
\Notes\xzq d\zql f\zq j\kqb0n\tbu0\kqb0n\enotes
\Notes\zql f\rlap\qp\ibu0m0\kqb0{nn}\enotes
\Notes\xzq d\zql f\zq j\kqb0n\tbu0\kqb0n\enotes
\bar
\Notes\zql f\rlap\qp\ibu0m0\oxqb0{nn}\enotes
\Notes\oxzq d\zql f\zq j\kqb0n\tbu0\oxqb0n\enotes
\Notes\zql f\rlap\qp\ibu0m0\oxqb0{nn}\enotes
\Notes\oxzq d\zql f\zq j\kqb0n\tbu0\oxqb0n\enotes
\setrightrepeat\endpiece
\end{music}

```

Here is an example of a single-line percussion staff using diamond-shaped note heads:

The image shows a musical score with three staves. The top staff is labeled 'monks' and contains diamond-shaped note heads on a single-line staff. The middle staff is labeled 'drum' and contains diamond-shaped note heads on a single-line staff with a G-clef. The bottom staff is labeled 'keyboard' and consists of two staves (treble and bass clef) with standard note heads. The music is in 2/4 time and spans three measures.

which is coded as follows:

```

\parindent 19mm
\instrumentnumber{3}
\setname1{keyboard} \setname2{drum} \setname3{monks}
\setlines2{1}
\setlines3{4}
\setinterinstrument1{-2\Interligne}% less vertical space above
\setinterinstrument2{-2\Interligne}% and below the percussion
\sepbarrules
\setsign1{-1} % one flat at keyboard
\generalmeter{\meterfrac24}
\setmeter3{\empty}
\setclef3{\alto}
\setclef1{\bass}
\setstaves1{2} % 2 staves at keyboard
\setclefsymbol3{\gregorianCclef} % gregorian C clef at instrument 3
\setclefsymbol2{\drumclef} % cancel G clef at instrument 2
\startextract
\Notes\hu F|\zh c\hu h&\dnq4&\squ{acd}\enotes\bar
\Notes\qu I|\zq N\qu d&\qp&\diapunc f\enotes
\Notes\qu J|\zq a\qu e&\ynq4&\diapunc f\enotes\bar
\notes\hu G|\zh b\hu d&\dnq4&\zsqu d\rsqu g\squ{hgh}\enotes
\endextract

```

2.23.19 musixpoi

Adds definitions of less common singly and doubly dotted notes. Available are `\ccup`, `\zccup`, `\cclp`, `\zcclp`, `\ccupp`, `\zccupp`, `\cclpp`, `\zcclpp`, `\cccup`, `\zcccup`, `\ccclp`, `\zcclp`, `\cccupp`, `\zcccupp`, `\cccclp`, `\zcccclp`, `\cccupp`, `\zcccupp`, `\cccclpp`, `\zcccclpp`, `\ccccup`, `\zccccup`, `\cccclp`, `\zcccclp`, `\ccccupp`, `\zccccupp`, `\cccclpp` and `\zcccclpp`.

2.23.20 musixps

Activates type K postscript slurs, ties, and hairpins (see 2.10.2).

2.23.21 musixstr

Provides bowing and other symbols for *string instruments*³². The symbol can be posted at the desired position using `\zcharnote{p}{command}`. The available symbols and their meanings are as follows:

- ▣ : `\AB` or `\downbow` down-bow
- ∨ : `\AUF` or `\upbow` up-bow
- ◁ : `\SP` at the top of bow
- ⇨ : `\FR` at the nut of bow
- ↔ or ↔ : `\GB` or `\Gb` whole bow
- ← or ← : `\UH` or `\Uh` lower half of bow
- or → : `\OH` or `\Oh` upper half of bow
- ↔ or ↔ : `\MI` or `\Mi` middle of bow
- ↔ or ↔ : `\UD` or `\Ud` lower third of bow
- ↔ or ↔ : `\OD` or `\Od` upper third of bow
- + : `\Pizz` left hand pizzicato or trill

2.23.22 musixsty

Provides certain text-handling facilities for titles, footnotes, and other items not related to lyrics. It should not be used with L^AT_EX. It includes

- definitions for commonly used fonts such as `\tenrm`, `\eightrm`, etc.;
- definitions of `\hsize`, `\vsize`, `\hoffset`, `\voffset` suitable for A4 paper; those using other sizes may wish to modify it once and for all;
- a set of text size commands:

- `\eightpoint` which sets the usual `\rm`, `\bf`, `\sl`, `\it` commands to 8 point font size;
- `\tenpoint` which sets the usual `\rm`, `\bf`, `\sl`, `\it` commands to 10 point font size;
- `\twlpoint` to get 12 point font size;
- `\frtpoint` to get 14.4 point font size;
- `\svtpoint` to get 17.28 point font size;
- `\twtypoint` to get 20.74 point font size;
- `\twfvpoint` to get 24.88 point font size;

- commands for creating titles:

³²provided by Werner ICKING

- `\author` or `\fullauthor` to be put at the right of the first page, below the title of the piece; the calling sequence is, for example


```
\author{Daniel TAUPIN\organiste \a Gif-sur-Yvette}
```

 where the `\\` causes the author's name to be displayed on two lines;
 - `\shortauthor` to be put at the bottom of each page;
 - `\fulltitle` which is the main title of the piece;
 - `\subtitle` is displayed below the main title of the piece;
 - `\shorttitle` or `\title` which is the title repeated at the bottom of each page;
 - `\othermention` which is displayed on the left of the page, vertically aligned with author's name. It may contain `\\` to display it on several lines;
 - `\maketitle` which displays all the previous stuff;
- commands for making *footnotes*:
 - The normal Plain- \TeX `\footnote` command, which has two arguments—not just one as in \LaTeX —namely the label of the footnote, which can be any sequence of characters, and the text of the footnote. This command does not work inside boxes, so it cannot be issued within music;
 - The `\Footnote` command, which counts the footnotes and uses a number as the label of the footnote (equivalent to \LaTeX 's `\footnote` command). The same restriction as with `\footnote` applies concerning its use within the music coding;
 - The `\vfootnote` command, taken from the Plain- \TeX , which places a footnote at the bottom of the current page, but does not put the footnote label at the place the command is entered in the main text. This also may not be used within music, but if a footnote is needed whose reference lies inside the music, it can be entered in two steps:
 1. manually insert the reference inside the music, using e.g. `zcharnote`;
 2. post the footnote itself with `\vfootnote` outside the music, either before `\startpiece` or between `\stoppiece` and `\contpiece` or equivalent commands.

2.23.23 musixtmr

Replaces the standard `musixtex` fonts by the Times series fonts, as default (see 2.16.2).

2.23.24 musixtri

Provides triply dotted note symbols. Available are: `\lpppt`, `\whppp`, `\zwppp`, `\huppp`, `\hlppp`, `\zhppp`, `\zhuppp`, `\zhlppp`, `\quppp`, `\qlppp`, `\zquppp`, `\zqlppp`, `\qppp`, `\cuppp`, `\zcuppp`, `\clppp`, `\zclppp`, `\qbppp` and `\zqbppp`.

2.23.25 tuplet

Causes the figure in xtuplets to be printed within a small gap in the bracket (see 2.17.3).

Chapter 3

Acquiring, Installing, and Using MusiX_{TEX}

This chapter will assume that $\text{T}_{\text{E}}\text{X}$ is installed. Most UNIX or Linux systems come with $\text{T}_{\text{E}}\text{X}$ already installed. If using MS Windows, you'll need to have manually installed some version of $\text{T}_{\text{E}}\text{X}$. MiK $\text{T}_{\text{E}}\text{X}$ is the most common variant by far and therefore will be assumed. If you need to install MiK $\text{T}_{\text{E}}\text{X}$, you can get the software from <http://www.miktex.org> or from CTAN. The Werner Icking Archive contains excellent [instructions for installing MiK \$\text{T}_{\text{E}}\text{X}\$](#) ¹.

3.1 Where to get MusiX_{TEX}: the Werner Icking Archive

As already mentioned, the home base for all matters related to MusiX_{TEX} is the Werner Icking Music Archive, at <http://icking-music-archive.org>. The most up-to-date versions of MusiX_{TEX} and friends are located in the [software](#) section of the archive. The stable distribution of MusiX_{TEX} will be contained in [musixtex.zip](#), which is intended to serve all operating systems. The CTAN servers are official mirrors. The main CTAN server's web link is <http://www.ctan.org/tex-archive/macros/musixtex/taupin/> and the FTP link is <http://ftp.ctan.org/tex-archive/macros/musixtex/taupin/>.

The files in the Werner Icking Archive have been contributed by such a wide range of authors that there is no common standard for the line endings in text files. But the key text files serve as input to programs such as $\text{T}_{\text{E}}\text{X}$ and METAFONT which are insensitive to this. If you use a PC and run into problems using any of these text files, you might be able to solve them with one of the two routines `dtou.exe` and `utod.exe`, which convert text files from/to the MS-DOS convention to UNIX.

3.2 Installing MusiX_{TEX}

The following subsections provide some very detailed instructions for several different approaches to installing and using the software in either MS Windows or UNIX-like operating systems. You may choose to follow one of these methods as is, or to adapt it to your needs. You may also skip this section entirely and refer to the HOWTO files in the archive for either [Windows](#) or [UNIX](#). Whatever course you choose, there are really just a few fundamental tasks you must accomplish:

- MusiX_{TEX} comprises some $\text{T}_{\text{E}}\text{X}$ source files (`.tex`, `.sty`), font files, and the executable `musixflx`. The $\text{T}_{\text{E}}\text{X}$ sources and font files must be placed in locations where $\text{T}_{\text{E}}\text{X}$ can find them;

¹Thanks to Eva JAKSCH

- \TeX must be advised of the locations of those files;
- If you are installing any postscript fonts, you'll have to add a map file somewhere in the \TeX file structure, and locate and modify a configuration file such as `config.ps` to record the addition of the map file.
- `musixflx` must be placed in a folder that is in your search path, so it can be executed from your working directory;
- To compile a score, you must run in sequence `tex`, `musixflx`, and `tex` again. This will produce a `.dvi` file;
- To view the result, you can either use a DVI viewer, convert the file to postscript with `dvips` and then view it with a postscript viewer such as **GSview** or **ghostscript**, or convert the `.dvi` directly to a PDF file with `pdftex` and then view it with any PDF viewer such as **Adobe Reader**. If you are using type K postscript slurs, note that DVI viewers will not display the slurs.

3.2.1 Installing on a UNIX system

Download the file [musixtex.zip](#) (or perhaps a newer, beta version, which may be named differently) into a temporary directory, for example `/usr/local/src`. Unpack it by saying `unzip musixtex.zip`. This should create a bunch of subdirectories and distribute all the unpacked files among them.

3.2.1.1 Setting up a “private” TEXMF tree

\TeX macros and fonts coming with MusiX \TeX are additions to the standard \TeX distribution. In order to keep the MusiX \TeX files independent of \TeX you may want to create a directory structure separate from that of the base \TeX installation. If for example \TeX has been installed within the directory structure `/usr/share/texmf` then you could create a “private” structure `/usr/local/share/texmf` for storing all \TeX macros and fonts belonging to MusiX \TeX . However, you must tell \TeX where to search for files in `/usr/local/share/texmf`. This is done in the configuration file `texmf.cnf` the location of which you may look up by saying

```
kpsewhich texmf.cnf
```

the output of which will be something like `/usr/share/texmf/web2c/texmf.cnf`.

A “private” TEXMF tree for all users. Below follows an excerpt from a \TeX `texmf.cnf` containing some commented out (`% = comment`) examples of adding “private” search directory structures for \TeX . If you have root privileges you may edit `texmf.cnf` as shown below by defining the environment variable `TEXMFLOCAL` and setting the overall environment variable `TEXMF` to incorporate `TEXMFLOCAL`.

```
% The main tree, which must be mentioned in $TEXMF, below:
```

```
TEXMFMAIN = /usr/share/texmf
```

```
% TEXMFLOCAL = /usr/share/texmf.local
```

```
TEXMFLOCAL = /usr/local/share/texmf
```

```
% If defined, teTeX's texconfig stores modifications here (instead of the
% TEXMFMAIN tree).
```

```
% VARTEXMF = /usr/share/texmf-var
```

```
% User texmf trees can be catered for like this...
%   HOMETEXMF = $HOME/texmf

% Now, list all the texmf trees. If you have multiple trees you can
% use shell brace notation, like this:
%   TEXMF = {$HOMETEXMF,!!$TEXMFLOCAL,!!$TEXMFMAIN}
% The braces are necessary. If you set VARTEXMF, you also have to
% - list $VARTEXMF in the TEXMF definition;
% - make sure that $VARTEXMF precedes $TEXMFMAIN in the TEXMF definition.
TEXMF = {!!$TEXMFLOCAL,!!$TEXMFMAIN}
```

When you have finished editing `texmf.cnf`, say as root `mktexlsr` or `texhash` to update the \TeX file search database to reflect the new `TEXMFLOCAL` directory structure.

A “private” TEXMF tree for single users. On some UNIX machines the global \TeX configuration file `texmf.cnf` may have defined an environment variable `HOMETEXMF` as `$HOME/texmf` and made it part of the environment `TEXMF`. In that case you’ll be able to establish your own “private” `TEXMF` tree by creating a directory `texmf` in your home directory. You should also create the directories `$HOME/texmf/tex` and `$HOME/texmf/fonts` for storing \TeX files and `METAFONT` font definitions respectively.

You may install all files belonging to MusiX \TeX within your “local” `texmf` directory structure. Don’t forget to make these—and other files added to `$HOME/texinf`—known to the \TeX file search database by saying `mktexlsr $HOME/texmf` or `texhash $HOME/texmf`.

If the global \TeX environment `HOMETEXMF` has not been defined you should ask the system administrator to do so. Alternatively you may define your own \TeX environment variables `TEXINPUTS` (search path for \TeX files) and `MFINPUTS` (search path for `METAFONT` font definitions). If you’re using `sh` as your shell interpreter add these lines to your configuration file `$HOME/.profile` (or to `$home/.bash_profile` if you’re using `bash`):

```
TEXINPUTS=".:$HOME/texmf/tex/"
MFINPUTS=".:$HOME/texmf/fonts/"
export TEXINPUTS MFINPUTS
```

If you’re using `csh` or `tcsh` add these lines to your configuration file `$HOME/.login`:

```
setenv TEXINPUTS ".:$HOME/texmf/tex/"
setenv MFINPUTS ".:$HOME/texmf/fonts/"
```

The two slashes after the path names cause \TeX to search all directories under `$HOME/texmf`.

3.2.1.2 Musix \TeX macros

Start installing files by unpacking the latest version of the MusiX \TeX distribution, for example `musixtex.tar.gz`, into a temporary directory, for example `/usr/local/src`. To perform the unpacking say `tar -zxf musixtex.tar.gz`. The result will be a new directory `/usr/local/src/musixtex`.

In your “private” `TEXMF` tree create a directory `tex/musixtex`. Copy all files from the subdirectory `tex` of the MusiX \TeX distribution to `tex/musixtex`.

3.2.1.3 Musix \TeX bitmapped fonts

From the subdirectory `mf` of the MusiX \TeX distribution copy all files to `/usr/share/texmf/fonts/source/public/musixtex/`. Use the command

```
kpsewhich musix20.tfm
```

to look up the directory storing the MusiX \TeX font metric files. The output will be something like `/usr/share/texmf/fonts/tfm/public/musixtex/musix20.tfm`. Then either delete all files from the directory `/usr/share/texmf/fonts/tfm/public/musixtex/` or copy all files from the directory `tfm` of the MusiX \TeX distribution.

If you just delete the files from `/usr/share/texmf/fonts/tfm/public/musixtex/` then they will be regenerated by \TeX when you start processing your MusiX \TeX source files.

3.2.1.4 Musix \TeX Type 1 Postscript Fonts for UNIX²

You should consider also installing the postscript type 1 versions of the MusiX \TeX fonts which allow you to generate high quality PDF output from your MusiX \TeX sources. Begin by downloading and unpacking the font distribution musixps-unix.tar.gz. Use the same method as above for the unpacking. Then follow these instructions:

1. Make a directory such as `$TEXMF/fonts/type1/musixtex`, for PFB fonts.
2. Copy all PFBs to the above directory; they should be readable by all users.
3. Copy the map file `./dvips/musix.map` to the appropriate directory, such as `$TEXMF/dvips/config`.
4. Add a line

```
p +musix.map
```

 to a suitable `config.xxx` file or, in a `tetex`-based distribution, add `musix.map` to the list of map files for Type1 fonts with METAFONT equivalents in the `updmap` script, which should then be executed by doing `./updmap`.
5. Copy the map file `./dvipdfm/musix.map` to the appropriate directory, such as `$TEXMF/dvipdfm/config`
6. Add the following line to the `config` file for `dvipdfm`:

```
f musix.map
```
7. Do `mktexlsr` or `texhash` or whatever is necessary on your system to re-generate the \TeX database.

3.2.1.5 Updating existing MusiX \TeX software in unix

Start by downloading and unpacking the latest version of the MusiX \TeX distribution, for example musixtex-T112.tar.gz into a temporary directory, for example `/usr/local/src`. To perform the unpacking say `tar -zxf musixtex-T112.tar.gz`. The result will be a new directory `/usr/local/src/musixtex-T112`.

You'll need to know the locations of the MusiX \TeX macros and METAFONT font sources. Use the command `kpsewhich musixtex.tex` to look up the directory storing the MusiX \TeX macros. The output will be something like `/usr/share/texmf/tex/generic/musixtex/musixtex.tex`.

Use the command `kpsewhich musix20.mf` to look up the directory storing the MusiX \TeX METAFONT fonts sources. The output will be something like `/usr/share/texmf/fonts/source/public/musixtex/musix20.mf`

In the following text, if necessary replace `/usr/share/texmf/tex/generic/musixtex/` and `/usr/share/texmf/fonts/source/public/musixtex/` with the actual paths to `musixtex.tex` and `musix20.mf` resp.

²Thanks to Takanori UCHIYAMA

MusiX_{TEX} macros. From the subdirectory `tex` of the MusiX_{TEX} distribution copy all files to `/usr/share/texmf/tex/generic/musixtex/`.

MusiX_{TEX} bitmapped fonts. From the subdirectory `mf` of the MusiX_{TEX} distribution copy all files to `/usr/share/texmf/fonts/source/public/musixtex/`.

Use the command `kpsewhich musix20.tfmto` look up the directory storing the MusiX_{TEX} font metric files. The output will be something like `/usr/share/texmf/fonts/tfm/public/musixtex/musix20.tfm`. Then either delete all files from the directory `/usr/share/texmf/fonts/tfm/public/musixtex/` or copy all files from the directory `tfm` of the MusiX_{TEX} distribution.

If you just delete the files from `/usr/share/texmf/fonts/tfm/public/musixtex/` then they will be regenerated by TeX when you start processing your MusiX_{TEX} source files.

MusiX_{TEX} type 1 fonts. You should consider also installing Takanori Uchiyama's type 1 versions of the MusiX_{TEX} fonts which allow you to generate high quality pdf output from your MusiX_{TEX} sources. Begin by downloading and unpacking the font distribution [musixps-unix.tar.gz](#). Then follow the instructions in the section "3. INSTALLATION" of the accompanying file `README`

Updating the TeX search path. In order to tell TeX where to look for the MusiX_{TEX} files update the TeX file search database by saying as root `mktextlsr` or `texhash`.

3.2.2 Installing on a MS Windows system with MiK_{TEX}

Assuming, you have installed MiK_{TEX}³, a text editor, **ghostscript**, and **GSview**, you are ready to install MusiX_{TEX}.

Two methods will be described below. Both will accommodate the preprocessors **PMX** and **M-Tx**.

1. Gathering all the binaries in a separate directory, i.e. `c:\texbin`.
2. Not touching the installation of MiK_{TEX}, Musixtex and the preprocessors, but creating PATHs to find the binaries and the (new) fonts.

You may employ either installation method as is, or modify it as required to suit your own installation.

3.2.2.1 1st method: gathering all executables⁴

This method is clearly explained in detail in [Installing MiK_{TEX} and MusiX_{TEX} on Windows 2000](#).

First you have to make a local directory for the binaries, e.g. `c:\texbin`. There are 3 ways to put this directory in the search path:

1. Add a line `set path=%path%;c:\texbin` to the file `autoexec.bat`; or
2. In the Control Panel, open the System Properties dialogue. Select **Advanced** and click on **environment variables**. Edit the PATH, adding `c:\texbin`; or
3. (for a local user only) In **environment variables** click on **new**, enter the word "path" in the box **Variable** and `c:\texbin` in the box **Variable Value**.

Then, proceed as follows:

1. Extract [musixtex.zip](#) in a temporary directory, i.e. `c:\musixtemp`, preserving the directory structure. You should get subdirectories `doc`, `dvipdfm`, `dvips`, `fonts`, `system`, and `tex`.

³The methods are based on MiK_{TEX} version 2.4

⁴Thanks to Eva JAKSCH

2. In your local MiK \TeX file structure, create a subdirectory `c:\localtexmf\tex\generic\musixtex` . Move all `*.tex` and `*.sty` files from `c:\musixtemp\tex` into this directory.
3. Move all `*.exe` files from `c:\musixtemp\bin` into `c:\texbin` .
4. Create `c:\localtexmf\fonts\source\public\musixtex` . Move all `*.mf` files from `c:\musixtemp\font` into this directory.
5. Create `c:\localtexmf\fonts\tfm\public\musixtex` . Move all `*.tfm` files from `c:\musixtemp\tfm` into this directory.
6. Create `c:\localtexmf\dvips\base` if necessary. Move `psslurs.pro` from `c:\musixtemp\dvips` into this directory.
7. Create `c:\localtexmf\doc` . Move everything from `c:\musixtemp\doc` into this directory.
8. Refresh the MiK \TeX file name fatabase.

To install **PMX** and **M-Tx**, copy their `*.exe` and `*.tex` files to the appropriate subdirectories and refresh the file name database as described above.

3.2.2.2 2nd method: a directory for MusiX \TeX separate from MiK \TeX

Here you create your own directory tree for MusiX \TeX and set the paths to it. The only thing that must be adapted in MiK \TeX is `miktex.ini`. Later, to update to new versions of MusiX \TeX and friends, simply copy the new versions over the old ones.

1. Create `c:\musixtex` where you plan to put all your music and program files. Extract [musixtex.zip](#) into this directory, preserving the subdirectory structure. Create additional subdirectories `pmx` and `mtx` for the preprocessors if applicable, and another subdirectory `mymusix` for your own input files . Now you should have the subdirectories `doc`, `dvipdfm`, `dvips`, `fonts`, `system`, `tex`, `pmx`, `mtx`, and `mymusix` in `c:\musixtex` .
2. Set the paths for the executables:

In `autoexec.bat` (or right click on My Computer, select properties, then advanced, then environment variables), edit the `PATH` so that it begins
`c:\musixtex\bin;c:\musixtex\pmx\bin;c:\musixtex\mtx\bin;c:\musixtex\mymusix\bin;...`

3. Set the `PATHs` for the source files, input files, and fonts:

First copy

`C:\texmf\miktex\config\miktex.ini` to `C:\localtexmf\miktex\config\miktex.ini`

If necessary create the (sub)directories.

Insert the paths to the `musixtex` tree in the new `miktex.ini` . The changed lines will look like⁵ :

```
[Dvips]
;; where Dvips searches for font mapping files (*.map)
MAPPath=.;C:\musixtex\dvips;%R\dvips//;%R\fonts\map//
;; where Dvips searches for all sorts of PS related files
;; (*.afm;*.pfb;*.pro)
PSPath=.;C:\musixtex\dvips;C:\musixtex\fonts//;%R\dvips//
; %R\fonts\type1//
```

⁵The lines are taken from `miktex.ini`, MiK \TeX version 2.4. It may slightly differ from other versions, i.e. eLa \TeX is no more needed in version 2.4: La \TeX also uses `etex`, the extended \TeX .

```

[eLaTeX]
;; where eLaTeX searches for input files
Input Dirs=.;C:\musixtex//;%R\etex\latex//;%R\etex\generic//;%R\etex//
           ;%R\tex\latex//;%R\tex\generic//;%R\tex//

[eTeX]
;; where eTeX searches for input files
Input Dirs=.;c:\musixtex//;%R\etex\plain//;%R\etex\generic//;%R\etex//
           ;%R\tex\plain//;%R\tex\generic//;%R\tex//

[LaTeX]
;; where LaTeX searches for input files
Input Dirs=.;C:\musixtex//;%R\etex\latex//;%R\etex\generic//;%R\etex//
           ;%R\tex\latex//;%R\tex\generic//;%R\tex//

[METAFONT]
;; where METAFONT searches for input files
Input Dirs=.;C:\musixtex\fonts\mf;%R\metafont//;%R\fonts\source//

[MiKTeX]
;; Search paths
;; used to locate MiKTeX executables (*.bat;*.com;*.exe)
EXEPath=%R\miktex\bin;C:\musixtex\mymusix\bin
;; used by MfFntMap to locate font mapping files (*.map)
MAPPath=.;%R\fonts\map//;%R\miktex\config//;%R\dvips//;%R\pdftex//
        ;%R\dvipdfm//;C:\musixtex\dvipdfm;C:\musixtex\dvips
;; used to locate all sorts of PostScript related files
PSPPath=.;%R\miktex\config//;%R\dvips//;%R\pdftex//;%R\dvipdfm//
        ;%R\fonts\type1//;C:\musixtex//
;; used to locate TeX font metric files (*.tfm)
TFMPath=.;C:\musixtex\fonts\tfm;%R\fonts\tfm//

[TeX]
;; where TeX searches for input files
Input Dirs=.;C:\musixtex//;%R\tex\plain//;%R\tex\generic//;%R\tex//

```

There may be older versions of some of the fonts in `\texmf`, but by placing the new paths at the start of the line, the new input files and fonts will be used. The “//” extends the search path to the subdirectories. Also the path

```
C:\musixtex\fonts\type1//;C:\musixtex\dvips//;C:\musixtex\dvipdfm//
```

is faster than `C:\musixtex//` because fewer directories will be searched.

In `C:\musixtex\mymusix\bin` you may keep your own executables and/or batch files.

Don't forget to refresh the the MiK_{TEX} File Name Database.

3.2.2.3 Installing Musix_{TEX} type 1 postscript fonts for Mik_{TEX}

Next, adapt the config files by copying them first to `c:\localtexmf` and then adding a line (if you like, you may put this lines in a batch file⁶:

```
SET TEXMF=c:\texmf
```

⁶In fact, this commands are taken from Christof Biebrichers batch file


```

SET LOCALTEXMF=c:\localtexmf
REM make backup copies of the config files and add the required map-files
copy %LOCALTEXMF%\dvips\config\config.ps %LOCALTEXMF%\dvips\config\configps.old
copy %TEXMF%\dvips\config\config.ps %LOCALTEXMF%\dvips\config\config.ps
copy %LOCALTEXMF%\dvips\tetex\config.pdf %LOCALTEXMF%\dvips\tetex\configpdf.old
copy %TEXMF%\dvips\tetex\config.pdf %LOCALTEXMF%\dvips\tetex\config.pdf
copy %LOCALTEXMF%\dvipdfm\config\config %LOCALTEXMF%\dvipdfm\config\config.old
copy %TEXMF%\dvipdfm\config\config %LOCALTEXMF%\dvipdfm\config\config
REM add the lines
echo p +musix.map >> %LOCALTEXMF%\dvips\config\config.ps
echo p +musix.map >> %LOCALTEXMF%\dvips\tetex\config.pdf
echo f musix.map >> %LOCALTEXMF%\dvipdfm\config\config

```

Don't forget to refresh the the MiK \TeX File Name Database.

3.3 Running Musix \TeX in a MS Windows system

In most systems the commands run in a shell. Unix people are familiar with it. In MS Windows systems, as a last refuge, you always can use the DOS cmd shell (start → run → type cmd → return).

In the examples, a file `mymp` (**my masterpiece**) is used, for demonstrating the commands for running musix \TeX and the preprocessors.

You also may run the commands without a shell. To forward the argument in the batch file, use an other batch file which stays in the same directory as your masterpiece `mymp.tex` and has the same name: `mymp.bat`. `mymp.bat` consists of 1 line: `musixtex.bat %~n0`.

Choose one batch file

making DVI	making postscript	running MPX	running MTX
<code>rem mymp.bat</code>	<code>rem mymp.bat</code>	<code>rem mymp.bat</code>	<code>rem mymp.bat</code>
<code>musixtex.bat %~n0</code>	<code>mupsgs.bat %~n0</code>	<code>pmxpsgs.bat %~n0</code>	<code>mtxpsgs.bat %~n0</code>

Thus `mymp.bat` passes his own name (`mymp`) via `%~n0` to the `musixtex.bat` (or other) command.

If you want to process i.e. all `pmx` files in a directory, then you could make a batch file `pmxall.bat` with the following commands:

```

rem pmxall
for /F %I in ('dir /B *.pmx') do call pmxpsgs.bat %I

```

Of course, this command also works for only one file in a directory, which means you may use `pmxall.bat` without giving a file name.

3.3.1 Batch commands for making DVI and postscript

The `musixtex.bat` command fulfills the sequence $\TeX \Rightarrow \text{musixflx} \Rightarrow \mathcal{D}\mathcal{V}\mathcal{I}$. It produces a DVI file that can be displayed by a viewer (normally a part of the \TeX installation).

The command must reside in a `bin` directory, with appropriate paths installed, see the installation procedures.

```

rem musixtex.bat
if not exist %1.tex goto end
if exist %1.mx1 del %1.mx1
if exist %1.mx2 del %1.mx2
tex %1
if errorlevel 2 goto end

```

```

musixflx %1
if errorlevel 1 goto end
tex %1
:end

```

The next command (named `mupsgs.bat`) processes `mymp.tex` by means of: `mymp.bat` (in the same directory) which consists of the line: `mupsgs.bat %~n0 .`

```

rem mupsgs.bat
call musixtex.bat %1
DVIPS.EXE -e0 -t a4 -P pdf -GO %1.dvi
gsview32.exe %1.ps

```

The `dvips` output looks a little bit nicer using the options `-e0` (character drift value = 0) and `-GO` (shift low chars to higher position). `-t a4` is for a4 paper (`-t letter` for letter format). If the output is wanted in Adobe Acrobat pdf format, the option `-P pdf` (load `config.pdf`) is needed.

The next command (named `pmxpsgs.bat`) processes `mymp.pmx` by means of: `mymp.bat` (in the same directory) which consists of the line: `pmxpsgs.bat %~n0 .`

```

rem pmxpsgs.bat
echo %1 | pmxab
mupsgs.bat %1

```

And finally the command (named `mtxpsgs.bat`) processes `mymp.mtx` by means of: `mymp.bat` (in the same directory) which consists of the line: `mtxpsgs.bat %~n0 .`

```

rem mtxpsgs.bat
prepmx -v %1
pmxpsgs.bat %1

```

Chapter 4

MusiX_{TEX} examples

The file `musixdoc.tex`, the source for this manual, contains many useful examples. In the PDF, many examples are accompanied by a display of the code that produced them, while for a few only an image of the extract is included in the PDF and you'll have to look in `musixdoc.tex` to see the coding.

Other useful examples cannot be embedded in the source, either because they are meant to be in $\text{T}_{\text{E}}\text{X}$, not $\text{L}\text{a}\text{T}_{\text{E}}\text{X}$, or because they are simply too large. For these the source files are provided separately.

When compiling or viewing any of the examples, you should keep in mind that most DVI previewers and laser printers have their origin one inch below and one inch to the right of the upper right corner of the paper, while the musical examples have their upper left corner just one centimeter to the right and below the top left corner of the page. Therefore, special parameters may have to be given to the DVI transcription programs unless special `\hoffset` and `\voffset` $\text{T}_{\text{E}}\text{X}$ commands have been included within the $\text{T}_{\text{E}}\text{X}$ source.

4.1 Small examples

- `ossiaexa.tex`: This is a stand-alone example of the use of `ossia`, provided by Olivier Vogel (section 2.18.3 on page 69).
- `8bitchar.tex`: Using 8bit characters. Here, the European character set is used. See section 7 on page 78 for other sets.

4.2 Full examples

All of these examples and many others are available in [musixexa.zip](#) or [musixexa-T112.tar.gz](#). Some of them require the presence of `musixcpt.tex` which makes examples created in Music $\text{T}_{\text{E}}\text{X}$ compatible with MusiX $\text{T}_{\text{E}}\text{X}$. Here we only mention a few of special interest.

4.2.1 Examples mentioned in the manual

- `avemaria.tex`: the “Méditation” (alias “Ave Maria”) by Charles GOUNOD for organ and violin or voice. To run this five-page example you'll also need `avemariax.tex`. It demonstrates the use of separated bar rules (section 2.11.2) and the use of staves of different sizes (section 2.19). Also, an additional instrument is created for lyrics. This was a common practice before the `musixlyr` package was created by Rainer Dunker.

- `glorias.tex` : a local melody for the French version of *Gloria in excelsis Deo*, a three-page piece demonstrating the use of the `hardlyrics` commands (section 2.21.1.2). `gloriab.tex` is the same piece, but with organ accompaniment.

4.2.2 Other examples, provided by the authors of MusiX_TE_X

- `traeumer.tex` : the famous “Träumerei” by Robert SCHUMANN for piano, in genuine MusiX_TE_X but with some additions to perform ascending bitmapped *crescendos*. There are also S-shaped slurs between 2 staves.
- `parnasum.tex` : the first page of “Doctor gradus ad Parnassum” by Claude DEBUSSY for piano. It contains a rather complex example of a new command `\Special` to create staff-jumping doubly beamed notes.

4.3 Compiling `musixdoc.tex`

There are many other files that are required to compile this document but all of them are included in `musixtex.zip`, and will be placed in the proper locations if you have followed the installation instructions for MusiX_TE_X. The commands that one of us (Don Simons) uses to compile it—using MiK_TE_X 2.4—and produce `musixdoc.ps` are as follows:

```

elateX musixdoc
musixflx musixdoc
elateX musixdoc
makeindex -l musixdoc
elateX musixdoc
(may require several more runs of elateX to get rid of
  "labels may have changed ..." warnings)
dvips -ta4 musixdoc

```

To produce `musixdoc.pdf`, open `musixdoc.ps` in **GSview**. Go to `File|Convert`, select `pdfwrite` at 600 dpi resolution, click OK.

Chapter 5

Pitches

'A 'B 'C 'D 'E 'F 'G A B C D E F G H I J K L M N a b c d e

Notes for letters A-Z in bass and treble clefs.

Notes, Accidentals, Accents, Clefs and Rests

`\longa` `\wq` `\wqq` `\wh` `\hu` `\hl` `\qu` `\ql` `\cu` `\cl` `\ccu` `\ccl` `\cccu` `\cccl` `\cccu` `\cccc` `\grcu` `\grcl`
`\maxima` `\breve`

Accidentals: `>` `\cdsh` `~` `\csh` `=` `\cna` `-` `\cfl` `<` `\cdf1`

`\dqu123` `\yqu123` `\dcqu2` `\dhqu2` `\doqu2` `\xqu2` `\oxqu2` `\roqu2` `\tgqu2` `\kqu2` `\squ3` `\lsqu3` `\rsqu3` `\cqu4` `\cql4` `\chu4` `\chl4`

1 musixdia.tex 2 musixper.tex 3 musixgre.tex 4 musixlit.tex 5 musixext.tex

`\lpz` `\upz` `\lsf` `\usf` `\lst` `\ust` `\lppz` `\uppz` `\lsfz` `\usfz` `\lpzst` `\upzst` `\downbow` `\flageolett`
`\upbow` `\whp` `\qupp`

Accent on beam with prefix `b` and beam reference number instead of the pitch

`\trebleclef` `\bassclef` `\smallaltoclef` `\drumclef2` `\gregorianFclef3`
`\altoclef` `\smalltrebleclef` `\smallbassclef` `\gregorianCclef3` `\oldGclef4`

`\qqs` `\hs` `\qs` `\ds` `\qp` `\hpause` `\hpausep` `\pause` `\pausep` `\PAuse` `\PAUSE` `\Hpause4`
`\liftpause` `\liftpause`

Other Symbols

`\Trille` `\trille` `\shake` `\Shake` `\mordent` `\Mordent` `\turn` `\backtu` `\Shakel` `\Shakes` `\Shakene` `\Shakew`
`tr` `~` `~` `~` `~` `~` `~` `~` `~` `~`

`\allabreve` `\meterC` `\reverseC` `\reverseallabreve` `\meterplus` `\duevolte` `\l[r]par`

`\setvoltabox` `\setvolta` `\coda` `\Coda` `\segno` `\Segno` `\caesura` `\cbreath` `\PED` `\sPED` `\DEP` `\sDEP`

`\metron` 1. 2. `0` `□` `§` `∞` `,`

`Red.` `♯` `*` `*`

`\fermataup` `\arpeggio d5` `\uptrio` `\octfinup` `\slide5` `\boxit A` `\circleit B`
`\Fermataup` `\bracket`

`\fermatadown` `\downtrio` `\octfindown` `\leftrepeat` `\rightrepeat`
`\fermatadown` `8a bassa` `\leftrepeat` `\rightrepeat`

Index

- `%`, 8
- `&`, 17
- `*`, 20, 28
- `&`, 2
- `|`, 2, 17

- `\AB`, 96
- `\absoluteaccid`, 31
- `\accshift`, 30
- acute accent, 30
- `\addspace`, 28
- `\afterruleskip`, 16, 17, 28
- `\akkoladen`, 14
- `\alaligne`, 6, 7, 42, 48
- `\alapage`, 6, 7, 42, 48
- `\allabreve`, 13
- `\allbarrules`, 45
- `\altitude`, 73
- `\altplancher`, 73
- `\altportee`, 73
- apostropha, 86
- `\apostropha`, 86
- apostropha aucta, 86
- `\apostropha aucta`, 86
- `\appendlyrics`, 75
- `\arithmeticsskip`, 16
- `\arpeggio`, 62
- `\assignlyrics`, 75
- `\assignlyricshere`, 75
- `\assignlyricsmulti`, 75
- `\atnextbar`, 27
- `\atnextline`, 67
- `\AUF`, 96
- `\author`, 97
- `\autoledgerlines`, 84
- `\auxlyr`, 75
- `\auxsetsongraise`, 76

- `\backturn`, 64
- `\bar`, 42, 48, 55
- `\barno`, 46
- `\barnumbers`, 46
- `\basslowoct`, 52
- `\bassoct`, 52
- beam, 12
- beams, 3
- `\beforeruleskip`, 16, 17, 28

- `\begin{music}`, 80
- `\beginmel`, 75
- `\bf`, 57, 96
- `\bigaccid`, 29
- `\bigfl`, 29
- `\biglbrace`, 82
- `\bigrbrace`, 82
- `\bigsh`, 29
- `\BIGtype`, 57
- `\BIGtype`, 57
- `\Bigtype`, 57
- `\bigtype`, 57
- `\bistropa`, 87
- `\bivirga`, 86
- `\blppz`, 61
- `\blpz`, 61
- `\blsf`, 61
- `\blsfz`, 61
- `\blst`, 61
- `\bltext`, 61
- `\boxit`, 47, 60
- `\boxitsep`, 60
- `\bporrectus`, 88
- `\braceheight`, 82
- `\bracket`, 66
- BRAHMS, J., 25
- `\breakslur`, 36, 42
- breath, 64
- `\breve`, 18
- `\bsk`, 28
- `\bslur`, 38
- `\buppz`, 61
- `\bupz`, 61
- `\busf`, 61
- `\busfz`, 61
- `\bust`, 61
- `\butext`, 61
- `\bye`, 15

- `\ca`, 18
- cadenzas, 68
- `\caesura`, 64
- `\catcode`, 11, 81
- `\catcodesmusic`, 81
- cautionary accidental, 30
- `\cbreath`, 64
- `\cca`, 18
- `\ccca`, 18

- `\ccccca`, 81
- `\cccccl`, 81
- `\cccccu`, 81
- `\cccccl`, 18
- `\cccclp`, 96
- `\cccclpp`, 96
- `\ccuccu`, 18
- `\ccuccup`, 96
- `\ccuccpp`, 96
- `\cccl`, 18
- `\ccclp`, 96
- `\ccclpp`, 96
- `\cccu`, 18
- `\cccup`, 96
- `\cccupp`, 96
- `\cchar`, 59, 74
- `\ccharnote`, 59, 74
- `\ccl`, 18
- `\cclp`, 96
- `\cclpp`, 96
- `\ccu`, 18
- `\ccup`, 96
- `\ccupp`, 96
- `\cdf`, 30
- `\cdsh`, 30
- `\centerbar`, 27
- `\centerhpause`, 27
- `\centerPAUSE`, 27
- `\centerPAuse`, 27
- `\centerpause`, 27
- `\cfl`, 30
- `\changeClefs`, 51
- `\changecontext`, 28, 54, 55
- `\changesignature`, 50
- `\chl`, 93
- chord, 12
- `\ChoirStrut`, 82
- `\chu`, 93
- `\circleit`, 60
- `\cl`, 18
- clefs (empty), 53
- `\climacus`, 89
- `\climacusauctdown`, 90
- `\climacusdeminut`, 91
- `\climacusresupinus`, 89
- `\clivis`, 87
- `\clivisauctdown`, 89

`\clivisauctup`, 89
`\clivisdeminut`, 91
`\clp`, 20
`\clpp`, 20
`\clppp`, 97
`\cmidstaff`, 59
`\cna`, 30
`\cnhl`, 93
`\cnhu`, 93
`\cnql`, 93
`\cnqu`, 93
`\Coda`, 56
`\coda`, 56
`\conbarrule`, 44
`\condashbarrule`, 44
`\condashmultibarrule`, 45
`\condotbarrule`, 44
`\condotmultibarrule`, 45
`\conmultibarrule`, 45
`\Contpiece`, 48
`\contpiece`, 48, 97
`\copylyrics`, 75
`\cql`, 93
`\cqu`, 93
`\crescendo`, 65
`crescendos`, 108
`\csh`, 30
`\csong`, 74
`\cu`, 18
`\cup`, 20
`\cupp`, 20
`\cuppp`, 97
`curly.tex`, 14
`\curlybrackets`, 14
`\curve`, 36

`\dateUSenglish`, 82
`\dccccl`, 83
`\dccccu`, 83
`\dcccl`, 83, 93
`\dcccc`, 83, 93
`\dccl`, 83, 93
`\dccu`, 83, 93
`\dcl`, 83
`\dclp`, 83
`\dclpp`, 83
`\dcqb`, 93
`\dcql`, 93
`\dcqu`, 93
`\dcu`, 83
`\dcup`, 83
`\dcupp`, 83

`\dczq`, 93
`DEBUSSY, C.`, 108
`\decrescendo`, 65
`\DEP`, 63
`\Dep`, 63
`\df1`, 29
`\dgroff`, 86
`\dhccl`, 94
`\dhccu`, 94
`\dhcl`, 94
`\dhcu`, 94
`\dhqb`, 94
`\dhql`, 94
`\dhqu`, 94
`\dhsong`, 75
`\dhzq`, 94
`\diapunc`, 86
`\diapunctumauctdown`, 86
`\dnq`, 83
`\doccl`, 94
`\doccu`, 94
`\docl`, 94
`\docu`, 94
`\documentstyle`, 79
`\doqb`, 94
`\doql`, 94
`\doqu`, 94
`\Dotted`, 41
`\dotted`, 35, 41
`\doublebar`, 42
`\doublethumb`, 66
`\downbow`, 96
`\downtrio`, 67
`\dozq`, 94
`\dqb`, 83
`\Dqbb1`, 21
`\Dqbbu`, 21
`\Dqbl`, 21
`\Dqbu`, 21
`\dql`, 83
`\dqlp`, 83
`\dqlpp`, 83
`\dqu`, 83
`\dqup`, 83
`\dqupp`, 83
`\Drtx`, 82
`drum clef`, 94
`\drumclef`, 94
`\ds`, 27, 84
`\dsh`, 29
`\Dtx`, 82

`\duevolte`, 57
`dvidvi`, 42
`\Dwr`, 84
`\dznq`, 83
`\dzq`, 83
`\dzqp`, 83
`\dzqpp`, 83

`\eeeer`, 84
`\eeer`, 84
`\eer`, 84
`\eightpoint`, 96
`\elemskip`, 5, 15–17, 28, 49
`\en`, 2
`\enableauxlyrics`, 76
`\end`, 15
`\end{music}`, 80
`\endcatcodesmusic`, 81
`\endextract`, 80
`\endmel`, 75
`\endmuflex`, 15
`\Endpiece`, 6, 48
`\endpiece`, 6, 48
`\endvolta`, 55
`\endvoltagebox`, 55
`\enotes`, 2
`\er`, 84
`\everystaff`, 67
`\extractline`, 80

`\f`, 64
`\Fermatadown`, 64
`\fermatadown`, 64
`\Fermataup`, 64
`\fermataup`, 64
`\ff`, 64
`\fff`, 64
`\ffff`, 64
`\fl`, 29
`\flageolett`, 60
`flats`, 12
`\folio`, 49
`\footline`, 49
`\Footnote`, 97
`footnote`, 48
`\footnote`, 97
`footnotes`, 97
`\forcelyrhyphensfalse`,
76
`\forcelyrhyphenstrue`, 76
`forzato`, 60
`\fp`, 64

`\FR`, 96
`french.sty`, 81
`\freqbarno`, 46
`\frtpoint`, 96
`\fullauthor`, 97
`\fulltitle`, 97

`\GB`, 96
`\Gb`, 96
`\gbarre`, 92
`\gdot`, 92
`\generalmeter`, 12, 54
`\generalsignature`, 8, 12, 48, 50
`\geometricsskipsscale`, 15
`\golyr`, 75
GOUNOD, C., 71, 107
grave accent, 30
`\grcl`, 69
`\grcu`, 69
`\gregorianCclef`, 85
gregorian chant, 84
`\gregorianFclef`, 86
GRIEG, E., 25
`\groff`, 86
`\groupbottom`, 13
`\grouptop`, 13
`\guitar`, 92
guitar tablatures, 91

`\ha`, 18
`\halfties`, 42
`\hardlyrics`, 75
`\hardspace`, 28
hard width, 4
harmonic notes, 83
HAYDN, J., 68
`\hb`, 24
`\headline`, 49
`\hf`, 82
`\Hidebarrule`, 43
`\hidebarrule`, 43, 44
`\hl`, 15, 18, 19
`\hloff`, 29
`\Hlonga`, 93
`\hlp`, 20
`\hlpp`, 20
`\hlppp`, 97
`\hoffset`, 48, 96, 107
`\Hpause`, 93
`\hpause`, 27, 84
`\hpausep`, 27, 84

`\hqsk`, 28
`\hr`, 84
`\hroff`, 29
`\hrp`, 84
`\hs`, 27, 84
`\hsize`, 48, 96
`\hsk`, 25, 28
`\hu`, 18
`\hup`, 20
`\hupp`, 20
`\huppp`, 97

`\Ibbbbbbbl`, 81
`\ibbbbbbbbl`, 81
`\Ibbbbbbbu`, 81
`\ibbbbbbbbu`, 81
`\Ibbbbbl`, 81
`\ibbbbbbl`, 81
`\Ibbbbbu`, 81
`\ibbbbbbu`, 81
`\Ibbbbbl`, 21
`\ibbbbbbl`, 21
`\Ibbbbbu`, 21
`\ibbbbbbu`, 21
`\Ibbbl`, 21
`\ibbbbl`, 21
`\Ibbbu`, 21
`\ibbbbu`, 21
`\Ibbl`, 21
`\ibbl`, 21
`\Ibbu`, 21
`\ibbu`, 21
`\Ibl`, 21
`\ibl`, 21
`\iBslurd`, 41
`\ibslurd`, 34
`\iBsluru`, 41
`\ibsluru`, 34
`\Ibu`, 21
`\ibu`, 21
`\Icresc`, 66
`\icresc`, 65
`\Idecresc`, 66
`\idecresc`, 65
`\ignorenats`, 50
`\ilcresc`, 65
`\ildecresc`, 65
`\ilslurd`, 39
`\ilsluru`, 39
`\includegraphics`, 81
`\indivbarrules`, 44
`\input`, 81

instrument name, 13
`\instrumentnumber`, 12
`\interbarrules`, 93
`\interbeam`, 66, 72, 73
`\interfacteur`, 72
`\interinstrument`, 72, 74, 77
`\Interligne`, 27, 72, 73
`\Internote`, 72
`\internote`, 32, 72, 92
`\interportee`, 72
`\interportee+\interinstrument`, 72
`\interstaff`, 72, 73
`\invertslur`, 37
`\Ioctfindown`, 32
`\ioctfindown`, 32
`\Ioctfinup`, 32
`\ioctfinup`, 32
`\ircresc`, 65
`\irdecresc`, 65
`\irslurd`, 39
`\irsluru`, 39
`\iSlur`, 40
`\islurd`, 33, 34
`\Islurdbreak`, 37
`\isluru`, 33, 34
`\Islurubreak`, 37
`\isslurd`, 34
`\issluru`, 34
`\it`, 57, 96
`\itenl`, 34
`\itenu`, 34
`\itied`, 34
`\itieu`, 34
`\ITrille`, 62
`\Itrille`, 62

`\kccl`, 94
`\kccu`, 94
`\kcl`, 94
`\kcu`, 94
key signatures, 12
`\kqb`, 94
`\kql`, 94
`\kqu`, 94
`\kzq`, 94

`\Largemusicssize`, 12, 73
`\largemusicssize`, 12, 73
`\Largevalue`, 70
`\largevalue`, 70

`\larpeggio`, 62
`LATEX`, 97
`\lchar`, 59, 74
`\lcharnote`, 59, 74
`\lcl`, 19
`\lclimacus`, 89
`\lclivis`, 87
`\lclyr`, 76
`\lcu`, 19
`\ldfl`, 29
`\ldsh`, 29
 ledger lines, 84
`\leftlyrfalse`, 76
`\leftlyrtrue`, 76
`\leftrepeat`, 55, 56
`\leftrightrepeat`, 55
`\let\extractline\leftline`,
 80
`\fl`, 29
`\lh`, 19
`\lhl`, 19
`\lhu`, 19
`\liftcresc`, 66
`\lifthpause`, 27, 84
`\lifthpausep`, 27
`\lifthr`, 84
`\Liftoctline`, 32
`\liftpause`, 27, 84
`\liftpausep`, 27
`\Liftslur`, 36, 42
`\liftslur`, 42
`\liftwr`, 84
`\linegoal`, 7, 49
 liquescens neumes, 89
`\llabel`, 75
`\llyr`, 76
`\lmidstaff`, 59
`\lna`, 29
`\loff`, 25, 29
`\loffset`, 29
`\longa`, 18
`\longaa`, 18
`\longledgerlines`, 84
`\lowlyrlink`, 76
`\lpar`, 66
`\lpodatus`, 87
`\lpppt`, 97
`\lppt`, 20
`\lppz`, 60
`\lpt`, 20
`\lpz`, 60
`\lpzst`, 60
`\lq`, 19
`\lql`, 19
`\lqu`, 19
`\lsf`, 60
`\lsfz`, 60
`\lsh`, 29
`\lsong`, 74
`\lsqu`, 86
`\lst`, 60
`\lw`, 19
`\lyr`, 75
`\lyrhyphenchar`, 76
`\lyric`, 75
`\lyric*`, 75
`\lyrich`, 75
`\lyrich*`, 75
 lyrics, 74
`\lyricsoff`, 75
`\lyricson`, 75
`\lyrlayout`, 76
`\lyrlink`, 76
`\lyrmodealter`, 76
`\lyrmodealterhere`, 76
`\lyrmodealtermulti`, 76
`\lyrmodenormal`, 76
`\lyrmodenormalhere`, 76
`\lyrmodenormalmulti`, 76
`\lyrnop`, 76
`\lyroffset`, 76
`\lyrpt`, 75
`\lyrraise`, 76
`\lyrraisehere`, 76
`\lyrraisemulti`, 76
`\lyrrule`, 75
`\lyrruleend`, 75
`\lyrstrutbox`, 76
`\maketitle`, 97
`\medtype`, 57
 meter, 12, 54
`\meterC`, 13
`\meterfrac`, 13
`\meterplus`, 13
`\meterskip`, 13
`\metron`, 60
`\metronequiv`, 60
`\mf`, 64
`\MI`, 96
`\Mi`, 96
`\midslur`, 35, 40
`\minlyrrulelength`, 76
`\minlyrspace`, 76
`\minmulthyphens`, 76
`\Mordent`, 64
`\mordent`, 64
 MOZART, W.A., 3
`\mp`, 64
`\mulooseness`, 6, 7, 49
`\multnoteskip`, 16
`\musicparskip`, 48
`musicadd.tex`, 9, 11, 14,
 33, 81
`musicdat.tex`, 82
`musicdbr.tex`, 44
`musicfll.tex`, 84
`musicflx`, 7
`musicgui.tex`, 91
`musicltx.tex`, 79
`musiclyr`, 75
`musiclyr.tex`, 93
`musicmad.tex`, 9, 11, 33, 93
`musicxps.tex`, 96
`musicxstr.tex`, 96
`musicxsty.tex`, 96
`musixtex.bat`, 105
`musixtex.sty`, 79
`musixtex.tex`, 11
`musixtmr.tex`, 97
`musixtri.tex`, 97
`MUTEX`, 10
 mxsk font, 39, 42
`\na`, 29
`\nbbbbbbbl`, 81
`\nbbbbbbbu`, 81
`\nbbbbbl`, 81
`\nbbbbbu`, 81
`\nbbbbl`, 22
`\nbbbbu`, 22
`\nbbbu`, 22
`\nbbbl`, 22
`\nbbu`, 22
 neumes, 84, 86
`\nextinstrument`, 17
`\nextstaff`, 17, 26
`\nh`, 19
`\nobarnumbers`, 46
`\nohalfties`, 39, 42
`\nolyr`, 75
`\nopagenumbers`, 49
`\normalbottom`, 48
`\normalmusicsize`, 12, 16,
 73

`\normalnotesize`, 68
`\normaltranspose`, 30
`\normalvalue`, 70, 72
`\normtype`, 57
`\Nosluradjust`, 41
`\nosluradjust`, 41
`\nostemcut`, 66
`\NOTEs`, 15
`\NOTes`, 15
`\NOTes`, 15, 17
`\Notes`, 15
`\notes`, 15, 17
`\notesintext`, 80
`\noteskip`, 15, 17, 20, 21, 28, 93
`\NOTesp`, 15
`\NOTesp`, 15
`\Notesp`, 15
`\notesp`, 15
`\Notieadjust`, 41
`\notieadjust`, 41
`\nq`, 19
`\nspace`, 28

octave clefs, 52
octave treble clef, 67
`\octfindown`, 31
`\octfinup`, 31
`\octnumberdown`, 31
`\octnumberup`, 31
`\OD`, 96
`\Od`, 96
`\off`, 28
`\OH`, 96
`\Oh`, 96
`\oldGclef`, 93
`\oldlyrlinestart`, 76
oriscus, 86
`\oriscus`, 86
ornaments, 64
`\othermention`, 97
`\ovbkt`, 67
`\oxccl`, 94
`\oxccu`, 94
`\oxcl`, 94
`\oxcu`, 94
`\oxqb`, 94
`\oxql`, 94
`\oxqu`, 94
`\oxzq`, 94

`\p`, 64

page and line layout
(global), 49
`\pageno`, 49
page number, 48
`\parindent`, 15, 48
`\PAUSE`, 27, 84
`\PAuse`, 27, 84
`\pause`, 27, 84
`\pausep`, 27, 84
`\PED`, 63
`\Ped`, 63
`\pesquassus`, 87
`\pesquassusauctdown`, 90
`\Pizz`, 96
`\podatus`, 87
`\podatusauctdown`, 90
`\podatusauctup`, 90
`\podatusdeminut`, 91
`\podatusinitiodebilis`, 87
`\Porrectus`, 88
`\Porrectusauctdown`, 90
`\Porrectusdeminut`, 91
`\Porrectusflexus`, 88
`\pp`, 64
`\ppff`, 57
`\ppp`, 64
`\pppp`, 64
`\pppt`, 20
`\ppt`, 20
`\prevstaff`, 17, 26
`psslurs.pro`, 42
`\pt`, 20
punctum, 86
`\punctum`, 86
`\punctumauctdown`, 86
punctum auctum, 86
`\punctumauctup`, 86
punctum deminutum, 86
`\punctumdeminutum`, 86

`\qa`, 18
`\qb`, 15, 21
`\qbp`, 20
`\qbpp`, 20
`\qbppp`, 97
`\ql`, 18
`\qlp`, 20
`\qlpp`, 20
`\qlppp`, 97
`\qp`, 27, 84
`\Qqbbu`, 21
`\Qqbl`, 21
`\Qqbu`, 21
`\qq`, 27, 84
`\qr`, 84
`\Qrtx`, 82
`\qs`, 27, 84
`\qsk`, 28
`\qspace`, 28
`\Qtx`, 82
`\qu`, 15, 18
quilisma, 86
`\quilisma`, 86
`\quilismapes`, 88
`\quilismapesauctdown`, 90
`\qup`, 20
`\qupp`, 20
`\quppp`, 97
`\Qwr`, 84

`\raggedbottom`, 48
`\raggedstoppiece`, 84
`\raise`, 27
`\raisebarno`, 47
`\raiseguitar`, 92
`\raiseped`, 63
`\raisevolta`, 56
raising rests, 27
`\rcl`, 19
`\rcu`, 19
`\relativeaccid`, 31
relative accidentals, 9
`\relax`, 8
repeated patterns, 24
`\resetclefsymbols`, 53, 85
`\resetlayout`, 73
`\resetlyrics`, 76
`\reverseallabreve`, 13
`\reverseC`, 13
`\rh`, 19
`\rhl`, 19
`\rhu`, 19
`\rightrepeat`, 55, 56
`\rm`, 57, 96
`\roccl`, 94
`\roccu`, 94
`\rocl`, 94
`\rocu`, 94
`\roff`, 23, 25, 29
`\roffset`, 29
`\roqb`, 94
`\roql`, 94

`\roqu`, 94
`\rozq`, 94
`\rpar`, 66
`\rq`, 19
`\rql`, 19
`\rqu`, 19
`\rsqu`, 86
`\rtx`, 82
`\rw`, 19

`\salicus`, 89
`\salicusauctdown`, 90
`\salicusflexus`, 89
scalable dimension, 28
scalable width, 4
`\scale`, 16, 77
`\scandicus`, 89
`\scandicusauctdown`, 90
`\scandicusdeminut`, 91
SCHUMANN, R., 108
`\sDEP`, 63
`\sDep`, 63
`\Segno`, 56
`\segno`, 56
`\selectinstrument`, 17
`\selectstaff`, 17
`\sepbarrule`, 44
`\sepbarrules`, 43
`\sepmultibarrule`, 44
`\setaltoclefsymbol`, 84, 85
`\setbassclefsymbol`, 52, 84, 86
`\setclef`, 12, 85
`\setclefsymbol`, 53, 85, 94
`\setcrescheight`, 65
`\setdoubleBAR`, 43
`\setdoublebar`, 42
`\setemptybar`, 43
`\setinterinstrument`, 72, 77
`\setleftrepeat`, 55
`\setleftrightrepeat`, 55
`\setlines`, 73
`\setlyrics`, 75
`\setlyrstrut`, 76
`\setmeter`, 13, 54
`\setname`, 13
`\setrightrepeat`, 55
`\setsign`, 8, 12, 50
`\setsize`, 70, 72
`\setsongraise`, 74, 76

`\setstaves`, 12
`\settrebleclefsymbol`, 52, 67
`\Setvolta`, 55
`\setvolta`, 56
`\sF`, 64
sforzando, 60
`\sh`, 29
`\Shake`, 64
`\shake`, 64
`\Shakel`, 64
`\Shakene`, 64
`\Shakenw`, 64
`\Shakesw`, 64
sharps, 12
`\shiftbarno`, 47
`\shortauthor`, 97
`\shortbarrules`, 93
`\shorttitle`, 97
`\showallbarrules`, 43
`\Showbarrule`, 43
`\showbarrule`, 43, 44
`\showdashbarrule`, 44
`\showdotbarrule`, 44
`\showlyrshifttrue`, 76
sizes, 12
`\sk`, 20, 28
`\sl`, 96
`\slide`, 84
`\slopebrkslursfalse`, 42
`\slopebrkslurstrue`, 42
`\slur`, 38
`\Sluradjust`, 41
`\sluradjust`, 41
`\slurtext`, 41
`\smallaccid`, 29
`\smallfl`, 29
`\smallmusicsize`, 12, 16, 73
`\smallnotesize`, 68
`\smallsh`, 29
`\Smalltype`, 57
`\smalltype`, 57
`\smallvalue`, 70
`\Solid`, 41
`\songbottom`, 13
`\songtop`, 13
`\sopir`, 27
`\SP`, 96
spacing note, 15
`\sPED`, 63

`\sPed`, 63
`\squ`, 86
`\sslur`, 38
staccatissimo, 60
staccato, 60
staccato/tenuto, 60
`\staffbotmarg`, 48, 72–74, 77
staff size, 70
`\stafftopmarg`, 48, 72, 73, 92
`\startextract`, 80
`\startmuflex`, 15
`\startpiece`, 15, 48, 97
`\stdbarrules`, 43
`\stdstemfalse`, 66
`\stemcut`, 66
`\stemfactor`, 73
`\stemlength`, 66, 73
`\stie`, 38
`\Stoppiece`, 6, 48
`\stoppiece`, 6, 7, 42, 48, 55, 97
string instruments, 96
`\subtitle`, 97
`\svtpoint`, 96
`\systemheight`, 73
`\systemnumbers`, 47

`\tbbl`, 22, 26
`\tbl`, 22, 26
`\tBslur`, 41
`\tblslurd`, 34
`\tblsluru`, 34
`\tbu`, 22, 26
`\Tcresc`, 66
`\tcresc`, 65
`\tdecresc`, 65
`\tenpoint`, 96
tenuto, 60
`\tfslur`, 39
`\tgcccl`, 94
`\tgccu`, 94
`\tgcl`, 94
`\tgcu`, 94

<code>\tgqb</code> , 94	<code>\ttie</code> , 34	<code>\wrp</code> , 84
<code>\tgql</code> , 94	<code>\Ttrille</code> , 62	<code>\xbar</code> , 48
<code>\tgqu</code> , 94	<code>\Ttx</code> , 82	<code>\xccl</code> , 94
<code>\tgzq</code> , 94	<code>tuplet.tex</code> , 97	<code>\xccu</code> , 94
<code>\thelyrics</code> , 75	<code>\turn</code> , 64	<code>\xcl</code> , 94
<code>\tHHslur</code> , 40	<code>\twfvpoint</code> , 96	<code>\xcu</code> , 94
<code>\tHslur</code> , 39	<code>\twlpoint</code> , 96	<code>\xqb</code> , 94
<code>\thslur</code> , 39	<code>\twtypoint</code> , 96	<code>\xql</code> , 94
<code>\thsong</code> , 75	<code>\tx</code> , 82	<code>\xqu</code> , 94
<code>\tie</code> , 38	<code>\txt</code> , 67	<code>\xtuplet</code> , 62
<code>\Tieadjust</code> , 41	<code>\UD</code> , 96	<code>\xzq</code> , 94
<code>\tieadjust</code> , 41	<code>\Ud</code> , 96	<code>\yccccl</code> , 83
<code>\tinynotesize</code> , 68	<code>\UH</code> , 96	<code>\yccccu</code> , 83
<code>\tinyvalue</code> , 70	<code>\Uh</code> , 96	<code>\ycccl</code> , 83
<code>\title</code> , 97	<code>\unbkt</code> , 67	<code>\ycccu</code> , 83
<code>\tlcresc</code> , 65	<code>\upbow</code> , 96	<code>\yccl</code> , 83
<code>\tldecrec</code> , 65	<code>\upperfl</code> , 29	<code>\yccu</code> , 83
<code>\tlslur</code> , 39	<code>\upperna</code> , 29	<code>\ycl</code> , 83
<code>\Toctfin</code> , 32	<code>\uppersh</code> , 29	<code>\yclp</code> , 83
<code>\today</code> , 82	<code>\uppz</code> , 60	<code>\yclpp</code> , 83
<code>\torculus</code> , 88	<code>\Uptext</code> , 59, 60	<code>\ycu</code> , 83
<code>\torculusauctdown</code> , 90	<code>\uptext</code> , 59	<code>\ycup</code> , 83
<code>\torculusdebilis</code> , 91	<code>\uptrio</code> , 67	<code>\ycupp</code> , 83
<code>\torculusdeminut</code> , 91	<code>\upz</code> , 60	<code>\ynq</code> , 83
<code>\torculusinitedebilis</code> , 88	<code>\upzst</code> , 60	<code>\yqb</code> , 83
<code>\tqb</code> , 22	<code>\usf</code> , 60	<code>\yql</code> , 83
<code>\Tqbbbl</code> , 21	<code>\usfz</code> , 60	<code>\yqlp</code> , 83
<code>\Tqbbu</code> , 21	<code>\ust</code> , 60	<code>\yqlpp</code> , 83
<code>\Tqbl</code> , 21	<code>\varaccid</code> , 29	<code>\yqu</code> , 83
<code>\Tqbu</code> , 21	<code>\varline</code> , 67	<code>\yqup</code> , 83
<code>\tqh</code> , 22	<code>\verses</code> , 76	<code>\yqupp</code> , 83
<code>\transpose</code> , 8, 30	<code>\vfootnote</code> , 97	<code>\yznq</code> , 83
<code>\trcresc</code> , 65	<code>virga</code> , 86	<code>\yzq</code> , 83
<code>\trdecrec</code> , 65	<code>\virga</code> , 86	<code>\yzqp</code> , 83
<code>\treblelowoct</code> , 52, 67	<code>\voffset</code> , 48, 96, 107	<code>\yzqpp</code> , 83
<code>\trebleoct</code> , 52	<code>volta</code> , 55	<code>\zalaligne</code> , 6, 48
<code>\trigonus</code> , 89	<code>\voltadot</code> , 56	<code>\zalapage</code> , 6, 48
<code>\Trille</code> , 62	<code>\vsize</code> , 48, 96	<code>\zapostropha</code> , 86
<code>\trille</code> , 62	<code>\wh</code> , 18	<code>\zbar</code> , 48
<code>trills</code> , 62	<code>\wholeshift</code> , 60	<code>\zbreath</code> , 64
<code>\triolet</code> , 62	<code>\whp</code> , 20	<code>\zbreve</code> , 19
<code>\tristropha</code> , 87	<code>\whpp</code> , 20	<code>\zccccl</code> , 81
<code>\trivirga</code> , 87	<code>\whppp</code> , 97	<code>\zccccu</code> , 81
<code>\trslur</code> , 39	<code>\wq</code> , 18	<code>\zccccl</code> , 19
<code>\Trtx</code> , 82	<code>\wqq</code> , 18	<code>\zcccclp</code> , 96
<code>\tSlur</code> , 40	<code>\wr</code> , 84	<code>\zcccclpp</code> , 96
<code>\tslur</code> , 33, 34	<code>\writebarno</code> , 47	<code>\zccccu</code> , 19
<code>\Tslurbreak</code> , 37	<code>\writethebarno</code> , 46	<code>\zccccup</code> , 96
<code>\tsslur</code> , 34	<code>\writezbarno</code> , 47	<code>\zccccupp</code> , 96
<code>\tten</code> , 34		

<code>\zcccl</code> , 19	<code>\zdiapunc</code> , 86	<code>\zqbp</code> , 20
<code>\zccclp</code> , 96	<code>\zendextract</code> , 80	<code>\zqbppp</code> , 97
<code>\zccclpp</code> , 96	<code>\zh</code> , 18	<code>\zql</code> , 19
<code>\zcccu</code> , 19	<code>\zhl</code> , 19	<code>\zqlp</code> , 20
<code>\zcccup</code> , 96	<code>\zhlp</code> , 20	<code>\zqlppp</code> , 97
<code>\zcccupp</code> , 96	<code>\zhlppp</code> , 97	<code>\zqp</code> , 20
<code>\zccl</code> , 19	<code>\zhp</code> , 20	<code>\zqpp</code> , 20
<code>\zcclp</code> , 96	<code>\zhpp</code> , 20	<code>\zqppp</code> , 97
<code>\zcclpp</code> , 96	<code>\zhppp</code> , 97	<code>\zqu</code> , 19
<code>\zccu</code> , 19	<code>\zhu</code> , 19	<code>\zqup</code> , 20
<code>\zccup</code> , 96	<code>\zhup</code> , 20	<code>\zquppp</code> , 97
<code>\zccupp</code> , 96	<code>\zhuppp</code> , 97	<code>\zrsqu</code> , 86
<code>\zchangeclfs</code> , 51	<code>\zlonga</code> , 19	<code>\zsong</code> , 74
<code>\zchar</code> , 59, 63, 74	<code>\zlsqu</code> , 86	<code>\zsqu</code> , 86
<code>\zcharnote</code> , 3, 59, 63, 67, 74	<code>\zmaxima</code> , 18, 19	<code>\zstoppiece</code> , 6, 48
<code>\zcl</code> , 19	<code>\zmidstaff</code> , 59	<code>\ztqb</code> , 22
<code>\zclp</code> , 20	<code>\znh</code> , 20	<code>\ztqh</code> , 22
<code>\zclppp</code> , 97	<code>\znotes</code> , 15	<code>\zw</code> , 19
<code>\zcu</code> , 19	<code>\znq</code> , 20	<code>\zwp</code> , 20
<code>\zcup</code> , 20	<code>\zoriscus</code> , 86	<code>\zwpp</code> , 20
<code>\zcuppp</code> , 97	<code>\zq</code> , 3, 18	<code>\zwppp</code> , 97
	<code>\zqb</code> , 19, 21	<code>\zwq</code> , 19