MusiXTeX

Using TeX to write polyphonic
or instrumental music
Version 1.40

Revised November 2, 2025

*If you are not familiar with TeX at all, I would recommend
to find another software package to do musical typesetting.
Setting up TeX and MusiXTeX 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 *(ca. 1995)*

In my humble opinion, that whole statement is obsolete.

Christof BIEBRICHER *(2006)*

MusiXTEX 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 MusiXTEX without specific suffix may be distributed under the name MusiXTEX if different from the original distribution (except obvious bug corrections).

Adaptations for specific implementations (e.g., fonts) should be provided as separate additional TEX or LATEX files which override original definitions.

---

[1]Thanks to the Free Software Foundation for advice. See http://www.gnu.org

Preface

MusiXTEX was developed by Daniel Taupin, Ross Mitchell and Andreas Egler, building on earlier work by Andrea Steinbach and Angelika Schofer. Unfortunately, Daniel Taupin, the main developer, died all too early in a 2003 climbing accident. The MusiXTEX community was shocked by this tragic and unexpected event. You may read tributes to Daniel Taupin that are archived at the Werner Icking Music Archive (WIMA).

Since then, the only significant update to MusiXTEX has been in version 1.15 (April 2011) which takes advantage of the greater capacity of the eTEX version of TEX. This manual is the definitive reference to all features of MusiXTEX version 1.39.

Novice users need not start here. Most music typesetting tasks can be accomplished entirely by using the **PMX** (for instrumental music) or **M-Tx** (for vocal music) preprocessors to generate the MusiXTEX input file, relieving the user of learning any of the commands or syntax of MusiXTEX itself. It is only for out-of-the-ordinary constructions that one must learn these details in order to insert the necessary MusiXTEX commands into the preprocessor's input file as so-called inline TEX.

Advanced users might want to use `autosp`, another preprocessor. It simplifies the production of MusiXTEX scores by automating the choice of note-spacing commands; see Section 3.4.

It is possible to create inputs for MusiXTEX, **PMX**, **M-Tx** or `autosp` using any text editor, such as `notepad` or `vi` or `emacs`. Currently there is no graphical user interface for generating input files, other than TEX-oriented integrated graphical development environments such as TEXWorks.

The Werner Icking Music Archive contains excellent and detailed instructions for installing TEX, MusiXTEX and the strongly recommended preprocessors **PMX** and **M-Tx** on Linux/UNIX, Windows and Mac OS. See this page at WIMA for documentation of **PMX** and **M-Tx**.

<div align="right">

Oliver Vogel
Don Simons
Andre van Ryckeghem
Cornelius Noack
Hiroaki Morimoto
Bob Tennent
November 2, 2025

</div>

Contents

Chapter 1

Introduction to MusiXTEX

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

MusiXTEX is a set of macros and fonts which enables music typesetting within the TEX system. It requires as a prerequisite a working installation of TEX. MusiXTEX 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 or she elects to proceed by entering MusiXTEX 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 MusiXTEX.

Lyrics can also be handled by MusiXTEX. 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 `musixlyr.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 MusiXTEX coexisted only grudgingly, owing primarily to the limited availabily of storage registers. But with modern versions of TEX and with the use of eLATEX, 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 MusiXTEX directly in the document file, namely, by using MusiXTEX to create `pdf` or `eps` files for each of the examples, and then embedding references to those in the file for the larger document. 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 24.1.

1.1 Primary features of MusiXTEX

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, in MusiXTEX the fundamental

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

macro used to represent one of those vertical combs (or one of the columns in Table 1.1) is of the form

$$\texttt{\textbackslash notes ... \& ... \& ... \textbackslash en}^1$$

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:

$$\texttt{\textbackslash notes ... | ... \& ...\textbackslash en}$$

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 basic macros in MusiXTEX: those whose tokens 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 produce 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, MusiXTEX can provide the desired spacing. The main mechanism is through a set of macros described in Section 3.2. 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 in 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 MusiXTEX include
  - note symbols without stems;
  - note symbols with stems, and flags for eighth notes and shorter;

---

[1]The command `\en` is an abbreviation for the command `\enotes`, which may also be used.

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

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 non-spacing 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`; the u in the spacing note macro `\qu` is what generates 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:

The coding is as follows:

```
\begin{music}
  \parindent10mm
  \instrumentnumber{1}                  % a single instrument
  \setname1{Piano}                      % whose name is Piano
  \setstaffs1{2}                        % with two staffs
  \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 l\sk\ql n\en
    \bar
    \Notes\ibu0f0\qb0{dgf}|\qlp i\en
    \notes\tbu0\qb0g|\ibbl1j3\qb1j\tbl1\qb1k\en
    \Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
  \zendextract                          % 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.
- \ibbl1j3 begins a double beam, aligned on the *C* (j at this pitch) with slope 15%.

1.3   The three pass system

1.3.1   Introduction

TEX'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. MusiXTEX needs a more sophisticated horizontal spacing algorithm than is used in TEX.

To understand how MusiXTEX 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 basic spacing unit. For example, in **PMX** all sixteenth notes are typically assigned a scalable width of 1.41\elemskip. One main job of MusiXTEX 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[2] etex jobname. Information about each bar is written to an external file named

---

[2]In many cases it will suffice to run tex, but with MusiXTEX 1.15, it is possible to create a file requiring the enhanced capacity

`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 basic 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 originally written in FORTRAN rather than TEX, the main reason being the lack of an array handling capability in TEX; the current version of `musixflx` is a LUA script, which may be executed without compilation in any standard TEX system.

`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 TEX-ed again, by entering `etex jobname`. 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.

You may wonder how the same command can cause something different to happen the second time it is issued. The explanation is that MusiXTEX checks for the presence of `jobname.mx2`. If it's not present, the first pass is executed; if it is, the third pass. Obviously if you've made some corrections that affect any horizontal spacing, you must remember to delete the old `jobname.mx2` and then rerun all three passes, or build or use a script that does so.

1.3.2 An example

Here's an example. Here, no value for `\elemskip` is explicitly specified, so MusiXTEX assigns a single, default value. 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 TEX-ing the second time you'll get:

---

of `etex`. Since `etex` can handle any file `tex` can, it's best always to use `etex`.

Now MusiXTEX 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 0pt
\generalsignature{-3}
\nostartrule
\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}\en
\notes\nbbu0\qb0{=a}\tqh0N\en
\Notes\Dqbu cf\en\bar
\NOtes\uptext{\it tr}\qu e\uptext{\it tr}\qu d\en\bar
\NOtes\qu c\qp\en\Endpiece
```

1.3.3 Adjusting the layout

One benefit of the 3-pass system is the quick and easy alteration to the layout which can be achieved by changing only one parameter, namely `\mulooseness`. This value acts analogously to TEX's `\looseness` command. For non-TEX-perts: if you state `\looseness=-1` somewhere inside any paragraph, then TEX will try to make the paragraph one line shorter than it normally would. With `\mulooseness`, MusiXTEX does the same, but for *systems* and *sections* rather than lines and 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. Somewhere[3] before the end of the section, you can change the value of `\mulooseness` to something different from the default of 0, and MusiXTEX 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:

---

[3] Advisably, at the beginning or at the end of the section, for the sake of clarity.
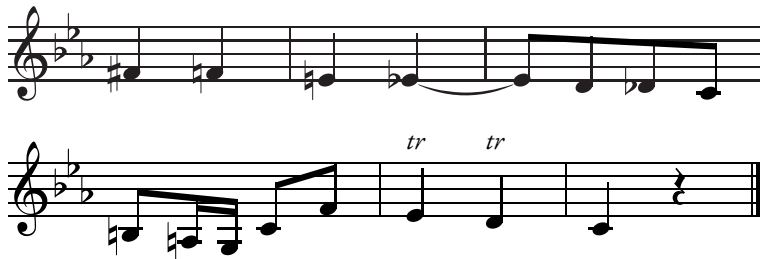
On the other hand,

```
\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 MusiXTeX input file manually (which in fact 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 relative horizontal spacing for each group (via the macros discussed in Section 3.2) 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 3.

2. TeX $\Longrightarrow$ `musixflx` $\Longrightarrow$ TeX.

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
   TeX $\Longrightarrow$ `musixflx` $\Longrightarrow$ TeX.

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 MusiXTeX's way of using scalable space and the three pass system. In TeXnical 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.

---

[4]This is the technique always used by **PMX** in constructing a MusiXTeX input file.

1.3.4　Using `musixflx`

On most systems, `musixflx` can be 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.mxl`
- s　to get the computed lines immediately on screen

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

1.3.5　Unrecorded spaces: the novice's bugaboo

Because of the way MusiXTEX 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 MusiXTEX macro, will not be properly recognized and recorded by MusiXTEX, but it will still be treated like ordinary text by TEX. The symptoms of such a transgression will be an **Overfull 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` and `\en`).

2. `\off` must only be used with scalable values, e.g., `\noteskip`, `\elemskip`, `\afterruleskip`, and `\beforeruleskip`.

3. Remember that `\qqsk`, `\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-MusiXTEX text must be put in zero-width 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-placement macros defined in `musixlyr.tex` (see Chapter 23, p. 111).

---

[5]Note that `\hardspace` does not fall in this category; it is specifically designed to enter hard space in a way that MusiXTEX can properly record it

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 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 with 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.

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 9.2. Now, finally, you can enter `\transpose=1` with more or less the desired effect.

More or less, because 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 MusiXTEX. These macros do the right thing in the face of transposition. They are discussed in Chapter 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 required[6].

1.4.3 Extracting parts from a score

It is possible to extract separate scores for each individual instrument of a multi-instrument score. If you use **PMX**, it is very easy; see the documentation for **PMX**. If you use MusiXTEX without a preprocessor, see Chapter 22.

---

[6]**PMX** will automatically adjust both stem and beam directions when transposing, which is yet another reason to use it. However if a piece is to be transposed, the typesetter must still explicitly activate relative accidentals and enter accidentals according to the relative-accidental convention.

1.4.4   Staff and note sizes

Although the standard staff size is 20pt, MusiXTEX allows scores with sizes of 16, 24, or 29pt. Further-more, 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 MusiXTEX'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 MusiXTEX 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 a MusiXTEX distribution, 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 num-ber 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 typeset-ting lyrics. In fact the latter two, while now included in MusiXTEX distributions, are not documented in this manual but in separate files, namely `musixps.tex` itself and `mxlyrdoc.pdf` respectively.

1.5   Installing and Using MusiXTEX

As stated in the preface, the Werner Icking Music Archive (WIMA) contains excellent and detailed instruc-tions for installing TEX, MusiXTEX and the strongly recommended preprocessors **PMX** (for instrumental music) and **M-Tx** (for vocal music) under Linux/UNIX, Windows and Mac OS.

   As stated in Section 1.3, to compile a MusiXTEX-formatted score, you must run in sequence `etex`, `musixflx`, and `etex` again. This will produce a `.dvi` file. From this point forward, this file can be pro-cessed like the device-independent output file for any TEX document. The most common approach and the only one to be discussed here is to next generate a Postscript file. This is accomplished by running `dvips`, which is included in every TEX system; to prevent inappropriate adjustments that `dvips` makes by default, it is recommended to add the option `-e0`. To view the Postscript, most users run `GSview` or `gv`, a front-end to `ghostscript`, which must be separately downloaded and installed. Finally, you may pro-duce a PDF file using `ps2pdf` or GSview, and view that with any PDF viewer, such as the Adobe Reader or GSview.

   You also can view the `.dvi` file directly using a DVI viewer such as Yap or `xdvi`, but, if you are using type K Postscript slurs or vectorized beams, some DVI viewers will not display them.

   Rather than executing all the foregoing programs separately, most users will find it more con-venient to use the `musixtex` script that comes in most distributions of MusiXTEX. For example, `musixtex jobname` will by default run the three passes described in Section 1.3, convert the result to PDF and delete intermediate files. There are many options available to change the behavior; run

   `musixtex -h`

to get a summary.

1.6 Where to get help using MusiXTEX

The home base for all matters related to MusiXTEX is the Werner Icking Music Archive (WIMA), at
http://icking-music-archive.org. There is a mailing list, hosted by TUG, the TEX Users Group, where you
will always find someone willing to answer questions and help solve problems. A multilingual of musical
terms is available at https://lilypond.org/glossary.html.

1.7 A very brief history of MusiXTEX

The idea of using TEX to typeset music appears to have originated around 1987 with the master's thesis of
Andrea Steinbach and Angelika Schofer[7]. They called their package MuTEX. 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 TEX glue to help control horizontal spacing and justification.

The next major step came around 1991 when Daniel Taupin created MusicTEX. Its major enhance-
ment 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.

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

Significant enhancements to MusiXTEX, 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, MusiXTEX has been maintained by a varying cast including
Olivier Vogel, Hiroaki Morimoto, Bob Tennent, Andre Van Ryckeghem, Cornelius Noack, and Don
Simons.

No discussion of the history of MusiXTEX would be complete without mentioning the contributions
of Werner Icking. From the early days of MuTEX 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 Bob Tennent, and the mailing list by Maurizio Codogno.

---

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

Chapter 2

Setting up the Input File

2.1 What makes a TEX file a MusiXTEX file?

A MusiXTEX 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 MusiXTEX macros. After that might follow `\input musixadd` or `\input musixmad` if you want to have, respectively, up to nine or twelve instruments or simultaneous beams, ties, or slurs. If you want to have greater numbers of these elements, you can assign them directly by including one or more of these commands: `\setmaxinstruments`, `\setmaxcxxviiibeams`, `\setmaxcclvibeams`, `\setmaxgroups`, `\setmaxslurs`, `\setmaxtrills`, `\setmaxoctlines`.

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.

2.2 Cautions for the non TEXpert

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 `\en` 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`.

2.3 Usual setup commands

2.3.1 Music size

The first decision is what size type to use. MusiXTEX offers four sizes: "small" (16pt-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.

2.3.2 Number of instruments

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. The default maximum number of instruments is 6 but this can be increased by using `\setmaxinstruments{`n`}`.

2.3.3 Number of staves per instrument

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

> `\setstaffs{`n`}{`p`}`

where p is the number of staves and n is the number of the instrument. In MusiXTEX, instruments are numbered *starting with the lowest*. So for example `setstaffs32` assigns two staves to the third instrument from the bottom.

2.3.4 Clefs for each staff

The default clef for every staff is the *treble* clef. To assign any other clef, the command is

> `\setclef{`n`}{`$s_1 s_2 s_3 s_4$`}`

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 used to specify an "empty" clef. $s = 9$ 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}`.

Treble and bass clefs with the digit 8 above or below are also possible; see Section 15.3.

2.3.5   Key signatures

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.

2.3.6   Meters

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* (e.g., 3/4) the command is `\generalmeter{\meterfrac{3}{4}}`. Other possible tokens m are `\meterC`, `\allabreve`, `\reverseC`, `\reverseallabreve`, `\meterN` and `\meterplus`. To insert extra space before the meter is written, use `\meterskip`d where d is any hard TeX dimension[2]. The assignment must occur outside `\startpiece...\endpiece` and will be reset to zero after first meter is posted.

These are illustrated in the following example:

which was coded as:

```
\generalmeter\meterC
\nostartrule
\parindent0pt\startpiece
\NOtes\qa{cegj}\en
\generalmeter\allabreve\changecontext
\NOTes\ha{ce}\en
\generalmeter\reverseC\changecontext
\NOTES\zbreve g\en
\generalmeter\reverseallabreve\changecontext
\NOTEs\zwq g\en
```

---

[1]We 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 MusiXTeX.

[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*.

```
\generalmeter{\meterfrac{3\meterplus2\meterplus3}8}\changecontext
\notesp\Tqbu ceg\Dqbl jg\Tqbu gec\en
\generalmeter{\meterN3}\meterskip4pt\changecontext
\Notes\qa{ceg}\en\setemptybar
\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.

2.3.7   Instrument names

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 `\parindent`*d* where *d* is any hard TEX dimension. For multi-line instrument names, use TEX commands as in the following:

```
\setname1{\vbox{%
  \hbox to\parindent{\hss Bass\hss}%
  \hbox to\parindent{\hss Recorder\hss}}}
```

2.4   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 .

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}
```

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` or `musixmad.tex`, the allowable number of groups is increased to four. Alternatively, you can specify the allowable number of groups to *m* by `\setmaxgroups{m}`[3].

---

[3]Using $m > 4$ may require e-TEX.

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{{`*lower₁*`}{`*upper₁*`}{`*lower₂*`}{`*upper₂*`}{`*lower₃*`}{`*upper₃*`}}`

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 two 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 command

`\curlybrackets{{`*lower₁*`}{`*upper₁*`}{`*lower₂*`}{`*upper₂*`}...`

to be used as illustrated in the third example below.

is coded as:	is coded as:	is coded as:

```
\sepbarrules
\smallmusicsize
\instrumentnumber{5}
\akkoladen{{1}{2}{3}{5}}
\startextract
\notes\en\bar\notes\en
\zendextract
```

```
\sepbarrules
\smallmusicsize
\instrumentnumber2
\setstaffs12
\setstaffs23
\startextract
\notes\en\bar\notes\en
\zendextract
```

```
\sepbarrules
\instrumentnumber5
\smallmusicsize
\curlybrackets{1235}
\startextract
\notes\en\bar\notes\en
\zendextract
```

Chapter 3

Preparing to Enter Notes

3.1   After the setup, what next?

The command \startmuflex initiates the serious business of MusiXTEX. 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.

3.2   Horizontal spacing commands

3.2.1   Basic note spacing

MusiXTEX provides a set of macros each of which defines a particular increment of scalable spacing. The default set is tabulated in Figure 3.1. 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, \hl) 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 \geometricskipscale. As implied by the name, this is a geometric progression, where \Notes is $\sqrt{2}$ times wider than \notes, \NOtes 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

Usage	spacing	Suggested use
`\znotes ... & ... & ... \en`	(non spacing)	specials
`\notes  ... & ... & ... \en`	`2\elemskip`	♪16th
`\notesp ... & ... & ... \en`	`2.5\elemskip`	♪·dotted 16th, 8th triplet
`\Notes  ... & ... & ... \en`	`3\elemskip`	♪8th
`\Notesp ... & ... & ... \en`	`3.5\elemskip`	♪·dotted 8th, quarter triplet
`\NOtes  ... & ... & ... \en`	`4\elemskip`	♩ quarter
`\NOtesp ... & ... & ... \en`	`4.5\elemskip`	♩·dotted quarter, half triplet
`\NOTes  ... & ... & ... \en`	`5\elemskip`	♩ half
`\NOTesp ... & ... & ... \en`	`5.5\elemskip`	♩·dotted half
`\NOTEs  ... & ... & ... \en`	`6\elemskip`	○ whole

Table 3.1: Spacing Commands

`\NOTES`, will be defined corresponding to factors 9.52 and 11.32. The original arithmetic progression can be restored by `\arithmeticskipscale`.

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\NOtes{\vnotes5.34\elemskip}` will redefine `\NOtes` in the obvious way, and the extension to the other spacing macros should likewise be obvious. The following abbreviations may be useful:

    \nnotes   for   \vnotes1.6\elemskip
    \nnnotes   for   \vnotes1.28\elemskip

In addition, inside any pair `\notes ... \en`, there exist 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 until `\noteskip` is further redefined. Finally, by issuing a command like `\scale{2.4}` outside any notes group, you can scale all subsequent `\noteskip`s 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.

3.2.2   Before and after rule skips

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 aways 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:

using	`\elemskip`	`\afterruleskip`	`\beforeruleskip`
`\normalmusicsize`	6pt	8pt	0pt
`\smallmusicsize`	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`[1].

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:
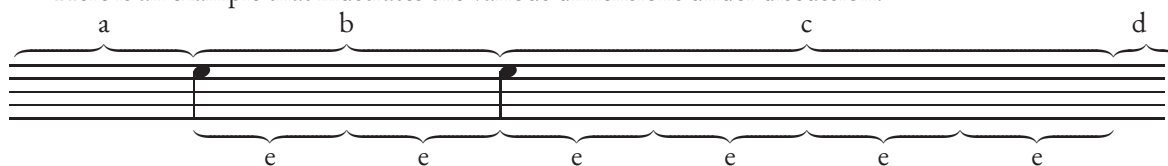

```
a → \afterruleskip
b → \notes = \vnotes 2\elemskip
c → \NOtes = \vnotes 4\elemskip
d → \beforeruleskip
e → \elemskip
```

3.3   Moving from one staff or instrument to another

When entering notes inside `\notes ... \en`, 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 24.3) 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 the *same* staff by using `\selectstaff{n}` with n for the *current* staff.

3.4   Automatic note spacing

A preprocessor called `autosp` automates the generation of note-spacing commands by converting (nonstandard) commands of the form `\anotes ... \en` into one or more conventional note-spacing commands, as determined by the note values themselves, with `\sk` spacing commands inserted as necessary. The coding for an entire measure can be entered one part at a time, without concern for note-spacing changes within the part or spacing requirements of other parts. For example, from the input

---

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

```
\anotes\qa J\qa K&\ca l\qa m\ca n\en
```

`autosp` generates

```
\Notes\qa J\sk\qa K\sk&\ca l\qa m\sk\ca n\en
```

`autosp` also supports non-standard commands for adding "global" skips (i.e., in every staff) and centered whole-bar rests.

The `musixtex` script will automatically use `autosp` as a pre-processor on files with the `.aspc` extension. The `autosp` package is available in [TeXLive](#) and [MiKTeX](#), and can also be found at the [Werner Icking Music Archive](#) or at [CTAN](#) (the Comprehensive TeX Archive Network).

Chapter 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.

Note pitches can be specified either by letters or numbers. If no transposition or octaviation 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 octaviation.

Notes lower than `A` and higher than `z` can be entered, with either numbers as just described, or with octaviation as will be explained in Chapter 9.

4.1   Normal (unbeamed) spacing notes

In the following, {p} signifies a pitch specification as described above. 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 direction[1] .
> `\maxima{`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.

---

[1]Editor's note: Evidently there is no explicit up-stemmed longa

`\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 note[2] 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*`}:` 64th note with stem up.
`\ccccl{`*p*`}:` 64th note with stem down.
`\cccca{`*p*`}:` 64th note with automatic stem direction.
`\cccccu{`*p*`}:` 128th note with stem up.
`\cccccl{`*p*`}:` 128th note with stem down.
`\ccccca{`*p*`}:` 128th note with automatic stem direction.

As an example, the sequence

was coded as

```
\Notes\cu c\cl j\en\bar
\Notes\ccu c\ccl j\en\bar
\Notes\cccu c\cccl j\en\bar
\Notes\ccccu c\ccccl j\en\bar
\Notes\cccccu c\cccccl j\en
```

   For `\breve`, there is a possibility of confusion with the command in TEX's math mode that has exactly the same name; however, generally there will not be problem because inside `\startpiece...\endpiece`, `\breve` will take the musical meaning, while outside, it will have the mathematical one.   If for some reason you need the math `\breve` inside, you can use `\endcatcodesmusic ... \breve ... \catcodesmusic`.

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.

---

[2]The "c" within this macro name stands for the equivalent British term "crotchet"

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

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 right[3].
`\lh{`*p*`}:` half note head shifted left.
`\rq{`*p*`}:` quarter note head shifted right.
`\lq{`*p*`}:` quarter note head shifted left.

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, \zcccccu:` eighth, ..., note with stem up but no spacing. They act like `\cu` for chord building.
`\zcl{`*p*`}, \zccl, \zcccl, \zccccl, \zcccccl:` eighth, ..., note with stem down but no spacing. They act like `\cl` for chord building.
`\rhu{`*p*`}, \rhl, \rqu, \rql, \rcu, \rcl:` `\rhu` acts like `\zhu`, but the note is shifted one note width to the right; others analogous.
`\lhu{`*p*`}, \lhl, \lqu, \lql, \lcu, \lcl:` 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.

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.

---

[3]Some may not have realized that half and whole note heads have different shapes

As an example, the sequence

was coded as

```
\notes\nq c\nq j\en\barre
\Notes\nh c\nh j\en\barre
\notes\nq {cdef}\en
```

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

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`, `\whpp`, `\zwp`, `\zwpp`, `\hup`, `\hupp`, `\hlp`, `\hlpp`, `\hap`, `\happ`, `\zhp`, `\zhpp`, `\qup`, `\qupp`, `\qlp`, `\qlpp`, `\qap`, `\qapp`, `\zqp`, `\zqpp`, `\cup`, `\cupp`, `\clp`, `\clpp`, `\qbp` and `\qbpp`, all followed by a pitch argument. 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{`*p*`}`, or `\pppt{`*p*`}` 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`. 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`, `\lppt` and `\lpppt`.

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`, followed by a pitch, and similarly with two p's for double-dotted notes. The following provide dotted chord notes shifted left or right: `\lhp`, `\rhp`, `\lqp`, `\rqp`, and similarly with two p's for double-dotted notes.

4.7   Sequences of equally spaced notes; collective coding

It isn't necessary to write a separate macro sequence `\notes...\en` 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 . 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 "ˆ" 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.

Chapter 5

Beams

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. It is needed because more than one beam may be open at a time, and it tells MusiXTEX to which beam subsequent beamed notes and other beam specification commands are assigned. By default, the reference number must be in the range $[0, 5]$, but the range for 8th to 128th beams will be expanded to $[0, 8]$ or $[0, 11]$ if `musixadd` or `musixmad` respectively has been \input.

Alternatively, you can specify the number of 8th to 64th beams directly[1] with

   \setmaxinstruments{m}

within the range $7 < m \leq 100$; the corresponding reference number may then be in the range from 0 to $(m-1)$. For 128th beams, use \setmaxcxxviiibeams{m}. For 256th notes, which can only appear in beams, see Section 25.4.

The "pitch" parameter p 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 okay 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*.

---

[1] 8th to 64th beams are so basic that the maximum number of these beams is related to the maximum number of instruments by this command. Using $m > 12$ may require e-TEX.

`\ibbbl{`*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`, MusiXTEX 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`\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 MusiXTEX will raise (or lower) the starting point.

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

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. MusiXTEX 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`, `\Dqbbl`, `\Tqbu`, `\Tqbl`, `\Tqbbu`, `\Tqbbl`, `\Qqbu`, `\Qqbl`, `\Qqbbu` and `\Qqbbl`. 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 as[2]:

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

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*`}`.

---

[2]Editor'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.

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
`\ztqh{`*n*`}{`*p*`}` is equivalent to `\tbu{`*n*`}\zqb{`*n*`}{`*p*`}` i.e., no spacing afterwards

The following synonyms may be used:

`\tql`	for	`\tqb`
`\tqu`	for	`\tqh`
`\ztql`	for	`\ztqb`
`\ztqu`	for	`\ztqh`

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 `\nbbl{`*n*`}`…`\nbbbbl{`*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\en
```

To decrease multiplicity to one, use `\tbbu{`*n*`}` or `\tbbl{`*n*`}`. To decrease to two or three use `\tbbbu{`*n*`}`…`\tbbbbl{`*n*`}`. For example,

has been coded as

```
\startextract
\Notes\ibbbu0h0\qb0e\tbbbu0\qb0e\tbbu0\qb0e\tbu0\qb0e\en
\zendextract
```

Although at first it may seem counterintuitive, the macros `\tbbu` and `\tbbl` and higher order counterparts may also be invoked when the multiplicity is one. In this case a second, third, or 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%`
                         `\roff{\tbbu0\tqh0e}\en`   `\roff{\tbbbu0\tbbu0\tqh0e}\en`

The following abbreviations have been provided:

    `\tqqb{`*n*`}{`*p*`}` is equivalent to `\tbbl{`*n*`}\tqb{`*n*`}{`*p*`}`
    `\tqqh{`*n*`}{`*p*`}` is equivalent to `\tbbu{`*n*`}\tqh{`*n*`}{`*p*`}`
    `\tqqqh{`*n*`}{`*p*`}` is equivalent to `\tbbbu{`*n*`}\tqqh{`*n*`}{`*p*`}`
    `\tqqqb{`*n*`}{`*p*`}` is equivalent to `\tbbbl{`*n*`}\tqqb{`*n*`}{`*p*`}`
    `\nqqb{`*n*`}{`*p*`}` is equivalent to `\tbbl{`*n*`}\qb{`*n*`}{`*p*`}`
    `\nqqh{`*n*`}{`*p*`}` is equivalent to `\tbbu{`*n*`}\qb{`*n*`}{`*p*`}`
    `\nqqqh{`*n*`}{`*p*`}` is equivalent to `\tbbbu{`*n*`}\qb{`*n*`}{`*p*`}`
    `\nqqqb{`*n*`}{`*p*`}` is equivalent to `\tbbbl{`*n*`}\qb{`*n*`}{`*p*`}`

and the following synonyms may be used:

`\tqql`	for	`\tqqb`
`\tqqu`	for	`\tqqh`
`\tqqql`	for	`\tqqqb`
`\tqqqu`	for	`\tqqqh`
`\nqql`	for	`\nqqb`
`\nqqu`	for	`\nqqh`
`\nqqql`	for	`\nqqqb`
`\nqqqu`	for	`\nqqqh`

The symmetrical pattern is also possible. For example:

has been coded as:

    `\Notes\ibbl0j0\roff{\tbbl0}\qb0j\tbl0\qbp0j\en`

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 MusiXTEX 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\ibl0p0\qb0p\nbbl0\qb0p\nbbbl0\qb0p\tbu0\qb0e\en
```

One may save some typing by defining personalized TEX 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.

5.5   Shorthand beam notations for repeated or alternated notes

Sometimes you may want to indicate repeated or alternated short notes with open note heads joined by a beam. Here are examples of how to do that using the \hb macro:

which has been coded as:

```
\Notes\ibbl0j0\hb0j\tbl0\hb0j\en
\Notes\ibbl0h4\hb0h\tbl0\hb0j\en
```

There are also dotted and double-dotted versions \hbp and \hbpp.

A different look could be obtained as follows:

which has been coded as:

```
\notes\ha j\loffset{0.5}{\ibbl1n4}\qsk\tbl1\qsk\ha l\sk\en
```

It is possible to combine this notation with a conventional beam, as in

which has been coded as:

```
\notes\ibl0h3\qb0j\loffset1{\ibbbl1m3}\qsk\tbl1\qsk\tql0l\en
```

or, for whole notes, to dispense with stems:

which has been coded as:

```
\Notes\loff{\zw j}\ibbl0l4\sk\tbl0\wh l\en
\Notes\ibbu0e4\wh g\tbu0\roff{\wh i}\en
```

Short "beams" centered on note stems (or directly over or under whole notes) indicate *tremolos*. Commands that generate tremolo indications are described in Section 18.4.4.

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 Troldhaugen":

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

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

5.7   Beams with notes on several different staves

Here's a simple example from Brahms's Intermezzo Op. 118, provided by Miguel Filgueiras:
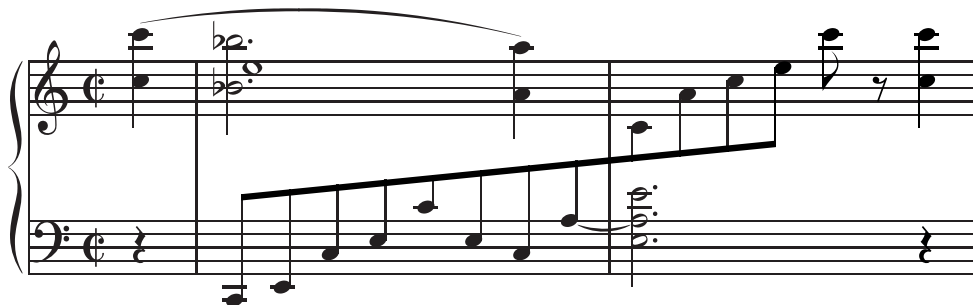
The coding is

```
\interstaff{13}
\instrumentnumber{1}
\setstaffs1{2}
\setclef1\bass
\generalmeter\allabreve
\startextract
\NOtes\qp\nextstaff\isluru0q\zq{q}\ql{j}\en
\bar
\nspace
\Notes\ibu0a1\qb0{CEJLcL}%
  \nextstaff\roff{\zw{l}}\pt{p}\zh{_p}\pt{i}\hl{_i}\en
\Notes\qb0J\itied1a\qb0a\nextstaff\tslur0o\zq{o}\ql{h}\en
\bar
\Notes\ttie1\zh{.L.a}\hl{.e}%
  \nextstaff\qb0{chj}\tbl0\qb0l\cl{q}\ds\en
\NOtes\qp\nextstaff\zq{q}\ql{j}\en
\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 MusiXTEX 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 3.3.

Here is an example:
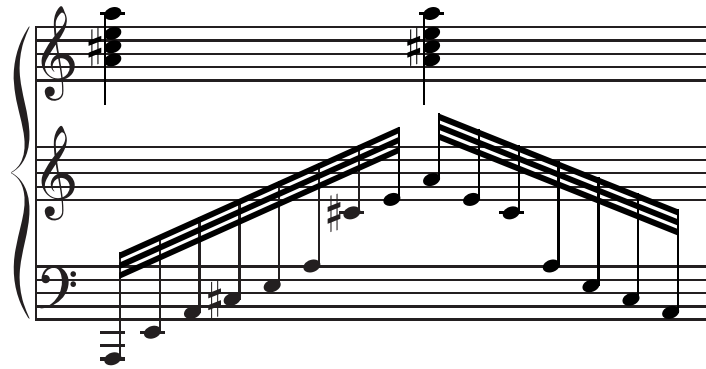
which is coded as:

```
\setstaffs13
\setclef1{6000}
\startextract
\notesp
\nextstaff\Ibbbu0Af7\prevstaff
\qb0{AEH^JLa}\relax\nextstaff
\qb0{*****^c}\tqh0e\relax
|\zq{h^jl}\ql o\en
\notesp
\nextstaff
\Ibbbu0hG6\qb0{hec}\prevstaff
\qb0{***aLJ}\tqh0H\relax\nextstaff
|\zq{h^jl}\ql o\en \nspace
\zendextract
```

In this example we see not only multiple uses of `\nextstaff` and `\prevstaff`, but also the character `*` to make virtual beam notes (see Section 4.7).

5.8 Discontinuities in Long Beams

By default, `musixtex` produces beams using special fonts. Unfortunately, long beams may have unsightly discontinuities (gaps or bumps). See Section 25.37 for a solution to this problem.

Chapter 6

Rests

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`, half rest `\hpause` or `\hp`, quarter rest `\qp` or `\soupir`, eighth rest `\ds`, sixteenth rest `\qs`, 32nd rest `\hs`, and 64th rest `\qqs`. Dotted rests may be obtained by using `\pausep`, `\hpausep` or `\hpp`, `\qpp`, `\dsp`, `\qsp`, `\hsp` and `\qqsp`, and double-dotted rests by `\pausepp`, `\hpausepp` or `\hppp`, `\qppp`, `\dspp`, `\qspp`, `\hspp` and `\qqspp`,.

Longer rests, normally interpreted as lasting two or four bars respectively, can be coded as `\PAuse` and `\PAUSe`, which yield:
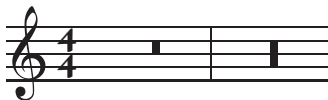
The command `\Hpause` p n provides an arbitrary length pause at pitch p and of length n `\noteskip`. For example, `\NOTes\Hpause4{0.83}\en` produces

6.2 Raising rests and non-spacing rests

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

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

where `\Interligne` is the distance from one staff line to the next. Similarly, non-spacing rests may be coded using the TEX command `\rlap`, as in

for which the coding is

```
\notes\ibbu0h{-1}\zccl e\qb0h\raise-4\Interligne\rlap\qs\qb0e\en
\notes\raise-4\Interligne\rlap\ds\qb0f\tqu0g\en
```

In addition, four special macros are available to raise or lower a whole or half rest. 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-spacing[1], are are

- `\liftpause` *n* to get a ▬ raised from original position by *n* staff line intervals,
- `\lifthpause` *n* to get ▬ raised the same way.
- `\liftpausep` *n* to get a ▬· raised from original position by *n* staff line intervals,
- `\lifthpausep` *n* to get ▬· raised the same way.

6.3 Bar-centered rests

Sometimes it is necessary to place a rest (or some other symbols) exactly in the middle of the notes region of a bar (i.e., excluding any initial clef and key or time signatures). This can usually be done with combinations of the commands `\atnextbar`, `\centerbar`, `\cPAUSe`, `\cPAuse`, `\cpause`, `\chpause`, `\cHpause` *n*, as demonstrated in the following example:

with the coding

```
\generalmeter\meterC
\generalsignature{3}
\setclef1\bass
\setstaffs1{2}
\parindent0pt
\startextract
\NOtes|\qa{cegj}\en
\def\atnextbar{\znotes\centerbar{\cpause}|\en}%
\generalsignature{-3}\changecontext
\NOTes\ha{Nc}|\en
\def\atnextbar{\znotes|\centerbar{\cpause}\en}%
\setclef1\alto\changeclefs%
\generalmeter{\meterfrac21}\changecontext%
\NOTEs\wh d\pause|\en
```

---

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

```
\def\atnextbar{\znotes|\centerbar{\cPAuse}\en}%
\bar
\NOTEs|\en
\def\atnextbar{\znotes\centerbar{\cHpause{12}}|\centerbar{\cHpause{12}}\en}%
\endextract
```

The following abbreviations are provided:

```
\centerpause       for   \centerbar{\cpause}
\centerhpause      for   \centerbar{\chpause}
\centerPAuse       for   \centerbar{\cPAuse}
\centerPAUSe       for   \centerbar{\cPAUSe}
\centerHpause n    for   \centerbar{\cHpause n}
```

Material other than these rests may be used as arguments to `\centerbar`, as in:

for which the coding is:

```
\generalmeter\meterC
\parindent0pt
\startextract
\NOTEs\en\def\atnextbar{\znotes\centerbar{\ccn{9}{\meterfont7}}%
\centerbar{\cPAUSe\off{2\elemskip}\cPAuse\off{2\elemskip}\cpause}\en}\bar
\NOtes\ql{ghij}\en
\def\atnextbar{\znotes\centerbar{\liftpause2}\en}\bar
\NOTes\en
\def\atnextbar{\znotes\centerbar{\duevolte}\en}\bar
\NOTes\en
\def\atnextbar{\znotes\loffset{0.7}{\centerbar{\Fermataup l\wh j}}\en}
\endextract
```

In the last measure, it has been necessary to use `\loffset` to correct the centering of the off-center command `\wh`. . . .

Chapter 7

Skipping Spaces and Shifting Symbols

We've already mentioned that when coding a sequence of notes inside a particular pair `\notes...\en`, 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,
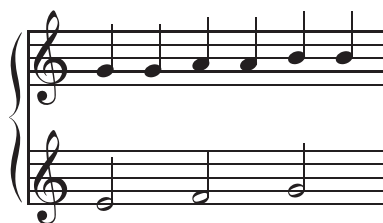
was coded as `\Notes\hu{e*f*g}|\qu{gghhii}\en`

To skip forward by one half of a `\noteskip`, use `\hsk`. To skip *backward* by one or half `\noteskip`, use `\bsk`, or `\hbsk`, respectively.

To insert spacing of approximately one note head width, you can use `\qsk`, or for two-thirds, half or one-quarter of that, `\tqsk`, `\hqsk` or `\qqsk`, respectively. To skip backward one note-head width or two-thirds or half that, use `\bqsk`, `\btsk` or `\bhsk`, respectively. 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...\en` 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...\en` 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...\en` 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 that enable shifting a note, symbol, or sequence inside `\notes...\en` 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\en
```

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\en
```

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.

Chapter 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*}, \dfl{*p*} and \dsh{*p*} respectively. Thinner accidentals are available using the definitions enabled by extension library \musixthacc described in Section 25.33.

The alternate macros \lfl, \lna, \lsh, \ldfl 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♯*. 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 is done by preceding the name of the accidental keyword with "c". Available cautionary accidentals are \csh, \cfl, \cna, \cdfl and \cdsh, which give

The distance between notes and accidentals is controlled by \accshift=*any TEX dimension*, where positive values shift to the left and negative to right, with a default of 0pt. For "big" cautionary accidentals, use, for example, {\largenotesize\csh{*p*}} or see Section 18.8.

Chapter 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*.

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` outside of notes commands, 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 increase or decrease `\transpose` by 7. A more convenient way to transpose locally up or down by one octave makes use respectively of the characters ' (close-quote, interpreted as an *acute accent*) and ' (open-quote, interpreted as a *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; thus, for example, `\qu{''A'A}` and `\qu{''A}\qu{'A}` are both equivalent to `\qu{ah}`. Alterations to the value of `\transpose` in notes commands are *local*: when changing to a different staff or instrument or encountering `\en`, `\transpose` will be reset to the value it had before the accents were used. (That value is stored in another register called `\normaltranspose`). Another way to localize changes to `\transpose` is to create an explicit TEX "group" by enclosing commands in `{...}` braces.

At any point it is possible to reset the `\transpose` register explicitly to the `\normaltranspose` value 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 `\normaltranspose`, regardless of the number of grave and acute accents earlier in that group.

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♯ occurs in the key of C major. If the piece is transposed up three steps to the key of F, the F♯ should logically become a B♮. But if all you do is set `\transpose=3`, the note will be typeset as a B♯. In other words, MusiXTEX will interpret the `\sh` or ^ to mean "print a ♯".

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 MusiXTEX 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. MusiXTEX 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♯ 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♯. With `\transpose=3` and `\generalsignature{-1}` it will appear (correctly) as B♮. Conversely, suppose you want to enter a B♮ 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♮. Now to transpose this to C major, set `\transpose=-3` and `\generalsignature0`, and it will appear as F♯.

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

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 *8va bassa*. Thus

is coded

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

while
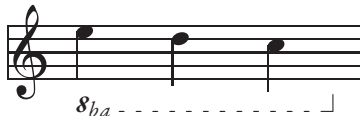
is coded as

```
\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 $n\,p$ indicates an upward octave transposition line with reference number n and with dashed line at pitch p. By default n must be in the range $[0, 5]$, but you can specify a larger maximum number directly with \setmaxoctlines{m} where $7 < m \le 100$[1]; the reference number will be in the range between 0 and $m - 1$.

Usually p will be numeric and >9, but it can also be a letter. \Ioctfindown $n\,p$ 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,
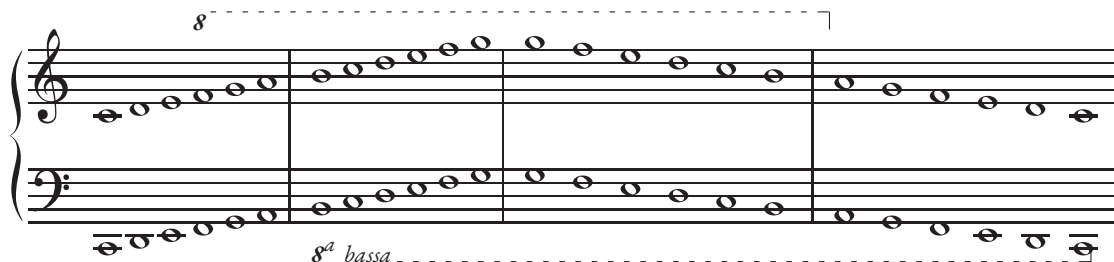
is coded as

```
\begin{music}\nostartrule
\instrumentnumber{1}
\setstaffs12
\setclef1{6000}
%
\startextract
\notes\wh{CDEFGH}|\wh{cde}\Ioctfinup 1p\wh{fgh}\en
\bar
\notes\Ioctfindown 2A\wh{IJKLMN}|\wh{ijklmn}\en
\bar
\Notes\wh{NMLKJI}|\wh{nmlkji}\Toctfin1\en
```

---

[1]This may require e-TeX.

```
\bar
\Notes\wh{HGFED}\Toctfin2\wh C|\wh{hgfedc}\en
\zendextract
\end{music}
```

The elevation of octaviation lines may be changed in midstream using `\Liftoctline` $n\,p$, where n is the reference number of the octave line, and p a (possibly negative) number of `\internote` (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 systems after the one where it was initiated.

Chapter 10

Font-based Slurs, Ties and Hairpins

Two fundamentally different implementations of slurs, ties, and hairpins are available. First, there are the original *font-based* versions. These are constructed with traditional TeX font characters that were created with METAFONT and stored in TeX font files. Second, it is now possible to generate these shapes directly with Postscript, dispensing altogether with the font characters[1]. We shall first describe the font-based versions, then Type K Postscript slurs, which are one of two available Postscript slur options[2]. If you plan to use Type K Postscript slurs, you may skip directly to Chapter 11.

Font-based slurs and ties provided by MusiXTeX 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 sections 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.

10.1   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[3]. 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.

---

[1]Please do not be confused by the availability of Postscript versions of the font-based slur fonts (along with all other MusiXTeX fonts). Once installed in a TeX 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.

[2]An alternate approach to Postscript slurs, called *Type M* after its developer Hiroaki MORIMOTO, is available from the Icking Music Archive.

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

As with beams, the reference number n by default can take values from 0 to 5, or up to 8 or 11 respectively if `musixadd.tex` or `musixmad.tex` is included. You can also specify the maximum number directly with `\setmaxslurs{m}` where $7 < m \leq 100$[4]; the reference number n will be in the range between 0 and $m - 1$.

To illustrate with an elementary example, the following passage

was coded as:

```
\NOtes\islurd0g\qu g\tslur0{'c}\qu c\en
\Notes\isluru0{'e}\ibl0e{-2}\qb0{edc}\tslur0b\tqb0b\en
\bar
\NOtes\islurd0{'a}\qu a\tslur0{'f}\qu f\en
\NOTes\hu g\en
```

Other macros are provided to change the horizontal starting and ending points 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.

10.2   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
\isluru, \islurd	\tslur
\issluru, \isslurd	\tsslur
\ibsluru	\tbsluru
\ibslurd	\tbslurd

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, or *vice versa*. These would be achieved using macro pairs `\ibslur...\tslur...` or `\islur...\tbslur...`, respectively; however, it may be necessary to adjust the vertical position of the slur termination. For example,

---

[4]This may require e-TeX.

has been coded as follows:

```
\startextract
\Notes\ibsluru1g\qa g\ha l\sk\tslur1e\zq g\zq i\qu k\en
\Notes\islurd1h\qu h\ha f\sk\tbslurd1c\zq h\zq j\ql l\en
\endextract
```

10.3   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 slur[5]. 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`}`[6], and terminated with `\tten{`n`}`.

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

It was coded as

```
\NOTes\islurd0g\qu g\tslur0g\qu g\isslurd0g\qu g\tsslur0g\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\tsslur1h\zh{df}\hu h\en
\NOTes\ibsluru0g\islurd1g\hu g\tubslur0h\hu h\en
\NOTes\itieu0k\hl k\ttie0\tdbslur1f\hl k\en
```

---

[5]Editor's note: In fact, it appears that the default positioning of the ends of ties is exactly the same as that of short slurs.

[6]Editor's note: It is not clear why this command uses "`l`" when all other similar ones use "`d`".

10.4 Dotted slurs

Any font-based slur may be made dotted by specifying \dotted just before it is initiated:
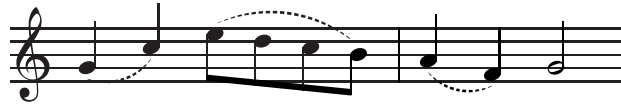
This was coded as:

```
\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
```

10.5 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,
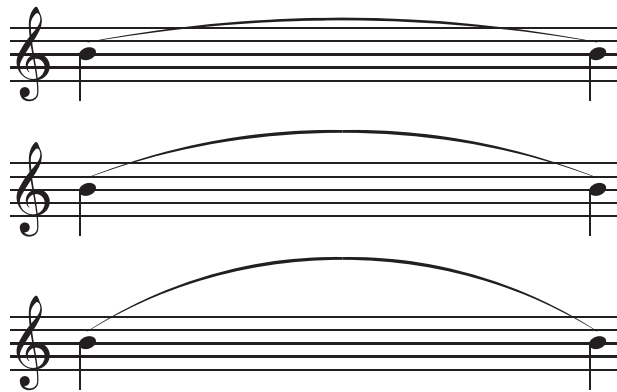
was coded as

```
\NOtes\multnoteskip8\isluru0i\ql i\en
\notes\tslur0i\ql i\en
\NOtes\multnoteskip8\isluru0i\ql i\en
\notes\midslur7\tslur0i\ql i\en
\NOtes\multnoteskip8\isluru2i\ql i\en
\notes\midslur{11}\tslur2i\ql i\en
```

For downward slurs, use negative values; for example,
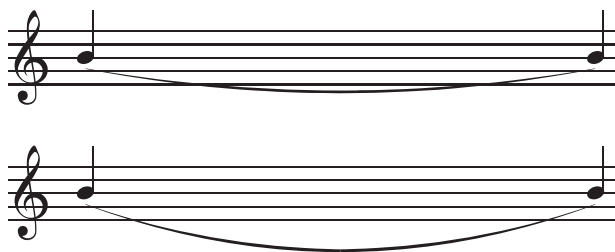
was coded as

```
\NOtes\multnoteskip8\islurd0i\qu i\en
\notes\tslur0i\qu i\en
\NOtes\multnoteskip8\islurd0i\qu i\en
\notes\midslur{-7}\tslur0i\qu i\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.
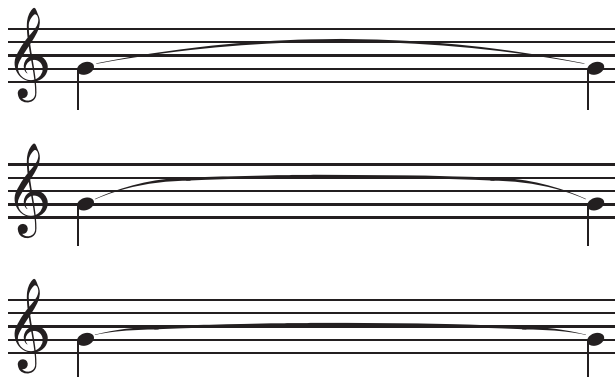
This was coded as

```
\NOtes\multnoteskip8\itieu0g\ql g\en
\notes\ttie0\ql g\en
\NOtes\multnoteskip8\itieu1g\ql g\en
\notes\curve 322\ttie1\ql g\en
\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\internote` relative to its initialization height.

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

Default, without adjustments:

With `\Liftslur` and `\breakslur`:

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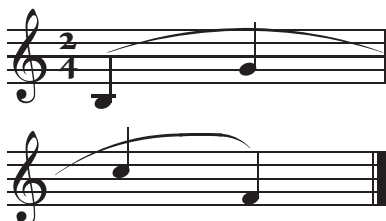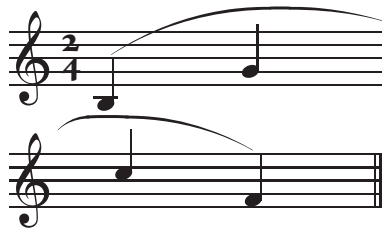
```
\def\atnextline{\Liftslur17}%
\Notes\ibsluru1b\qu b\qu g%
\breakslur1g\en\bar
\Notes\qu{'c!}\tslur1f\qu f\en
```

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.
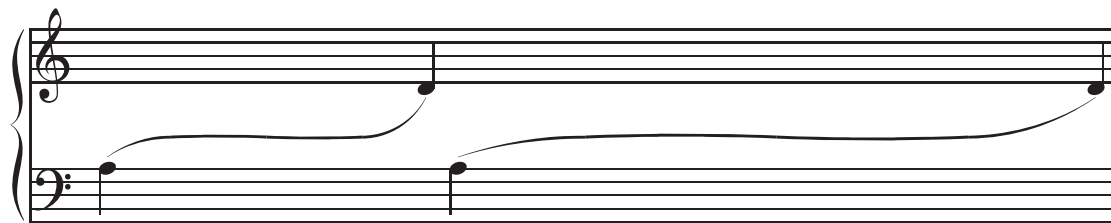
This was coded as

```
\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 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
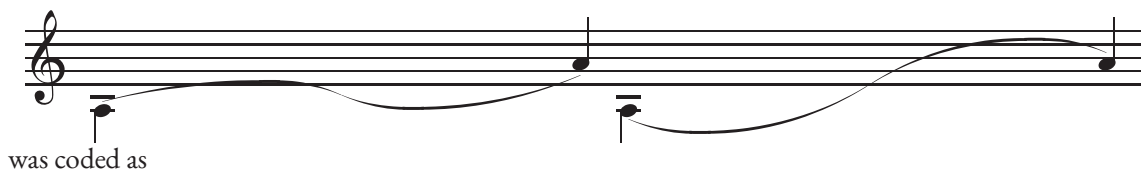
was coded as

```
\begin{music}\nostartrule
\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}
```

10.6   Simple slurs

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{p_1}{p_2}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{p_1}{p_2}sl and \stie{p}sl respectively. Finally, simple beam slurs can be forced with \bslur$s${p_1}{p_2}l.

10.7   Font-based hairpins (crescendos and decrescendos)

There are two categories of font-based hairpins.   The first type requires only one command, `\crescendo{ℓ}` or `\decrescendo{ℓ}`, where $ℓ$ is any TeX 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,
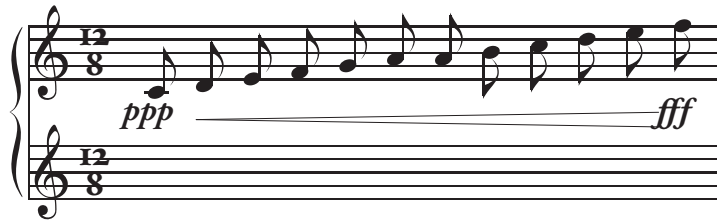
which was coded as

```
\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
```

10.8   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
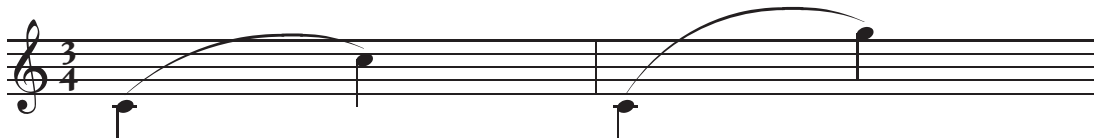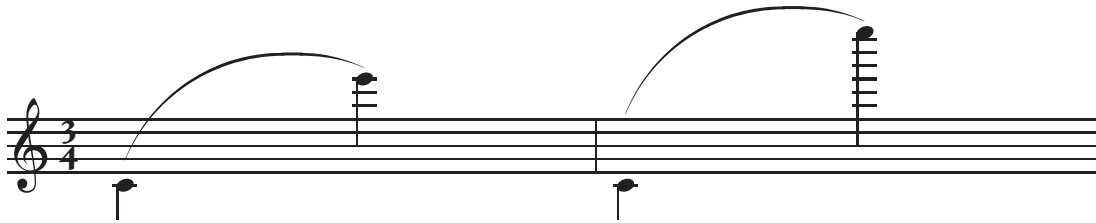
whose coding was

```
\NOTes\multnoteskip3\isluru0c\ql c\tslur0j\ql j\en
  \bar\NOTes\multnoteskip3\isluru0c\ql c\tslur0n\ql n\en
\NOTes\multnoteskip3\isluru0c\ql c\tslur0s\ql s\en
  \bar\NOTes\multnoteskip3\isluru0c\ql c\tslur0z\ql z\en
  \end{music}
```

Furthermore, if the slope becomes too steep, even worse results can occur, such as
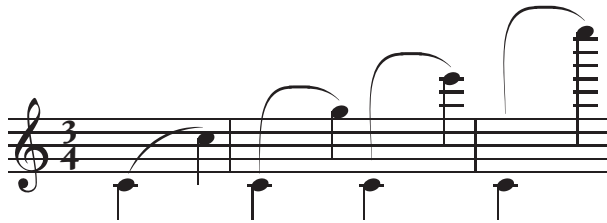
Another limitation of font-based slurs crops up when one attempts to generate bitmapped versions for high-resolution printers: the characters for long ties can exceed METAFONT's maximum capacity.

Chapter 11

Postscript Slurs, Ties and Hairpins

All of the aforementioned limitations of font-based slurs can be circumvented by using type K Postscript slurs[1]. As well as slurs, the package includes ties and crescendos. 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, ties and hairpins, you must first place `musixps.tex` anywhere TeX can find it. You must also place `psslurs.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 installed and the TeX filename database is refreshed, the type K package can be invoked by including the command `\input musixps` near the beginning of your source file (but after loading `musixtex.tex`). The resulting dvi file should then be converted into Postscript using **dvips**. If desired, a PDF file can then be generated with **ps2pdf**, **ghostscript**, or **Adobe Acrobat** (see Section 1.5 for more information on this).

Two minor inconveniences with type K Postscript slurs are that (1) they won't appear in many dvi previewers, and (2) they won't appear in PDF files generated with **pdftex** or **dvipdfm**. 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.

11.1 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.

Termination macros are not restricted to being used with their initiation counterpart. For example, a slur beginning as a "beam" slur may be terminated as a normal slur, or *vice versa*. These would be achieved by using macro pairs `\ibslur...\tslur...` or `\islur...\tbslur...`, respectively. Unlike font-based slurs, adjusting the vertical positioning of the slur termination should not be necessary.

---

[1]"K" stands for Stanislav Kneifl, the developer of the type K Postscript slur package.

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 `\tlslur`. For shifts farther to the left, use `\iLsluru`, `\iLslurd` or `\tLslur`; for shifts farther to the right, use `\iRsluru`, `\iRslurd` or `\tRslur`.

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 `\tfslur0f`; a bit higher, `\thslur0f`; higher still, `\tHslur0f`; or even higher, `\tHHslur0f`. 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}\nostartrule
\input musixps
\startextract\NOTes\multnoteskip3\isluru0c\ql c\tslur0s\ql s\en
....
```


For maximal control over type K slurs, you can use one of the commands `\iSlur` *n p v h* and `\tSlur` *n p v h c a*, where the characters in *n p v h c a* respectively stand for ID number, height, vertical off-set, 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:


```
\irslur...\tlslur      \irslur...\tlfslur      \iSlur...\tSlur
```

with this coding:

```
\NOtes\irslurd0{'b}\zhl b\sk\zq{!e}\qu{'c}\en\bar
\NOtes\zh{'d}\zhu{'f}\off{7.2pt}\tlslur0{'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., \tbsluru{17684}{16}; however \ibsluru{152867}{16} is okay.

It's also okay 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.

'

11.2 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{jl}\tbu0%
 \slurtext{6}\tBslur00\qb0e\en
```

produces

This 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 disappears[2].

11.3 Type K Postscript ties

All of the foregoing Type K slur commands except the shape-changing ones 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. If you want to change the shape of a tie, redefine \pstiehgt from its default of 0.7.

11.4 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

---

[2]If you insist on viewing files with a dvi viewer that doesn't display type K slurs, you may also find that figures placed with \slurtext will appear at the end of the slur rather than the middle.

`\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.

11.5   Avoiding collisions with staff lines.

In Postscript it is possible to do computations that 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.

11.6   Type K Postscript hairpins

There are two different types. The first type is normally initiated either with `\icresc{`*n*`}` or with `\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,
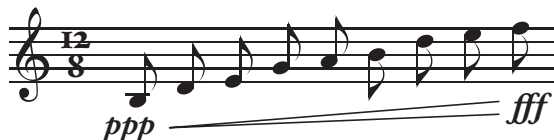
was coded as

```
\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
\zendextract
```

11.7   Line-breaking slurs, ties and hairpins

Tyle K slurs, 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`.

In order 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 $10 \leq \text{ID} < 15$. 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`.

11.8   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 solution is 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`.

Chapter 12

Bar Lines

12.1 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 typeset a double-bar line in the middle of a bar, use `\zdoublebar`; use `\zzdoublebar` to force the mid-bar double-bar line to the end of a system.

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

You can suppress the beginning vertical rule with saying `\nostartrule` and restore the default with `\startrule` after that. Note that it is conventional practice to omit beginning rules for single-staff music.

12.2 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 20.

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 25.21.

12.3 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.
- \showallbarrules resets all defined instruments to \showbarrule{*n*}. This command is automatically inserted with double bars, final bars and repeats.

Thus, this example
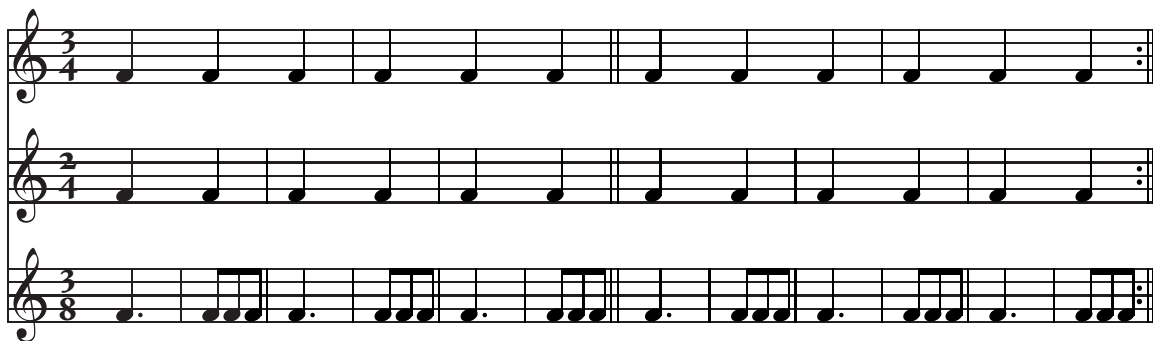
was obtained with the following coding:

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

\startextract
\NOtes\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
\zendextract
```

12.4   Dotted, dashed, asynchronous and discontinuous bar lines

The extension package <u>musixdbr.tex</u> 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\setstaffs23\setstaffs32\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\zendextract
```

12.5   Ledger Lines

Ledger lines are the short horizontal lines above or below the staff for high or low notes. As of version 1.36, these have been thickened. To restore the thickness used in previous versions, add `\def\hlthick{0.2pt}` to the preamble after inputting `musixtex.tex`.

Chapter 13

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`.

MusiXTₑX 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.

13.1   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\Dqbbl hk\en\bar
\Notes\Tqbu ghi\Tqbl mmj\en
\def\fontbarno{\bf}%
\notes\Tqbbu fgj\Tqbbl njh\en\bar
\Notes\Qqbu ghjh\Qqbl jifh\en\bar
```

```
\notes\Qqbbu fgge\Qqbbl jhgi\en
```

13.2   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}
```

Similarly, the horizontal position 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 MusiXTEX macro `\boxit` which will enclose any text string in a box.

Here are some possible alternate formats for system bar numbers:
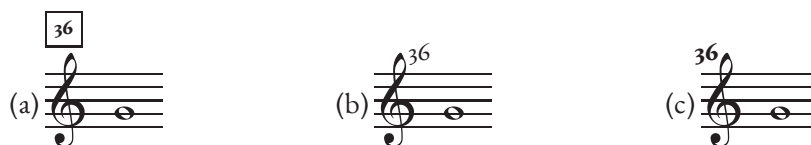
These were coded as

  (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{+}`.

Chapter 14

Managing the Layout of your Score

14.1   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 `6pt` 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`; to force the mid-bar bar line to the end of a system, use `\zzbar`.

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`. If you want to suppress right-justification of the last line of a score, use `\raggedstoppiece`; the bar line can be suppressed by using `\zendpiece` (or `\setemptybar`).

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 12.1. Those commands can also be used before `\stoppiece`, `\alaligne`, or `\alapage`

14.2   Page layout

Blank space above and below each staff is controlled by the dimension registers `\stafftopmarg` and `\staffbotmarg`. For more info see Chapter 21.

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 as much material as possible on a page, provided the printer supports these dimensions which imply approximately half-inch margins:

A4	letter
`\hsize=190mm`	`\hsize=7.5in`
`\vsize=270mm`	`\vsize=10in`

You may have to be more conservative if you have headlines or footlines or want even/odd-page asymmetry, such as with the `\rectoverso` command in `musixsty.tex` (Section 25.32). If a score is going to be "published" at a public repository (such as IMSLP), you may want to allow for *either* A4 *or* letter paper, for example by setting the `\hsize` to 190 mm and the `\vsize` to 10 in.

To generate a score in "landscape" mode, adjust the text-size parameters above accordingly (e.g., `\hsize=10in` and `\vsize=7.5in`) and add a suitable `papersize` "special" to the TEX output, as in `\special{papersize=11in,8.5in}`.

For *any* values of `\hsize` and `\vsize`, material will normally be centered on the page by using command sequences such as the following, which simply split the difference between the page size and the text size, and then subtract one inch because of TEX conventions:

For A4 paper, portrait mode (210 mm × 297 mm):

```
\hoffset=210mm\advance\hoffset-\hsize\divide\hoffset2
\advance\hoffset-1.0in
\voffset=297mm\advance\voffset-\vsize\divide\voffset2
\advance\voffset-1.0in
```

For letter-size paper, portrait mode (8.5 in × 11 in):

```
\hoffset=8.5in\advance\hoffset-\hsize\divide\hoffset2
\advance\hoffset-1.0in
\voffset=11in\advance\voffset-\vsize\divide\voffset2
\advance\voffset-1.0in
```

To determine the maximal dimensions that *your* printer is capable of, process the standard file `testpage.tex` with LATEX and print the result. The "rulers" on four sides will indicate how much of the one-inch "margin" material outside the frame can actually be printed. Your `\hsize` and `\vsize` parameters can be increased, or, if necessary, decreased accordingly. If your score is going to be printed by a publisher or a colleague, you can send `testpage.pdf` to them to determine what margins are necessary for their printer.

However, if the margins on the `testpage` output are *not* symmetric, the printer is mis-aligned. If this can't be corrected, the margin offsets will have to be adjusted. Ideally, this should be done in the TEX configuration so all TEX output will be corrected, but offsets can be adjusted for just `musixtex` output using `\advance\hoffset ...` and/or `\advance\voffset ...` after the "normal" settings given above.

14.3   Page numbering, headers and footers

There are no special page numbering facilities in MusiXTEX; you must rely on macros from plain TEX. There is a count register in TEX 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, TEX 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) TEX 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
```

14.4 Controlling the total number of systems and pages

Once all the notes are entered into a MusiXTEX 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 possible[1]. 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.

---

[1]It *is* possible in **PMX**.

Chapter 15

Changing Clefs, Keys and Meters

15.1   Introduction

To define clefs, key signatures, or meters at the start of a piece, or to change one or more of these attributes[1] anywhere else in a score, MusiXTeX 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`, `\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 eleven 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.

15.2   Key Signatures

We've already seen in Section 2.3 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 eleven 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.

---

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

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\setstaffs22%
\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%
```


15.3   Clefs

Macros that define clefs have already been discussed in Section 2.3. By way of review, here are all of the possible clefs (applied to the lowest staff):

`\setclef1\treble` `\setclef10`	`\setclef11`	`\setclef12`	`\setclef1\alto` `\setclef13`	`\setclef14`
`\setclef15`	`\setclef1\bass` `\setclef16`	`\setclef17`	`\setclefsymbol1\empty`[2] `\setclef18`	`\setclef19`

Just as with key signatures, these commands only *prepare* for clef changes. To *activate* them, any of the first eleven commands listed in Section 15.1 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, whereas, 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...\en` 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\setstaffs22%
\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%
% 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%
\zendextract
```

---

[2]Details of the macro `\setclefsymbol` will be discussed later

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

If `\atnextbar` is being used to center material and new clefs before the bar line should be excluded from the "notes" region within which material is centered, you may use the command `\Changeclefs`, as in the following:
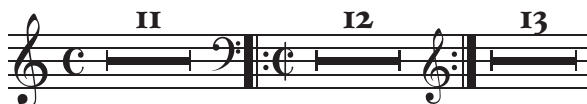
which was coded as

```
\generalmeter{\meterC}
\startextract%
\NOTEs\en
\def\atnextbar{\znotes\centerHpause{11}\en}%
\setclef1\bass\Changeclefs%
\setleftrepeat%
\generalmeter{\allabreve}%
\changecontext%
\NOTEs\en
\def\atnextbar{\znotes\centerHpause{12}\en}%
\setclef1\treble\Changeclefs%
\setrightrepeat\bar%
\NOTEs\en
\def\atnextbar{\znotes\centerHpause{13}\en}%
\endextract
```

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, MusiXTEX 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}&\en
\zendextract
```

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.

Four other small clef symbols are available: \smalltrebleoct, \smalltreblelowoct, \smallbassoct, and \smallbasslowoct. They look just like the corresponding normal-sized symbols, and are useful by clef symbol substitution for clef changes after the beginning of a piece, as demonstrated in the following example.

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 three clefs after (5) in comparison with (2). These also illustrate the use of the small octave clef symbol. Obviously the second clef after (5) is nonsense; \resetclefsymbols puts matters in order at (6) and (7).

This is the code:

```
\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\smalltreblelowoct\changeclefs
\notes\qu i\en\setclef1\bass\changeclefs\bar
\notes\qu I\en\setclef1\treble\changeclefs
\notes\zchar{-5}{6}\qu i\en
\resetclefsymbols\setclef1\bass\changeclefs\bar
\notes\zchar{-5}{7}\qu J\en\setclef1\treble\changeclefs
\notes\qu i\en
\endpiece
```

15.4   Meter changes

As mentioned in Section 2.3.6, 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*`}{{`*m*1`}{`*m*2`}{`*m*3`}{`*m*4`}}`, where *n* is the number of the instrument, *m*1 specifies the meter of the first (lowest) staff, *m*2 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 15.1.

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

```
\instrumentnumber2\setstaffs22%
\generalmeter{\meterfrac{4}{4}\meterfrac{2}{4}\meterfrac{1}{4}}%
\setclef1{\bass}\generalsignature2%
\startextract
\setmeter1{{\meterfrac{2}{4}}}%
\setmeter2{{\lower2pt\hbox{\meterfrac{\meterlargefont 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
```


15.5 Changes at the end of a system

It is conventional music-typesetting practice to indicate changes of clef, time signature, and key signature that apply to the start of a *new* system also at the end of the previous system, as in the following:

Chapter 16

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, MusiXTEX will automatically post a right repeat at the end of the system and a left repeat at the beginning of the next.

For example,
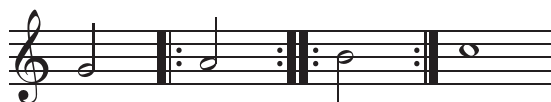
has been coded as:

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

To insert a right-repeat at a forced line break or at the end of a piece, use `\setrightrepeat` *before* `\alaligne` or `\endpiece`. In fact it is possible to use `\setleftrepeat`, `\setrightrepeat` or `\setleftrightrepeat` before any `\bar`, `\stoppiece` or `\changecontext`; but be aware that whereas `\setrightrepeat` behaves properly if the bar is at the end of a system, `\setleftrepeat` and `\setleftrightrepeat` do not, placing the left-repeat symbol at the end of the system rather than at the beginning of the next system; see the `\zleftrepeat` `\zzleftrepeat` `\zzleftrightrepeat` and `\zzleftrightrepeat` commands described below.

The following commands generate repeat symbols as usual, but are assumed to be in the *middle* of a bar; that is, the bar counter is not incremented:

```
\zrightrepeat
```

```
\zleftrepeat
```

```
\zleftrightrepeat
```

For example, to insert a left-repeat at a forced line break or at the beginning of a piece, simply use `\zleftrepeat` immediately *after* `\startpiece` or `\alaligne`. To insert a left-right repeat at a forced line break, use `\setrightrepeat\alaligne\zleftrepeat`.

The following commands are similar but force the mid-bar break to the end of a system:

> `\zzrightrepeat`
>
> `\zzleftrepeat`
>
> `\zzleftrightrepeat`

For `\zzleftrepeat` and `\zzleftrightrepeat`, the left-repeats occur at the beginning of the following system. The bar numbers are correctly adjusted, using `\writezbarno` for the continuation if `\systemnumbers` is in use.

As an example, here is a *bourrée* by G.-F. Handel coded using `\zleftrepeat` at the beginning of the **A** section and `\zzleftrightrepeat` between the **A** and the **B** sections:

No explicit adjustment of `\barno` was necessary. The right-repeat at the end of the **B** section was produced by `\setrightrepeat\endpiece`.

16.1   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:
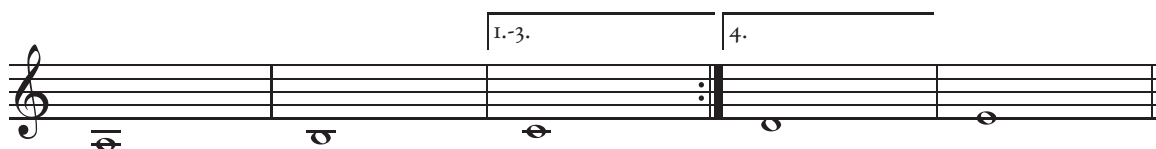
which was coded as:

```
\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 may be removed by using \def\voltadot{}, and restored by \def\voltadot{.}.

16.2 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\en\bar
\NOtes\coda m\en
\NOtes\Segno\en\bar
\NOtes\Coda m\en
```

16.3 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\setstaffs1{2}
\startextract
\NOtes|\qa{cegj}\en
\def\atnextbar{\znotes\centerpause|\en}\bar
\NOtes\qa{cdef}|\en
\def\atnextbar{\znotes|\centerbar{\duevolte}\en}%
\endextract
```

Chapter 17

Font Selection and Text Placement

17.1   Predefined text fonts

While any font with support for TEX can be used by MusiXTEX, certain styles and sizes can be selected using shortcut commands. For ordinary text, the shortcuts cover fonts of nine different sizes and six styles. The sizes in points are 7, 8, 9, 10, 12, 14, 17, 20, and 25; the styles are from the standard Computer Modern family: Roman, bold, italic, bold italic and small capitals. The size selection macros from smallest to biggest are `\tinytype`, `\smalltype`, `\Smalltype`, `\normtype`, `\medtype`, `\bigtype`, `\Bigtype`, `\BIgtype` and `\BIGtype`. The style may be selected or changed using `\rm` (Roman), `\bf` (bold), `\it` (italic), `\bi` (bold italic) or `\sc` (small-capitals). If no style is explicitly selected, Roman style will be used for the sizes `\medtype` or smaller; for the larger sizes, bold style is the default. 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`, `\bf`, `\bi` or `\sc`, as in Plain TEX. When MusiXTEX 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 8, 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 `\tinydyn`, `\smalldyn`, `\normdyn`, or `\meddyn` may be used to redefine `\ppff` to represent one of the smallest four.

A selection of predefined fonts is summarized in Table 17.1. The second column gives an explicit control sequence that can alternatively be used locally as a font specification.

17.2   User-defined text fonts

Since MusiXTEX is a superset of TEX, you are free to use the standard TEX 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 TEX file-name database has been updated, as required by your TEX system.   Then you can use the font just as in any TEX 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 question. You can use bitmapped fonts, which are converted to Postscript by e.g., `dvips`, but

Size and style	Font specification	Example
\tinytype	\sevenrm	tiny Roman
\tinytype\bf	\sevenbf	**tiny bold**
\tinytype\it	\sevenit	*tiny italic*
\smalltype	\eightrm	small Roman
\smalltype\bf	\eightbf	**small bold**
\smalltype\it	\eightit	*small italic*
\Smalltype	\ninerm	Small Roman
\Smalltype\bf	\ninebf	**Small bold**
\Smalltype\it	\nineit	*Small italic*
\normtype	\tenrm	normal Roman
\normtype\bf	\tenbf	**normal bold**
\normtype\it	\tenit	*normal italic*
\normtype\bi	\tenbi	***normal bold italic***
\normtype\sc	\tensc	NORMAL SMALL CAPITALS
\medtype	\twelverm	medium Roman
\medtype\bf	\twelvebf	**medium bold**
\medtype\it	\twelveit	*medium italic*
\medtype\bi	\twelvebi	***medium bold italic***
\medtype\sc	\twelvesc	MEDIUM SMALL CAPITALS
\bigtype	\frtbf	**big bold**
\Bigtype	\svtbf	**Big bold**
\BIgtype	\twtybf	**BIg bold**
\BIGtype	\twfvbf	**BIG bold**
	\ppfftwelve	***pp ff diminuendo***
	\ppffsixteen	***pp ff diminuendo***
	\ppfftwenty	***pp ff diminuendo***
	\ppfftwentyfour	***pp ff diminuendo***
	\ppfftwentynine	***pp ff diminuendo***

Table 17.1: Various predefined fonts

you also may replace them by native Postscript fonts.

The extension library `musixtmr.tex`[1] replaces the default Computer Modern text fonts by the Times series of fonts; see Section 25.34. Other extension libraries, `musixplt.tex`, `musixebgr.tex`, `musixhv.tex` and `musixcls.tex`, replace the default text fonts by Palatino, EBGaramond, Helvetica and Classico fonts, respectively; see Sections 25.27, 25.12, 25.19 and 25.6, respectively. Here is a comparison of some Times, Palatino, EBGaramond, Helvetica and Classico fonts:

Times	Palatino	EBGaramond	Helvetica	Classico
Roman	Roman	Roman	Roman	Roman
bold	**bold**	**bold**	**bold**	**bold**
italic	*italic*	*italic*	*italic*	*italic*
big	BIG	BIG	big	BIG
Big	BIG	BIG	Big	BIG

A "handwriting" font (especially suitable for jazz scores) is supported by the extension library `musixjt.tex`; see Section 25.20.

For users who prefer to stick with the default Computer Modern family but want to use the T1-encoded EC variants, the extension library `musixec.tex` is available; see Section 25.11.

17.3 Text placement

Special macros are provided to allow precise placement of any TeX 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...\en`. 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. The following abbreviations are available:

```
\zcn   for   \zcharnote
\lcn   for   \lcharnote
\ccn   for   \ccharnote
```

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 TeX 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

---

[1]by Hiroaki Morimoto

system to accommodate any text placed with these macros. On the other hand, if text is placed far above or below a system, MusiXTEX will usually insert additional vertical space if needed.

17.4   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 TEX dimension if the default value of `3pt` is unsatisfactory. To place the mark, use `\Uptext` or any of the other macros defined in the previous section.

Chapter 18

Miscellaneous Other Notations

18.1 Metronomic indications

By way of example, the notations

$$\text{♩} = 60 \quad \text{and} \quad \text{♩.} = \text{♩}$$

are respectively coded as \metron{\qu}{60} and \metronequiv{\qup}{\qu}, which are normally placed using \Uptext. Also, you can set a smaller metronomic indication like

$$\text{♩} = \text{ca. 72} \quad \text{or} \quad \text{♩} = \text{ca. 72}$$

by writing

```
\metron{\smallnotesize\lqu}{ca.~72}
```

or

```
\metron{\tinynotesize\lqu}{ca.~72}
```

18.2 Accents

You may use[1]
- \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,

---

[1]Note from the editor: the reason the names of some of these macros don't seem to suggest the terms used in the descriptions is that apparently Daniel Taupin had in mind other terms, such as *pizzicato*.

- \upzst{*p*} (upper *mezzo-staccato/portato*) to put a dot and a hyphen above a note head at pitch *p*,
- \lpzst{*p*} (lower *mezzo-staccato/portato*) 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 may use

> \wholeshift{*Any nonspacing macro*}

to center accents (and other items) above or below a whole note. For example, \Fermataup{*p*} (described in Section 18.4) is *defined* as \wholeshift{\fermataup{*p*}}.

There are also variants of the most common accents[2] 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
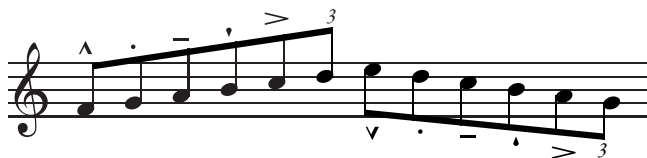
was coded as

```
  \startextract
\Notes\ibu0f3\busfz0\qb0f\bupz0\qb0g\bust0\qb0h%
  \buppz0\qb0i\busf0\qb0j\butext0\tqh0k\en
\Notes\Ibl0lg5\blsfz0\qb0l\blpz0\qb0k\blst0\qb0j%
  \blppz0\qb0i\blsf0\qb0h\bltext0\tqb0g\en
\zendextract
```

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

18.3   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 which will be placed by the first and third through sixth macros, and is initially defined as \def\txt{3}. The macro \tuplettxt serves the same role for the last two macros. The font used is defined by \txtfont and is, by default, \eightit (in normal music size) but may be re-defined.

The first four commands in the table are normally used with beamed xtuplets. As indicated in the last column, the last four produce a sloping bracket and may be used with unbeamed xtuplets.

---

[2]Thanks to Klaus Bᴇᴄʜᴇʀᴛ's corrections.

Macro and arguments	Number printed	Where invoked	Needs Bracket
`\triolet{`*p*`}`	`\txt`	before beam	none
`\xtuplet{`*k*`}{`*p*`}`	*k*	before beam	none
`\butext{`*n*`}`	`\txt`	before note at number	none
`\bltext{`*n*`}`	`\txt`	before note at number	none
`\uptrio{`*p*`}{`*w*`}{`*s*`}`	`\txt`	before first note	solid
`\downtrio{`*p*`}{`*w*`}{`*s*`}`	`\txt`	before first note	solid
`\uptuplet{`*p*`}{`*w*`}{`*s*`}`	`\tuplettxt`	before first note	with gap
`\downtuplet{`*p*`}{`*w*`}{`*s*`}`	`\tuplettxt`	before first note	with gap

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

whose coding is

```
\notesp\xtuplet6n\isluru0l\ibl0l0\qb0{lllll}\tslur0l\tqb0l\en\bar
\notesp\triolet n\isluru0l\Ibl0ln2\qb0{lm}\tslur0n\tqb0n\en
\notesp\ibslurd0k\Ibl0km2\qb0k\bltext0\qb0l\tdbslur0m\tqb0m\en\bar
\Notesp\triolet o\isluru0l\ql{lm}\tslur0n\ql n\en\bar
\Notesp\uptrio o16\ql l\en\notesp\cl n\en
\Notesp\downtrio O16\qu e\en\notesp\cu g\en
```

Next are examples using `\uptuplet` and `\downtuplet`. A re-definition of `\tuplettxt` inside a notes group only applies inside that group and leaves the default definition intact. To get the number properly centered in the gap, you must include some extra space after the number in the definition of `\tuplettxt`. 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

```
\parindent0mm
\generalmeter{\meterfrac24}
\startextract
\addspace\afterruleskip
\notesp\triolet o\Ibl0ln2\qb0{lm}\tqb0n\en
\notesp\Ibl0km2\qb0k\bltext0\qb0l\tqb0m\en\bar
\notesp\def\tuplettxt{5\/\/}\uptuplet o{4.1}2\cl{jklmn}\en\bar
\Notesp\uptuplet o16\ql l\en\notesp\cl n\en
\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
```

A modern notation uses a *ratio* such as 5:6 to specify the intended duration of an x-tuplet. This is supported in MusiXTEX by the command `\xxtuplet`, as in

with coding `\xxtuplet5:6o\ibl0j3\qb0{jklm}\tql0n`. For the other x-tuplet commands (apart from `\triolet`, which is defined in terms of `\xtuplet`), it is possible to define `\txt` or `\tupletxt` as a ratio, as in

    \def\txt{5:6}

18.4   Ornaments

18.4.1   Simple 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
- `\Shakel{`*p*`}` for
- `\Shakesw{`*p*`}` for
- `\Shakene{`*p*`}` for
- `\Shakenw{`*p*`}` for
- `\turn{`*p*`}` for
- `\backturn{`*p*`}` for
- `\tr{`*p*`}` for *tr*
- `\trt{`*p*`}` for   (J. S. Bach's *trillo und mordant*)
- `\xtr{`*p*`}` for ×
- `\ptr{`*p*`}` for +

To compensate for the fact that whole notes are wider, you may use

    \wholeshift{*Any nonspacing macro*}

to center any of these ornaments above (or below) a whole note.

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:

18.4.2 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.

18.4.3 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 ~~~ or \Trille{*p*}{*l*} for *tr* ~~~~~ , where *p* is the pitch and *l* the length in current \noteskips.

To let MusiXTEX compute the length of the trill, or if it extends across a system break, you can use \Itrille*n*{*p*} to start a plain trill, where *n* is a trill reference number between 0 and 5; then \Ttrille*n* to terminate it. To get the *tr* at the beginning, use \ITrille*n*{*p*} to start the trill and \Ttrille*n* to close it.

As with other elements, you can specify the maximum number directly with \setmaxtrills{*m*}, where $7 < m \leq 100$[3]; the reference number *n* will be in the range between 0 and $m - 1$.

For example:
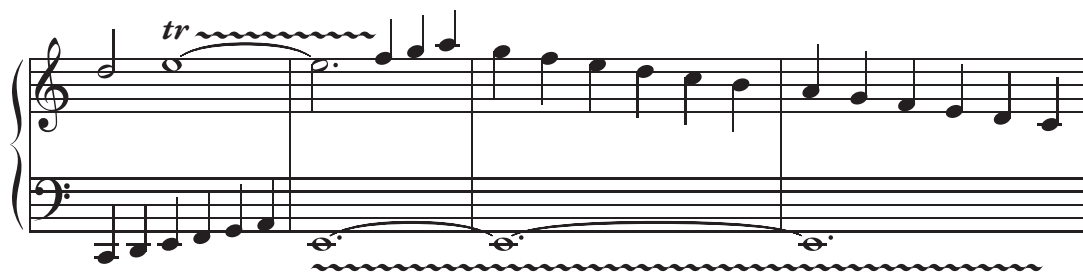
whose coding is

```
\begin{music}\nostartrule
\instrumentnumber{1}
\setstaffs12
\setclef1{6000}
%
\startextract
```

---

[3]This may require e-TEX.

```
\notes\qu{CDEFGH}|\hu k\sk\ITrille 1p\itenu1l\wh l\en
\bar
\notes\Itrille 2A\itenu1e\whp E|\tten1\hlp l\sk\Ttrille 1\qu {mno}\en
\bar
\Notes\tten1\itenu1E\whp E|\ql{nmlkji}\Toctfin1\en
\bar
\Notes\tten1\whp E\sk\sk\sk\sk\Ttrille2|\qu{hgfedc}\en
\endextract
```

18.4.4 Tremolos

The command $\texttt{\textbackslash trml\{}p\texttt{\}}$ is used to indicate that the immediately following quarter or half note (normally at pitch p and with a lower stem) is to be repeated in eighth notes. For example, $\texttt{\textbackslash trml j\textbackslash ha j}$ generates

and is an abbreviation for

Similarly, $\texttt{\textbackslash trmu\{}p\texttt{\}}$ generates an eighth-note tremolo on a quarter or half note with an *upper* stem:

16th-note and 32-note tremolos are similarly generated by commands $\texttt{\textbackslash trrml}$, $\texttt{\textbackslash trrmu}$, $\texttt{\textbackslash trrrml}$ and $\texttt{\textbackslash trrrmu}$, as in

which is coded as

```
\NOtesp\trrml h\zqlp h\trrmu j\qup j\en
\Notes\stemlength{6.4}\trrml j\zcl j\trrmu l\cu l\en
\NOtes\stemlength{6.4}\trrrml h\zql h\trrrmu j\qu j\en
```

The $\texttt{\textbackslash stemlength}$ command is used to adjust stem lengths; the new length remains effective to the end of the current notes group. If necessary, the usual stem length can be restored by using $\texttt{\textbackslash stemlength\{\textbackslash DefaultStemlength\}}$.

For *whole* notes, tremolo indications should be wider and centered directly over or under the note. These are generated by commands $\texttt{\textbackslash Trml}$, $\texttt{\textbackslash Trmu}$, $\texttt{\textbackslash Trrml}$, $\texttt{\textbackslash Trrmu}$, $\texttt{\textbackslash Trrrml}$ and $\texttt{\textbackslash Trrrmu}$; for example,

which is coded as

```
\NOTEs\Trml h\zw h\Trmu j\wh j\en\bar
\NOTEs\Trrml h\zw h\Trrmu j\wh j\en\bar
\NOTEs\Trrrml h\zw h\Trrrmu j\wh j\en
```

For all of the commands described in this section, the pitch parameter may be used if it is necessary to raise or lower the tremolo indication.

18.5   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,
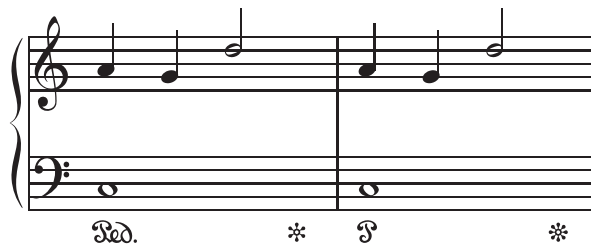
was coded as

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

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 ℗. symbol is rather wide, it might collide with adjacent bass notes. To shift it horizontally, you could use \loff{\PED}.

The following excerpt from *Liebesträume* by Franz Liszt illustrates a more modern approach to specifying piano-pedal usage:

This has been coded using the following commands:

`\pdld`	pedal down	⌞
`\pdlu`	pedal up	⌟
`\pdlud`	pedal up and immediately down	⋀

The elevation is determined by the value of `\raiseped`. Here is how the excerpt may be coded:

```
\smallmusicsize\relativeaccid\nobarnumbers
\generalmeter{\meterfrac64}
\generalsignature{-4}
\setstaffs12\setclef1{\bass}
\def\raiseped{-11}
\def\qpl{\raise-2\Interligne\qp}
\def\qpu{\raise2\Interligne\qp}
\def\hppl{\lifthpausep{-2}}
\startpiece\addspace\afterruleskip%
\NOtes\zqu L\qpl|\qp\en
\bar%
\Notes\loff{\pdld}\ql H\sk\qpl\sk\qpl|%
  \zhl{.c}\ds\ibu1i0\qb1{ehjh}\tqu1e\en
\Notes\zhu{.c}\hppl|%
  \ds\ibu1i0\qb1{ehjh}\tqu1e\en
\bar%
\Notes\pdlud\ql G\sk\qpl\sk\qpl|%
  \itied1c\zhl{.c}\ds\ibu1i0\qb1{^eiji}\tqu1e\en
\Notes\hppl\sk\sk\sk\qu c|%
  \tltie1\zhl{c}\ds\ibu1i0\qb1{eiji}\tqu1e\en
\stoppiece%
\def\raiseped{-15}
\contpiece%
\Notes\pdlud\ql F\sk\qpl\sk\zqu c\qpl|%
  \zhl{.c}\ds\ibu1i0\qb1{=e^hjh}\tqu1e\en
\Notes\hppl\sk\sk\sk\qu c|%
  \zhl{d}\ds\ibu1i0\qb1{ehjh}\tqu1e\en
\bar%
\Notes\pdlud\ql B\sk\qpl\sk\qpl|%
  \zhl{c}\ds\ibu1i0\qb1{^d^hjh}\tqu1d\en
\Notes\zqu{M}\hppl\sk\sk\qpu\sk\qu M|%
```

```
   \ds\ibu1i0\qb1{dhjh}\tqu1d\en
\bar%
\Notes\zmidstaff{$\cdots$}\sk\pdlu\sk|\en
\endpiece
```

The following "mixed" patterns are also supported:

```
\PED ... [\pdlud].. ... \pdlu

\sPED ... [\pdlud].. ... \pdlu
```

18.6   Alphabetic dynamic marks

Conventional dynamic symbols *pppp*, *ppp*, *pp*, *p*, *mp*, *mf*, *f*, *fp*, *sf*, *ff*, *fff*, *ffff*, *sfz* and *sfzp* can be posted using the macros \pppp, \ppp, \pp, \p, \mp, \mf, \f, \fp, \sF, \ff, \fff, \ffff, \sfz and \sfzp, respectively, as the second argument of \zcn (\zcharnote) or \ccn (\ccharnote). See also Section 25.29 for an alternative set of dynamic marks.

18.7   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.

18.8   Brackets, parentheses, oblique lines and slides

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 can be used to enclose notes or to build "big" cautionary accidentals. For example,

  is coded as

```
\NOtes\lpar g\rpar g\hu g\sk%
\loffset{1.5}{\lpar g\rpar g}\loffset{.4}{\sh g}\hu g\sk%
\loffset{2.1}{\lpar g}\loffset{1.5}{\rpar g}\loffset{.4}{\dfl 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 1.125 degrees.

- \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-width 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.

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

For example,
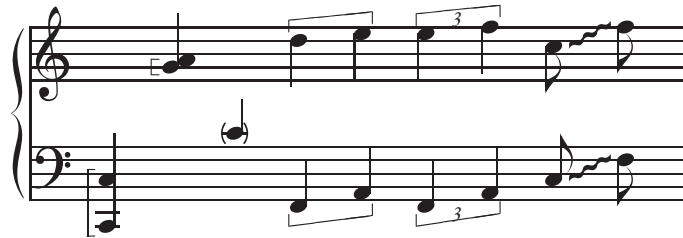
is coded as

```
\begin{music}
\setstaffs1{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{kl}\en
\NOtes\downtrio C16\qu {FH}|\uptrio o14\ql{lm}\en
\NOtesp\roffset{1.5}{\slide J33}\ca{JM}|\roffset{1.5}{\slide j33}\ca{jm}\en
\zendextract
\end{music}
```

18.9   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 MusiXTeX 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 15.3. 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
```

18.10   Tablature

Tablature is a notational system used for fretted stringed instruments such as lute, guitar, banjo, mandolin, and so on. Staff lines represent strings and conventional note heads are replaced by numbers (or sometimes letters) indicating which fret is to be stopped, with 0 representing an open string.

A special clef symbol `\tabclef` may be used with `\setclefsymbol` for a tablature staff; commands `\setlines`, `\setsign`, `\setsize` and `\setname` may be used as usual for a tablature staff. A size-dependent font `\tabstringfnt` can be used for string names.

The macros that put numbers (or letters) on staff lines are as follows:

- `\tab{`n`}{`p`}` puts p on string n (counting from top to bottom); p is centered, using `\ccharnote`. Space for a "rest" can be generated by a command like `\qa{*}`. Ledger lines are replaced by slashes (/) for n greater than the number of staff lines, or simply the number of courses below the staff if this is greater than 3.

- `\ltab{`n`}{`p`}` uses `\zcharnote` so p spills to the right and `\rtab{`n`}{`p`}` uses `\lcharnote` so p spills to the left.

- `\ztab{`n`}{`p`}` is a non-spacing `\tab`, and can be used for building chords.

- `\zltab{`n`}{`p`}` is a non-spacing `\ltab` and `\zrtab{`n`}{`p`}` is a non-spacing `\rtab`.

A size-dependent font `\tabfnt` is used for tab characters. Helvetica bold is the default font family but the traditional script letters of French tablature may be obtained by inputting the extension library `musixftab.tex` described in Section .

The default tablature style is to place fingering notations *on* staff lines; Postscript specials are used to "blank" staff lines below the tab characters. Here is an example for baroque lute (from *Ciaccona* in D minor by J.S. Bach):

The coding is

```
\instrumentnumber{2}
\songbottom1\songtop2
\generalmeter{\meterfrac34}
\interinstrument=4\internote
\nobarnumbers
\setlines16\setclefsymbol1\tabclef\setsize1{1.2}\setsign10
\setsign2{-1}\setclefsymbol2\treblelowoct
\parindent=2ex
\setname1{\tabstringfnt\baselineskip=1.2\internote%
\vbox{\hbox{f}\hbox{d}\hbox{a}\hbox{F}\hbox{D}\hbox{A}}}
\startextract
\NOtes\qa{*}&\qp\en
\NOtesp\zltab50\zltab40\ltab30&\zhl{d}\zhu{f}\qu{.h}\en
\Notes\ltab30&\cu{h}\en
\bar%
\NOtes\zltab50\zltab42\zltab31\ltab22&\zql d\zqu g\zq i\qu l\en
\NOtesp\zltab64\zltab55\zltab30\ltab22&\rhl{^c}\rhu{g}\zh{h}\qu{.l}\en
\Notes\ltab22&\cu{l}\en
\bar%
\NOtes\zltab50\zltab40\zltab30\ltab10&\zql d\zq f\zq h\qu{m}\en
\NOtes\zltab61\zltab40\ltab20&\zhl b\zh f\qu{.k}\en
\Notes\ltab33&\cu j\en
\bar%
\NOtes\zltab70\zltab55\ltab45&\zql N\zq g\qu i\en
\NOtes\zltab60\zltab53\ltab44&\zql a\zq f\qu h\en
\notes\zltab64\ltab42\ilsluru03\ltab53\ltab52\tlslur03\ltab53%
&\zql{^c}\ibbu0g{-1}\qb0g\slur eed2\qb0{fe}\tqu0f\en
\bar%
\notes\ztab{10}0\ltab42\ilsluru03\ltab52\ltab53\tlslur03\ltab50%
&\zql K\ibbu0g{-1}\qb0g\slur edd2\qb0e\qb0f\tqu0d\en
\NOtesp\zltab50\zltab40\ltab30&\zhl{d}\zhu{f}\qu{.h}\en
\Notes\ltab30&\cu{h}\en
\bar\zendextract
```

If it is desired that fingering notations be placed *above* staff lines, use the command

    \tabstylespacetrue.

Here is the same example with `\tabstylespacetrue`:

In pure tablature, note durations can be specified by tiny notes above the staff, as in:

A tiny note can be generated above the staff using one of the commands `\tzwh`, `\tzwhp`, `\tzhu`, `\tzhup`, `\tzqu`, `\tzqup`, `\tzcu`, `\tzcup`, `\tzccu`, … *before* a spacing command. These commands suppress spurious ledger lines and avoid raising dots on dotted notes.

Coding for the notes in the example above is as follows:

```
\startextract
\NOtes\tzqu\qa{*}\en
\NOtesp\zltab50\zltab40\tzqup\ltab30\en
\Notes\tzcu\ltab30\en
\bar%
\NOtes\zltab50\zltab42\zltab31\tzqu\ltab22\en
\NOtesp\zltab64\zltab35\zltab30\tzqup\ltab22\en
\Notes\tzcu\ltab22\en
\bar%
\NOtes\zltab50\zltab30\zltab20\tzqu\ltab10\en
\NOtes\zltab61\zltab40\tzqup\ltab20\en
\Notes\tzcu\ltab33\en
\bar%
\NOtes\zltab70\zltab55\tzqu\ltab45\en
\NOtes\zltab60\zltab53\ltab44\en
\notes\zltab64\tzccu\ltab42\ilsluru03\ltab53\ltab52\tlslur03\ltab53\en
\bar%
\notes\ztab{10}0\tzccu\ltab42\ilsluru03\ltab52\ltab53\tlslur03\ltab50\en
\NOtesp\zltab50\zltab40\tzqup\ltab30\en
\Notes\tzcu\ltab30\en
\bar%
\zendextract
```

Tablature support in MusiXTeX has been adapted from `tabdefs.tex` by R.J. Gelten, which is available [here](#) at WIMA. See extension libraries in Section 25.16 and 25.25 for "french" and "modern" variants, respectively.

Chapter 19

Smaller (or Larger) Notes in Normal-Sized Staves

Here we describe how to change the size of note symbols without changing the size of the staff itself. Changing overall staff size will be treated in Chapter 20.

19.1   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),
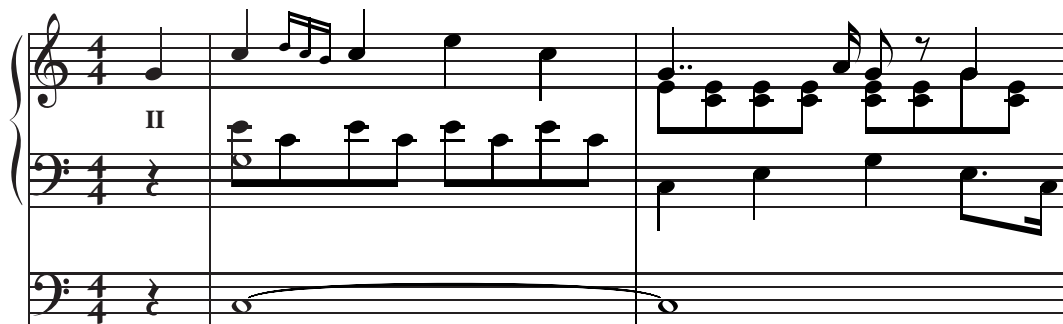
can be coded as

```
\instrumentnumber{2}
\generalmeter{\meterfrac44}
\setstaffs2{2}
\setclef2{\bass}
\setclef1{\bass}
\startbarno=0
\startextract
\NOtes\qp&\zmidstaff{\bf II}\qp|\qu g\en
```

97

```
% mesure 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
% mesure 2
\bar\Notes\ttie2\wh J&\ql J\sk\ql L|\zqupp g\qbl1e0%
  \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\tbbl0\tbl0\qb0J|\tbl1\zq c\qb1e\en
\endextract
```

Similarly, you may use `\largenotesize` or `\Largenotesize` to get *larger* notes.

To support *appoggiaturas* and grace notes (discussed in Section 19.2) in a tiny-notesize staff, the command `\verytinynotesize` may be used; for example,

can be coded as

```
\instrumentnumber2
\setstaffs12
\setstaffs21
\setsize2\tinyvalue
\setclef1{\bass\treble}
\setclef2\treble
\startextract
\notes|\tinynotesize\cu f&\verytinynotesize\cu h\en
\NOtes\qa{J}|\qa{e}&\qa{g}\en
\endextract
```

19.2 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\en
\notes\multnoteskip\smallvalue\smallnotesize\grcu j\en
\NOTes\hu i\en
\bar
\notes\multnoteskip\tinyvalue\tinynotesize\zq h\grcl j\en
\NOTEs\wh i\en
\zendextract
```

19.3 Ossia[1]

This clever example had been provided by Olivier Vogel:

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\ibbbl2{'c}0\qb2b\qb2c\qb2d\tbbbl2\qb2c\en%
\notes\xnum{1.15}{'e}3\qb2d\qb2c\tbl2\qb2d\en%
\notes\ibl2{'c}1\usf e\qb2c\en%
\notes\tbl2\qb2{'d}\en%
\zstoppiece%
}\hss}
\normalmusicsize\nopagenumbers
```

---

[1]Italian *o sia* (or else)

```
\def\nbinstruments{1}%
\setstaffs12\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\tbl1\qb1F|%
\ibbl2{'b}0\qb2b\qb2a\qb2b\tbl2\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%
```

Chapter 20

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 2.3 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). MusiXTEX 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
\setstaffs1{2}
\startpiece\addspace\afterruleskip
%Takt 9
\notes\zhl c\liftqs6\qupp e|\ds&\oct
  \itieu5h\hl h&\tx ~~~gra---*\itied4h\hu h\en
\notes|\ibbl0j3\qb0h\tqb0l\en
\notes|\ibbl1k0\qb1{ohl}\tqb1o\en
\notes\zhl c\liftqs6\qupp e|\ds&\oct
  \ttie5\ibl4c0\qb4h&\ttie4\ibu5g{-3}\qb5h\en
\notes|\ibbl0j3\qb0h\tqb0l&\oct\qb4a&\tx ---*\tqh5a\en
\notes|\ibbl1k0\qb1o\qb1h&\oct\qb4b&\tx ~~ti~-*\cu b\en
\notes|\qb1l\tqb1o&\oct\tqb4c&\tx a*\cu c\en
\bar
%Takt 10
\notes\zhl c\liftqs6\qupp d|\ds&\oct
  \qlp d&\tx ~~~ple---*\ibsluru4e\qup d\en
\notes|\ibbu1g3\bigaccid\qb1{^f}\tqh1h\en
\notes|\ibbu2i0\qb2k\qb2f\en
\notes|\qb2h\tqh2k&\oct\cl e&\curve222\tubslur4f\cu e\en
\notes\zhl c\liftqs6\qupp d|\ds&\oct\ql d&\tx na,*\qu d\en
\notes|\ibbu1g3\qb1f\tqh1h\en
\notes|\ibbu2i0\qb2{kfh}\tqh2k&\qp&\qp\en
\endpiece
```

Chapter 21

Layout Parameters

Most layout parameters are set by MusiXTₑX 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.

21.1   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*. Note that each distinct instrument may have a different `\internote` (see Chapter 20).

`\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`. Default is 9. `\interstaff` applies to all the instruments; to set the inter-staff spacing for a single instrument, use `\setinterstaff{`*n*`}{`*m*`}`, *n* is the instrument and *m* is the replacement factor for that instrument. Note that after you have used `\setinterstaff`, you cannot reset the distances for that instrument with `\interstaff`; you must subsequently use `\setinterstaff` for that purpose.

`\interportee :` distance between the bottom of one staff and the bottom of the next one. It is always reset to 2 times `\internote` times the `\interfacteur` for the current instrument at the next system. Therefore, trying to change `\interportee` will have no effect. Change `\interstaff` or

use `\setinterstaff` instead.

`\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 `\interportee`+`\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 TEX, 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.

21.2   A convenient macro for changing layout parameters

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`.

21.3   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.

21.4   Resetting normal layout parameters

The general size can only be changed by using 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.

21.5   Split and Shortened Systems

It is sometimes desirable to typeset a "shortened" system or to split a system into left and right segments, as in the following (extracted from a Geminiani *concerto grosso*):

This was coded as follows:

```
\setclef1\bass\setclef5\treble\settrebleclefsymbol5\trebleoct
\generalsignature{-2}
\instrumentnumber5\songbottom{1}\songtop{5}
\nobarnumbers
\generalmeter{\meterfrac38}
\hbox to\hsize\bgroup\vbox\bgroup\hsize=1.7in % specify length of left segment
\startpiece\addspace\afterruleskip
\Notes\qa G\sk&\ibu1i{-1}\qb1i\qb1{.g}
  &\ca d\qa g&\ca n\itieu3n\qa n&\ca g\ds\en
\notes\ds\sk&\sk\tbbu1\tqu1g
  &\sk\sk&\sk\sk\ttie3&\ds\sk\en
\znotes&&&&\lcn{q}{\bf D.S. al\kern4ex}\coda p\en
\setdoublebar\endpiece
```

```
\egroup\hfill\vbox\bgroup\hsize=3.7in         % specify length of right segment
\generalmeter{\allabreve}
\startpiece\addspace\afterruleskip
\znotes&&&&\coda o\en
\NOtes\qa{.G}&\qa{.k}&\qa{.d}&\qa n&\qa g\en
\notes\sk\sk\qs\cca N&\sk\sk\qs\cca i
  &\sk\sk\qs\cca k
  &\qs\ibbl3p{-2}\qb3{p}\qb3{o}\tql3n
  &\qp\sk\sk\sk\en
\NOtes\qa{.M}&\qa{.h}&\qa{.k}&\qa o&\ha k\en
\Notes\sk&\sk&\sk&\itieu3r\qa r&\sk\en
\notes\qs\cca M&\qs\cca h&\qs\cca k
  &\sk\sk&\sk\sk\en
\bar
\NOtes\wh L\sk&\qa{g}\qa{g}&\ha{l}\sk
  &\ttie3\ha r\sk&\ha{g}\sk\en
\NOTes\sk&\ha j&\ha{n}&\tr s\ha q&\ha{n}\en
\bar
\nspace
\NOtesp\ha{.K}&\ha{.h}&\ha{.k}&\ha{.r}&\ha{.^m}\en
\setdoublebar\endpiece
\egroup\egroup                                % close open \bgroups
```

Chapter 22

Extracting Parts from a Multi-Instrument Score

From the typesetting viewpoint, the major characteristic of orchestral and chamber music is that the scores not only have several instruments, but variants of the same basic score are needed in which one of the instruments is highlighted while some others are typeset in smaller notes or are omitted. Of course, it must be possible to permute the highlighted instruments and the small-type instruments or omitted instruments, depending on the instrument to which the version of the score is dedicated.

The following is the most general approach to part extraction when using MusiXTEX without **PMX** or **M-Tx** pre-processors. A simpler approach which will work for most purposes is described in Section 25.35.

22.1   Coding rules

To make a "selectable" orchestral score you must prepare the master score (i.e., the score with all instruments typed) as follows:

1. Avoid referring to instrument numbers with roman numerals. For example use `\setclefs` *n* instead of `\cleftoksiii=`.

2. Introduce symbolic names for the instruments; for example, define:
   ```
   \def\Piano{1}%
   \def\Flute{2}%
   \def\Oboe{3}%
   \def\Soprano{4}%
   ```

   and code, for example

   `\setstaffs\Piano2` rather than `\setstaffs12`.

3. If, initially, the Piano is the instrument number 1, replace all `\notes`, `\Notes`, `\NOtes`, etc., with `\notes\selectinstrument\Piano`, `\Notes\selectinstrument\Piano`, `\NOtes\selectinstrument\Piano`, etc.

4. Instead of using & or `\nextinstrument` to move to instrument *n* + 1, use

   `\selectinstrument\Flute`

   and similar instead.

For example,

```
\Notes\selectinstrument\Piano ... | ...
      \selectinstrument\Flute ...
      \selectinstrument\Oboe  ...
      \selectinstrument\Soprano ... \en
```

With this coding, difficult things such as putting the Flute above the Oboe are done easily: just say
\def\Flute{3} and \def\Oboe{2}.

22.2   Selecting, hiding or putting instruments in the background

To put, for example, the Flute and the Oboe in the background, i.e., typesetting them in small notes, state
at the beginning:

> \setsize\Flute\tinyvalue\setsize\Oboe\tinyvalue

where the value \tinyvalue for \setsize corresponds to notes and staffs of size \tinynotesize. If,
instead of putting an instrument in the background, one wants to omit it, this is done by:

> \setstaffs\Flute{0}\setstaffs\Oboe{0}

since nothing is typeset for instruments having *zero* staffs (not to be confused with one-line staffs such as
percussions).

22.3   Recommendations

- When hiding instruments, reduce \instrumentnumber by the number of hidden instruments;
  otherwise bars and leading braces will enclose the position of these dummy arguments, which would
  be ugly.

- Exchange the actual instrument numbers so that hidden instruments have numbers *greater*
  than the value of \instrumentnumber. Hidden instruments with numbers less than
  \instrumentnumber will cause an excess of vertical space at their phantom positions; this is not
  recommended.

- In hidden instruments, rests no longer behave like \hbox{...} and raise-ing them will result in
  an error.

- In hidden instruments, explicit \hboxes will remain as empty boxes, thus causing abnormal vertical
  spacings between instruments. Therefore, anything suspect should be made conditional with:

  > \ifactiveinstrument *code to be omitted if instrument is hidden* \fi

  Most MusiXTEX commands become properly hidden if requested. But problematic parts of code
  can nonetheless be protected with \ifactiveinstrument.

Chapter 23

Lyrics
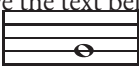
MusiXTEX itself doesn't manage lyrics very well. You should use `musixlyr` instead, a MusiXTEX extension package for lyrics handling by Rainer Dunker. The TEX source and <u>documentation</u> are included in the MusiXTEX distribution.

But first we recall briefly the older methods, which may still be useful when only a small number of words are involved.

23.1 Native lyrics method: placing single words

23.1.1 Native MusiXTEX commands for lyrics

1. An obvious solution is to use the commands `\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: ![notation] 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: ![notation] 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

109

was coded as:

```
\generalsignature{1}
\startextract
\geometricskipscale
\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
\zendextract
```

23.1.2   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 [clair] 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.

23.2   Musixlyr

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

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

The manual, the input file and a few examples are in MusiXTEX distributions, or they may be downloaded [for Windows] or [for UNIX]. Look at the manual for a detailed description. Here is an overview of the commands and an example of use.

Example	Explanation
`\setlyrics{sopr}{the ly_-ric words_}`	defining the lyrics text
`\copylyrics{sopr}{alto}`	alto has same lyrics as soprano
`\appendlyrics{alto}{more words}`	alto lyrics is longer
`\assignlyrics2{sopr,alto}`	soprano and alto lyrics at staff 2
`\assignlyricsmulti{1}{2}{alto}`	assign alto lyrics to staff 2 of instrument 1
`\NOtes\assignlyricshere{alto}\qa c\en`	assigning without staff number
`\auxlyr\assignlyrics{2}{sopr}`	assign soprano above staff 2
`\lyrrule\qu c...\lyrruleend\qu c`	make a melisma by hand
`\beginmel\qu c...\endmel\qu c`	melisma, same as word extension underline
`\lyr\qu c`	force a syllable from lyrics text at this note or rest
`\lyric{word}\qu c`	insert syllable 'word' at this note
`\loffset{2}{\lyric*{1.}}\qu c`	combine 1 . with regular syllable
`\lyrich{syl}\qu c`	same as `\lyric`, but with hyphenation
`\lyrich*{}\qu c`	same as `\lyric*`, but with hyphenation
`\lyricsoff...\lyricson`	stop lyrics, then start again
`\nolyr\qu c`	no syllable at this note
`\llabel{labelname}name`	labelling a "go to" target in text
`\golyr{labelname}\qu c`	perform a jump, in music code
`\lyrpt,\qu c`	add a comma to the syllable under this note
`\lyrnop\qu c`	remove last character in syllable
`\lclyr\qu c`	make first character lower case
`\llyr\qu c`	left justified syllable
`\leftlyrtrue\qu c...\leftlyrfalse\qu c`	start and stop left justification as the default
`\lyroffset{-4}\qu c`	shift syllable 1 notehead to the left
`\minlyrspace{3pt}\qu c`	define minimum space between the words
`\forcelyrhyphenstrue\qu c`	always use a hyphen from now on
`\forcelyrhyphensfalse\qu c`	remove hyphen and make one word if necessary
`\showlyrshifttrue\qu c`	show the lyric shift
`\lyrraise{1}{a 2\Interligne}`	raise lyrics below staff 1 by 2`\Interligne`
`\lyrraisemulti{1}{2}{a 2\Interligne}`	raise alto lyrics above staff 2 of instrument 1
`\lyrraisehere{b 2\Interligne}\qu c`	raise lyrics below this staff by 2`\Interligne`
`\minlyrrulelength{2mm}`	melismas shorter than 2mm are not shown
`\minmulthyphens{15mm}`	distance between hyphens in 'hyphen melisma'
`\def\lyrhyphenchar{-}`	chose a hyphen character
`\setlyrics{\lyrlayout{\it}..}`	apply italics to all lyrics lines
`\verses{,\beginmel}\qu c`	initiate melisma at second verse
`\small\setlyrstrut`	adapt the vertical distance between lyrics lines
`\lyrstrutbox{10pt}`	(re)define the distance between the lyrics lines
`\lyrmodealter2`	attach lyrics of staff 2 to the upper voice
`\lyrmodealtermulti{1}{2}`	attach lyrics of instr. 1 staff 2 to the upper voice
`\lyrmodealterhere\qu c`	attach lyrics of this staff to the upper voice
`\lyrmodenormal2`	restore the default behaviour
`\lyrmodenormalmulti{1}{2}`	restore the default behaviour at staff 2 of instr. 1
`\lyrmodenormalhere\qu c`	restore the default behaviour of this staff
`\lyrlink`	linking two words with a '‿'
`\lowlyrlink`	same as `\lyrlink` but a little bit lower
`\resetlyrics`	set word pointer to the first word in all lyrics lines
`\enableauxlyrics`	don't use this anymore

```
\setsongraise{1}{2\Interligne}          same as {\lyrraise}{1}{b 2\Interligne}
\auxsetsongraise{1}{2\Interligne}       same as {\lyrraisemulti}{1}{b 2\Interligne}
\oldlyrlinestart                        don't let the lyrics extend to the left margin
```

As a further illustration of the use of the commands, have a look at the following example[1]:
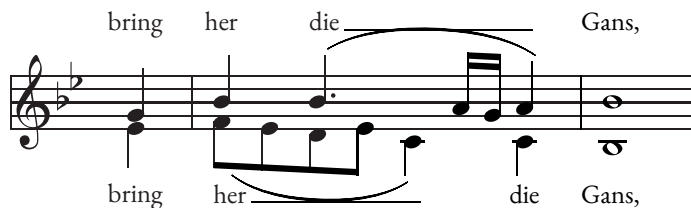
which was coded as:

```
% 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
\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\ibsluru1i\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\tbsluru1h\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
```

---

[1]The example is taken from the `musixlyr` manual.

23.3   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`.

23.4   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 so that the notes are farther apart. This may 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, \qqsk, ...`
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 \geometricskipscale
\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-move verse number

    ```
    \setlyrics{alto}{...\kern1exword...}
    ```
    right-move a single word

    ```
    \setlyrics{alto}{...\lower2pt\hbox{word}...}
    ```
    lower a single word

    ```
    \setlyrics{alto}{...\raise2pt\hbox{word}...}
    ```
    raise a single word

    ```
    \def\strut{\vbox to 2\Interligne{}}\setlyrics{alto}{\lyrlayout{\strut}...}
    ```
    control distance between verses

    ```
    \lyrlayout{\vphantom{Mp(\lowlyrlink}
    ```
    minimum distance between verses

    ```
    \setbox\lyrstrutbox=\hbox{\vphantom{yM\lyrlink}}
    ```
    redefine 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
                        \zendextract
    ```

7. Using an 8-bit encoded character set. If you use default (Computer Modern) fonts, you will want to switch to the EC variants by putting `\input musixec` after `\input musixtex` in your source file.

Chapter 24

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[1]. 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 case[2].

If for some reason you choose not to use LaTeX for 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 `\zendextract`. But some details in what follows will also apply without 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) or PDF file containing only the excerpt, and then "paste" it into the LaTeX file. That will be covered in Section 24.2.

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/PDF 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/PDF file for every excerpt, and that the excerpt must not span any page breaks. On balance, the direct method is probably to be preferred.

24.1 Directly embedding excerpts in LaTeX documents

To use the direct method, add `\usepackage{musixtex}` to the preamble. 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 line and is to be horizon-

---

[1] We'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

[2] Please 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.

tally 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 MusiXTEX 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 example[3]:

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

An even shorter type of extract is one that is embedded *within* a line of text. To insert MusiXTEX symbols within a line of text, you could begin by defining `\notesintext` as follows[4]:

```
\begin{music}\nostartrule
\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\rql1\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. 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 the source files for this document[5].

---

[3]The 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.

[4]provided by Rainer DUNKER

[5]Do note, however, that `musixdoc.tex` loads a `musixdoc.sty` package in its preamble, not just `musixtex.sty`; the former performs the functions of `musixtex.sty` as well as numerous tasks peculiar to this particular document.

24.2 Embedding musical excerpts as PDF or EPS files

To use this method of including excerpts, you first must create a separate MusiXTeX input file for each excerpt. Process each such file with TeX and `musixflx` to generate a `.dvi` file. Generate a Postscript file from each `.dvi` using `dvips`. Then convert each Postscript file to a `.pdf` or `.eps` file using `ps2pdf` or `ps2eps`; if you are using Windows, GSview can be used for either. Note that conversion to EPS (Encapsulated Postscript) is possible only for *single-page* Postscript files.

To set up your LaTeX document for including `.pdf` or `.eps` files, you must post the command `\usepackage{graphicx}` in the preamble of the document. Now, you may include each `sample.pdf` or `sample.eps` file at the appropriate place in the LaTeX document with a command like `\includegraphics{sample}`. If an embedded file is a PDF (rather than an EPS), the document must be processed with `pdflatex` (rather than `latex`).

24.3 Issues concerning `\catcodes`

MusiXTeX 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 MusiXTeX 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 MusiXTeX values, and `\endcatcodesmusic` to restore them to their prior values.

Chapter 25

Extension Library

All of the following files are invoked by saying \input *filename*. In some cases, `musixtex` must be input *before* the extension, but most of them are fully compatible with `musixtex.tex` in that they do not redefine any existing macros and just provide additional functionality. In future versions of MusiXTEX we may very well incorporate many of them directly into `musixtex.tex`, but for now we leave them separate.

25.1 musixadd

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

25.2 musixadf

This file and the associated fonts (by Hiroaki Morimoto) define some additional and alternative musical accents, as follows.

small swells, larger sforzatos:
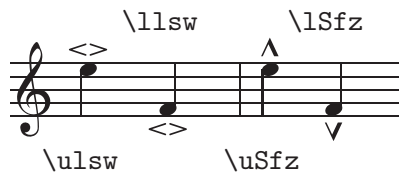
breaths:

coda with serifs, upright segno:
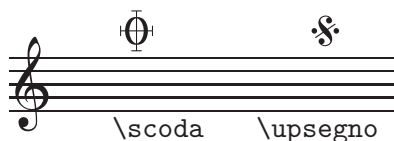
The following commands will replace the default specified accents by the new ones defined in `musixadf.tex`: \replacesforzato, \replacecoda, \replacesegno.

25.3   musixbm

This file does nothing; it is retained only for compatibility with MusiXTeX version 1.10 or earlier. Since version 1.11, `musixtex.tex` itself contains all the functions of the older `musixbm`, namely commands for 128th notes, either with flags or with beams: `\ibbbbbu`, `\ibbbbbl`, `\nbbbbbu`, `\nbbbbbl`, `\tbbbbbu`, `\tbbbbbl`, `\Ibbbbbu`, `\Ibbbbbl`, `\cccccu`, `\cccccl`, `\ccccca`, `\zcccccu` and `\zcccccl`.

25.4   musixbbm

Provides 256th notes, but only for use within beams, via the commands `\ibbbbbbu`, `\ibbbbbbl`, `\nbbbbbbu`, `\nbbbbbbl`, `\tbbbbbbu`, `\tbbbbbbl`, `\Ibbbbbbu` and `\Ibbbbbbl`.

By default `musixbbm` provides six 256th beams with reference number 0 to 5. You can specify a larger maximum number directly with `\setmaxcclvibeams{`m`}` within the range[1] $7 \leq m \leq 100$.

25.5   musixcho

Enables certain macros intended for choral music[2]. 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 then 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:

25.6   musixcls

Replaces the default Computer Modern text fonts by URWClassico fonts. URWClassico is a Hermann Zapf-authorized clone of the famous Optima sans-serif family from Linotype. See Section 17.2. Use as follows:

```
\input musixtex
\input musixcls
```

---

[1] This may require e-TeX.

[2] Remember 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.

```
...
```

The `classico` package is available at CTAN and in MiKTeX; however, it is not available in TeXLive because it is licensed for non-commercial usage only. TeXLive users can install the `classico` package using the `getnonfreefonts` script.

The usual "small" type commands are supported:

`\tinytype`	7pt
`\smalltype`	8pt
`\Smalltype`	9pt
`\normtype`	10pt
`\medtype`	12pt

For all of these, the following variants are supported:

`\rm`	Roman
`\bf`	bold
`\it`	italic
`\bi`	bold italic
`\sc`	small capitals

The "big" type commands are as follows:

`\bigtype`	14pt
`\Bigtype`	17pt
`\BIgtype`	20pt
`\BIGtype`	25pt

The default variant for the "small" types is Roman and for the "big" ones, bold small-caps. To letter-space a title, use the `\so` command in the `soul` package.

The following bold-italic fonts for dynamic marks are defined:

`\ppfftwelve`	8pt
`\ppffsixteen`	10pt
`\ppfftwenty`	12pt
`\ppfftwentyfour`	14pt
`\ppfftwentynine`	17pt

25.7   musixcpt

Empowers MusiXTeX to run files created under MusicTeX, the predecessor of MusiXTeX, such as some of the examples provided by Daniel Taupin. It is not needed for any files created under MusiXTeX, and is included mainly for historical completeness.

25.8   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 changed, either at the end of `musixdat.tex` for a permanant change, or right before issuing `\today`. Available choices and sample results are summarized below:

\dateUSenglish	November 2, 2025
\dateaustrian	2. November 2025
\dateenglish	2nd November 2025
\datefrench	2 novembre 2025
\dategerman	2. November 2025

25.9   musixdbr

Enables dashed and dotted bar lines (see Section 12.4).

25.10   musixdia

Enables notes with diamond-shaped heads as follows:

- Solid note heads (◆) are obtained using the macros \yqu, \yqup, \yqupp, \yql, \yqlp, \yqlpp, \yzq, \yzqp, \yzqpp, \yqb, \ycu, \yccu, \ycccu, \yccccu, \ycl, \yccl, \ycccl, \yccccl, \ycup, \ycupp, \yclp, \yclpp. (Think of d*y*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, \dcccu, \dccccu, \dcl, \dccl, \dcccl, \dccccl, \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,
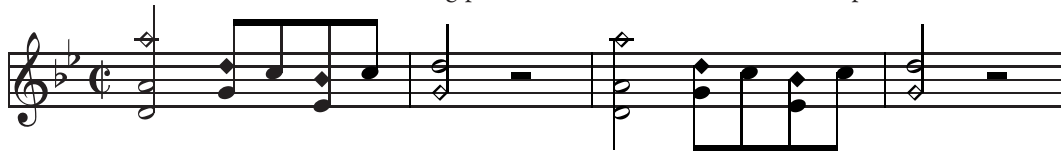
was coded as follows:

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

Another use is for percussion parts. In fact the file musixdia.tex is automatically loaded if you input musixper.tex (see Section 25.26).

25.11   **musixec**

This package will replace the `OT1`-encoded Computer Modern fonts by `T1`-encoded EC versions. Use as follows:

```
\input musixtex
\input musixec
[\input musixsty]
...
```

For example,
     é è â ç ü ñ
may be coded as

```
\input musixec
é è â ç ü ñ\\
```

25.12   **musixebgr**

Replaces the default Computer Modern text fonts by EBGaramond fonts; see Section 17.2. Use as follows:

```
\input musixtex
\input musixebgr
...
```

You may need to install the `ebgaramond` package.

The usual "small" type commands are supported:

```
\tinytype     7pt
\smalltype    8pt
\Smalltype    9pt
\normtype    10pt
\medtype     12pt
```

For all of these, the following variants are supported:

```
\rm    Roman
\bf    bold
\it    italic
\bi    bold italic
\sc    small capitals (and old-style figures)
```

The "big" type commands are as follows:

```
\bigtype    14pt
\Bigtype    17pt
\BIgtype    20pt
\BIGtype    25pt
```

The default variant for the "small" types is Roman and for the "big" ones, small-caps; however, `\font` commands are defined for all the usual variants at all of these sizes. To letter-space a title, use the `\so` command in the `soul` package.

The following bold-italic fonts for dynamic marks are defined:

```
\ppfftwelve        8pt
\ppffsixteen      10pt
\ppfftwenty       12pt
\ppfftwentyfour   14pt
\ppfftwentynine   17pt
```

25.13 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:

original	*alternate*
\PAUSe	\Qwr
\PAuse	\Dwr
\liftpause	\liftwr
\pausep	\wrp
\pause	\wr
\lifthpause	\lifthr
\hpausep	\hrp
\hpause	\hr
\qp	\qr
\ds	\er
\qs	\eer
\hs	\eeer
\qqs	\eeeer

25.14 musixext

This file does nothing; it is retained only for compatibility with MusiXTEX version 1.17 or earlier. Since version 1.18, `musixtex.tex` itself contains all the functions of the older `musixext`, namely definitions of

- `\raggedstoppiece`, which inhibits right-justification of the last line of a score.
- `\slide{`p`}{`x`}{`s`}`, which provides a glissando that starts at pitch p and extends for x `\internotes` with slope s (ranging from -8 to 8).

25.15 musixfll

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, MusiXTEX shortens the ledger lines if notes are set so close together that the ledger lines may meet. But because MusiXTEX does not know whether consecutive notes need ledger lines, this automatic shortening may be superfluous. The extension file `musixfll.tex` allows this feature to be switched off and on. Upon inputting `musixfll.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), whereas single notes requiring ledger lines look better with `\longledgerlines`.
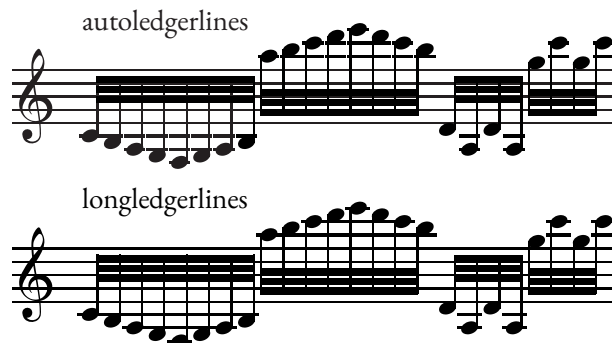
25.16   musixftab

This extension replaces the default (Helvetica) numerals of the tablature commands of Section 18.10 with the traditional script letters of French tablature. These glyphs are generated in `\tabfnt` by arguments `0–9` or by (capital) Latin letters `A–R` (excluding `J`).

The example of Section 18.10 with `musixftab.tex` is as follows:

25.17 musixgre

Gregorian chant is often coded using four line staffs (see Section 21.3) 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 MusiXTEX 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 15.3), 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}&\en
\zendextract
```

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

25.17.1   Clefs

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

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

25.17.2   Elementary symbols

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

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

25.17.3   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 comples 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
\setstaffs 1 1
\setlines 1 4
\setclef 1{3000}
\setaltoclefsymbol 1 \gregorianCclef
\startextract
\notes \bivirga ab\en
\notes \bivirga cc\en
\zendextract
```

`\trivirga{`p_1`}{`p_2`}{`p_3`}` , for example:

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

`\tristropha{`p_1`}{`p_2`}{`p_3`}` , for example:

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

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

\podatus{p_1}{p_2} , for example:

\podatusinitiodebilis{p_1}{p_2} , for example:

\lpodatus{p_1}{p_2} , for example:

\pesquassus{p_1}{p_2} , for example:

\quilismapes{p_1}{p_2} , for example:

\torculus{p_1}{p_2}{p_3} , for example:

\torculusinitiodebilis{p_1}{p_2}{p_3} , for example:

\Porrectus{p_1}{p_2}{p_3} , for example:

coded:

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

\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{p_1}{p_2}{p_3}{p_4} , for example:

coded:

```
\notes \Porrectusflexus  bacN\en
\notes \Porrectusflexus  bNdb\en
\notes \Porrectusflexus  bMeb\en
\notes \Porrectusflexus  bLea\en
```

\climacus{p_1}{p_2}{p_3} , for example:

\climacusresupinus{p_1}{p_2}{p_3}{p_4} , for example:

\lclimacus{p_1}{p_2}{p_3} , for example:

\scandicus{p_1}{p_2}{p_3} , for example:

\salicus{p_1}{p_2}{p_3} , for example:

\salicusflexus{p_1}{p_2}{p_3}{p_4} , for example:
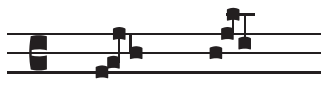
\trigonus{p_1}{p_2}{p_3} , for example:

25.17.4   Liquescens complex neumes

\clivisauctup{p_1}{p_2} , for example:

\clivisauctdown{p_1}{p_2} , for example:

\podatusauctup{p_1}{p_2} , for example:

\podatusauctdown{p_1}{p_2} , for example:

\pesquassusauctdown{p_1}{p_2} , for example:

\quilismapesauctdown{p_1}{p_2} , for example:

\torculusauctdown{p_1}{p_2}{p_3} , for example:

\Porrectusauctdown{p_1}{p_2}{p_3} , for example:

\climacusauctdown{p_1}{p_2}{p_3} , for example:

\scandicusauctdown{p_1}{p_2}{p_3} , for example:

\salicusauctdown{p_1}{p_2}{p_3} , for example:

\clivisdeminut{p_1}{p_2} , for example:

\podatusdeminut{p_1}{p_2} , for example:

\torculusdeminut{p_1}{p_2}{p_3} , for example:

\torculusdebilis{p_1}{p_2}{p_3} , for example:

`\Porrectusdeminut{`p_1`}{`p_2`}{`p_3`}` , for example:

`\climacusdeminut{`p_1`}{`p_2`}{`p_3`}` , for example:

`\scandicusdeminut{`p_1`}{`p_2`}{`p_3`}` , for example:

25.18   musixgui

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

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 chords, 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}%`

25.19   musixhv

Replaces the default Computer Modern text fonts by Helvetica (sans serif) fonts; see Section 17.2. Use as follows:

```
\input musixtex
\input musixhv
...
```

The usual "small" type commands are supported:

```
\tinytype     7pt
\smalltype    8pt
\Smalltype    9pt
\normtype    10pt
\medtype     12pt
```

For all of these, the following variants are supported:

```
\rm   "Roman" (sans serif)
\bf   bold
\it   "italic" (actually oblique)
\bi   "bold-italic" (actually bold-oblique)
\sc   small capitals
```

The "big" type commands are as follows:

```
\bigtype   14pt
\Bigtype   17pt
\BIgtype   20pt
\BIGtype   25pt
```

The default variant for all of these is small-caps; however, `\font` commands are defined for all the usual variants at all of these sizes. To letter-space a title, use the `\so` command in the `soul` package.

The following bold-oblique fonts for dynamic marks are defined:

```
\ppfftwelve       8pt
\ppffsixteen     10pt
\ppfftwenty      12pt
\ppfftwentyfour  14pt
\ppfftwentynine  17pt
```

25.20   musixjt

Jazz scores often use a "handwriting" font `JazzText` originally designed by Richard Sigler and now bundled with Finale. This font is widely available and is said to be "free for personal use" but this is not sufficiently "free" for packaging into TeXLive. A similar font `MuseJazzText` is bundled with MuseScore and because it is OFL-licensed, it can be distributed with MusiXTEX. This font will be used for all textual elements of your MusiXTEX score if you input `musixjt.tex` after `musixtex.tex`. Here is an example:

25.21   musixlit

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

This package provides:

- \oldGclef which replaces the ordinary G clef with an old one, using (for instrument 2 as an example): \settrebleclefsymbol2\oldGclef
- \cqu *p* provides a square-headed quarter note with stem up at pitch *p*.
- \cql *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*.
- \cqb *n p* provides a square-headed beam note for beam *n* at pitch *p*.
- \zcqb *n p* provides a non-spacing square-headed beam note for beam *n* at pitch *p*.
- \cnqu *p* and \cnql *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*.
- \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; in the first example, \Hpause was used to denote an arbitrary length note rather than a rest.
- \shortbarrules has been used to provide bar rules shorter than the staff vertical width.
- \interbarrules has been used to provide bars between the staffs, rather that over them. This is an arbitrary question of taste…

25.22   musixlyr

Enables the recommended method for adding lyrics to a score (see Section 23.2).

25.23   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`. If you need greater numbers of these elements, see Sections 2.4, 5.1, 9.3, 10.1 and 18.4.3.

25.24   musixmkm

The following special accidentals used in Turkish classical music (makam) are supported:

These are generated by, respectively,

```
\shtanini p
\shbuyukmucenneb p
\shkucuk p
\shbakiye p
\shkoma p
\flkoma p
\flbakiye p
\flkucuk p
\flbuyukmucenneb p
\fltanini p
```

25.25   musixmtab

Modern instrumentalists of string instruments (guitar, banjo, mandolin, etc.) use a hybrid form of tablature in which notes have (downward) stems and beams attached.[3] The `musixmtab` package combines traditional Gelten-style tablature notation already in MusiXTEX with the stems and beams of standard music notation. It is assumed that there is at most one beam (beam number 0) and that all stems are downward.

In the following commands, {s} is a string number (counting from the top) and {n} is a numerical note (i.e., fret number).

---

[3]It seems this "modern" form of tablature was introduced by Pete Seeger in his 5-string banjo instruction book, first published in 1948.

\uh{*s*}{*n*}	a half note (boxed, to differentiate from a quarter note)
\zuh{*s*}{*n*}	a non-spacing stemless half note
\uq{*s*}{*n*}	a quarter note (with a centered stem)
\uc{*s*}{*n*}	an eighth note (with a flag)
\ucc{*s*}{*n*}	a sixteenth note (with a double flag)
\zu{*s*}{*n*}	a non-spacing stemless quarter, eighth or sixteenth note
\uib{*s*}{*l*}	initiate a single beam; *s* is the starting string and *l* is the slope
\uibb{*s*}{*l*}	initiate a double beam; *s* is the starting string and *l* is the slope
\ub{*s*}{*n*}	a beamed note
\unbb	increase the beam multiplicity to 2
\utbb	decrease the beam mutiplicity to 1 after the following note
\utb	terminate the beam after the following note
\utq{*s*}{*n*}	terminate the beam after this note

Conventional MusiXTEX commands such as pt and arpeggio may be used but arguments must reference MusiXTEX note designations rather than string numbers.

Here is an example, *Cripple Creek*, for 5-string banjo in standard G tuning, clawhammer style:

25.26   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 ⊗ symbol which is obtained using the \qu, \qb, \cu, etc. macros preceded by a "dc" (think of *d*iagonal *c*ross). Available are \dcqu, \dcql, \dcqb, \dczq, \dccu, \dcccu, \dccl and \dcccl.
- The ⊗ symbol which is obtained using the \qu, \qb, \cu, etc. macros preceded by a "dh" (think of *d*iagonal cross *h*alf open). Available are \dhqu, \dhql, \dhqb, \dhzq, \dhcu, \dhccu, \dhcl and \dhccl.
- The ✕ symbol which is obtained using the \qu, \qb, \cu, etc. macros preceded by a "do" (think of *d*iagonal cross *o*pen). Available are \doqu, \doql, \doqb, \dozq, \docu, \doccu, \docl and \doccl.
- The ✗ symbol which is obtained using the \qu, \qb, \cu, etc. macros preceeded by "x" (e.g. for spoken text of songs). Available are \xqu, \xql, \xqb, \xzq, \xcu, \xccu, \xcl and \xccl.
- The ⊗ 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 ◢ symbol which is obtained using the \qu, \qb, \cu, etc. macros preceded by "ro" (think of *r*h*o*mbus). Available are \roqu, \roql, \roqb, \rozq, \rocu, \roccu, \rocl and \roccl.

- The ⟨symbol which is obtained using the \qu, \qb, \cu, etc. macros preceeded by "tg" (think of *t*rian*g*le). Available are \tgqu, \tgql, \tgqb, \tgzq, \tgcu, \tgccu, \tgcl and \tgccl.
- The + symbol which is obtained using the \qu, \qb, \cu, etc. macros preceeded by "k" . Available are \kqu, \kql, \kqb, \kzq, \kcu, \kccu, \kcl and \kccl.

The diamond shaped noteheads described in Section 25.10 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 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 latter[4]:
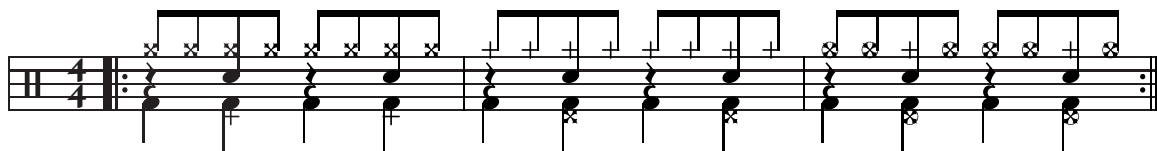
Its coding is

```
\begin{music}
\instrumentnumber{1}
\generalmeter{\meterfrac44}
\setclefsymbol1\drumclef
\parindent0pt\startpiece
\leftrepeat
\Notes\zql f\rlap\qp\ibu0m0\xqb0{nn}\en
\Notes\kzq d\zql f\zq j\xqb0n\tbu0\xqb0n\en
\Notes\zql f\rlap\qp\ibu0m0\xqb0{nn}\en
\Notes\kzq d\zql f\zq j\xqb0n\tbu0\xqb0n\en
\bar
\Notes\zql f\rlap\qp\ibu0m0\kqb0{nn}\en
\Notes\xzq d\zql f\zq j\kqb0n\tbu0\kqb0n\en
\Notes\zql f\rlap\qp\ibu0m0\kqb0{nn}\en
\Notes\xzq d\zql f\zq j\kqb0n\tbu0\kqb0n\en
\bar
\Notes\zql f\rlap\qp\ibu0m0\oxqb0{nn}\en
\Notes\oxzq d\zql f\zq j\kqb0n\tbu0\oxqb0n\en
\Notes\zql f\rlap\qp\ibu0m0\oxqb0{nn}\en
\Notes\oxzq d\zql f\zq j\kqb0n\tbu0\oxqb0n\en
\setrightrepeat\endpiece
```

---

[4]provided by Agustí Martín Domingo

```
\end{music}
```

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

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}
\setstaffs1{2} % 2 staffs 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}\en\bar
\NOtes\qu I|\zq N\qu d&\qp&\diapunc f\en
\NOtes\qu J|\zq a\qu e&\ynq4&\diapunc f\en\bar
\notes\hu G|\zh b\hu d&\dnq4&\zsqu d\rsqu g\squ{hgh}\en
\endextract
```

25.27   musixplt

Replaces the default Computer Modern text fonts by Palatino fonts; see Section 17.2. Use as follows:

```
\input musixtex
\input musixplt
...
```

You may need to install additional packages: `psnfss`, `palatino` and `fpl`.

The usual "small" type commands are supported:

`\tinytype`	7pt
`\smalltype`	8pt
`\Smalltype`	9pt
`\normtype`	10pt
`\medtype`	12pt

For all of these, the following variants are supported:

`\rm`	Roman
`\bf`	bold
`\it`	italic
`\bi`	bold italic
`\sc`	small capitals (and old-style figures)

The "big" type commands are as follows:

`\bigtype`	14pt
`\Bigtype`	17pt
`\BIgtype`	20pt
`\BIGtype`	25pt

The default variant for the "small" types is Roman and for the "big" ones, small-caps; however, `\font` commands are defined for all the usual variants at all of these sizes. To letter-space a title, use the `\so` command in the `soul` package.

The following bold-italic fonts for dynamic marks are defined:

`\ppfftwelve`	8pt
`\ppffsixteen`	10pt
`\ppfftwenty`	12pt
`\ppfftwentyfour`	14pt
`\ppfftwentynine`	17pt

25.28 musixpoi

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

25.29 musixppff

This replaces the default definitions used for dynamic marks to use a "mini-font" `xppff10` designed by Hiroaki MORIMOTO. It is used as follows:

```
\input musixtex
\input musixppff
...
```

The new dynamic marks are as follows:

pppp ppp pp p ᵐp ᵐf f fp sf ff fff ffff sfz sfzp

Note that the xppff10 font has defined glyphs only for the following five characters: f, m, p, s and z; use \ppff etc. for other bold-italic text in music.

25.30   musixps

Activates type K Postscript slurs, ties, and hairpins; see Chapter 11. Use as follows:

```
\input musixtex
\input musixps
...
```

25.31   musixstr

Provides bowing and other symbols for *string instruments*[5]. 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
V **:** \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

25.32   musixsty

Provides certain text-handling facilities for titles, footnotes, and other items not related to lyrics. It should not be used with LᴬTEX. It includes
- definitions of \hsize, \vsize, \hoffset, \voffset suitable for A4 paper; those using other sizes may wish to modify it once and for all;
- a \rectoverso{*dimension*} command to set up even/odd-page asymmetric margins;
- 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 (or \twelvepoint) to get 12 point font size;
  \frtpoint to get 14.4 point font size;
  \svtpoint to get 17.28 point font size;

---

[5]provided by Werner Icking

`\twtypoint` to get 20.74 point font size;

`\twfvpoint` to get 24.88 point font size;

- commands for creating title-page headings:
    - `\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;
    - `\headline` which displays the argument at the top of the title page (use `\hss` after or before the argument text or both to left-align or right-align or center the text);
    - `\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. enter the footnote itself using `\vfootnote` outside the music, either before `\startpiece` or between `\stoppiece` and `\contpiece` or equivalent commands.

25.33   musixthacc

Provides "thin" accidentals `\thna` p, `\thsh` p, `\thfl` p, `\thdsh` p and `\thdfl` p:

Any necessary adjustment of horizontal position and spacing is up to the user and `\relativeaccid` adjustment is not effective.

25.34   musixtmr

Replaces the default Computer Modern text fonts by Times fonts; see Section 17.2. Use as follows:

```
\input musixtex
\input musixtmr
...
```

The usual "small" type commands are supported:

```
\tinytype    7pt
\smalltype   8pt
\Smalltype   9pt
\normtype    10pt
\medtype     12pt
```

For all of these, the following variants are supported:

```
\rm    Roman
\bf    bold
\it    italic
\bi    bold italic
\sc    small capitals
```

The "big" type commands are as follows:

```
\bigtype    14pt
\Bigtype    17pt
\BIgtype    20pt
\BIGtype    25pt
```

The default variant for all of them is Roman; however, \font commands are defined for all the usual variants at all of these sizes. To letter-space a title, use the \so command in the soul package.

The following bold-italic fonts for dynamic marks are defined:

```
\ppfftwelve       8pt
\ppffsixteen      10pt
\ppfftwenty       12pt
\ppfftwentyfour   14pt
\ppfftwentynine   17pt
```

25.35   musixtnt

This package (tnt = Transform Notes) provides a macro \TransformNotes which makes possible several "transformations" of the effect of notes commands such as \notes. The musixtnt package is distributed separately from musixtex.

In general the effect of \TransformNotes{*input*}{*output*} is that subsequent notes commands in the source will expect their arguments to match the *input* pattern but the notes will be typeset according to the *output* pattern.

For example,

```
\TransformNotes{#2&#3&#4&#5}{#2&#3&#5}
```

would be appropriate for a four-instrument score (arguments #2, #3, #4, and #5, separated by three &s, but the third instrument (#4) will be discarded.

The instrument/staff numbers in the first argument must start at 2 and increase consecutively, using & (or | for multi-staff instruments) as a separator. The reason that the segment identifiers start at 2 is that the first argument for \vnotes is a spacing parameter.

It is essential that every \znotes, \notes, \Notes, \NOtes, etc. command in the score match the pattern of the first argument to \TransformNotes exactly; for example, too few (or too many) note segments will result in unintentionally discarded material and possibly compilation failure. Similarly, it is not possible to use \nextinstrument, \nextstaff, \selectinstrument or \selectstaff, or to hide the & (or |) tokens inside a user-defined macro. An auxiliary program msxlint distributed in the musixtnt package can be used to detect such incompatibilities.

\TransformNotes may be used anywhere between \startpiece and the command that ends the piece.

25.35.1   Extracting single-instrument parts from multi-instrument scores

To extract a single-instrument part from a (copy of a) multi-instrument MusiXTEX score:

- Set \nbinstrument to 1; for example, with command \instrumentnumber1.

- Use \TransformNotes to discard all but one of the note segments in notes commands. For example, the following line placed after \startpiece (but before any note commands) would be appropriate for a four-instrument score and will result in a single-instrument part for the second of these:

    \TransformNotes{#2&#3&#4&#5}{#3}%

Caveats:

- Some additional revisions to the source for the part might be necessary:
    – adjusting \setname, \setclef, \setsign, \setmeter and \setstaffs commands, as necessary; see Section 22.1 for coding techniques that will make this easier.

    – ensuring that tempo and roadmap markings (**D.C.**, **Fine**, etc.) are in the appropriate instrument segment;

- When the extracted part score is compiled and viewed, it may be seen that horizontal-spacing commands designed for *multiple* instruments can produce bad spacing when used for a *single* instrument. Bad spacing can be corrected manually but this is very tedious; an auxiliary program called autosp can be used to automate this process; see Section 3.4.

25.35.2   Other applications

The \TransformNotes macro may be used for other purposes. Here are some examples:

    \TransformNotes{#2&#3}{#2&\transpose+7#3}%

will begin transposing just the second instrument (argument #3).

    \TransformNotes{#2|#3&#4}{#2|#3&\tinynotesize#4}%

will begin typesetting the notes of the second instrument (#4) in tiny size.

    \TransformNotes{#2&#3}{#3&#2}%

will switch the order of the two instruments.

        \TransformNotes{#2&#3}{#2&#3}%

will restore normal two-instrument processing.

The `musixtnt` implementation of `\TransformNotes` is not compatible with the `musixlyr` extension package for lyrics described in Section 23.2. For extracting or omitting parts in scores with lyrics, one may either use the `autosp` preprocessor discussed in Section 3.4, which provides an alternative implementation of `\TransformNotes`, or the techniques described in Chapter 22.

25.36 musixtri

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

25.37 musixvbm

By default, `musixtex` produces beams using special fonts. Unfortunately, long beams may have unsightly minor discontinuities (gaps or bumps). The `musixvbm` add-on by Hiroaki MORIMOTO produces smooth vectorized beams; simply input `musixvbm` in the preamble of your score (after inputting `musixtex`). For example, compare

 and
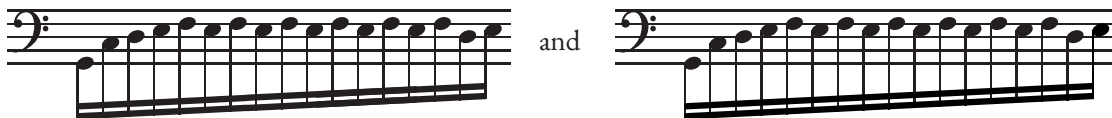
with various magnifications. The output on the left shows a font-based beam; the one on the right shows the same beam with `musixvbm` activated. (All the other beams in this report are vectorized.) `musixvbm` also supports slopes greater than and less than 9 and -9, respectively.

The only known disadvantage with use of `musixvbm` is that many `dvi` viewers (including `xdvi` and `yap`) do not support the "`tpic` specials" that implement vectorized beams in `dvi` output, so that no beams are shown[6]; however, Postscript and PDF output generated using `dvips` and `ps2pdf` (or using `dvipdfm`, or by processing the score using `pdfetex`) should be satisfactory.

If Postscript hairpins (as described in Section 11) are not being used, the `musixvbm` add-on also vectorizes crescendos and decrescendos.

25.38 tuplet

This file does nothing; it is provided for compatibility with MusiXTEX version 1.20 or earlier and legacy scores. Since version 1.21, `musixtex.tex` itself contains all the functions of the older `tuplet.tex`, namely definitions of `\tuplettxt`, `\uptuplet` and `\downtuplet`. See Section 18.3.

---

[6]The `dviout` program for Windows supports `tpic` specials; it is available from CTAN and in the Win32 distribution of TeXLive.

Chapter 26

MusiXTEX Examples

The file `musixdoc.tex`, the source for this manual, contains many useful examples. In the manual, 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 and you'll have to look in the `musixdoc` source files to see the coding.

Other useful examples cannot be embedded in the source, either because they are meant to be in TEX, not LATEX, or because they are simply too large. For these the source files also 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` TEX commands have been included within the TEX source.

26.1   Small examples

- `ossiaexa.tex` : This is a stand-alone example of the use of ossia, provided by Olivier Vogel (Section 19.3 on page 100).
- `accents.tex` : Accents using an 8-bit font.

26.2   Full examples

The small examples above and the longer ones mentioned below are included in the documentation folder of the MusiXTEX package. Some of them require `musixcpt.tex` which makes examples created in MusicTEX compatible with MusiXTEX. Here we mention a few of special interest.

26.2.1   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 12.2) and the use of staves of different sizes (Section 20). 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 23.1.2). `gloriab.tex` is the same

piece, but with organ accompaniment.

26.2.2 Other examples, provided by the authors of MusiXTEX

- `traeumer.tex` : the famous "Träumerei" by Robert SCHUMANN for piano, in genuine MusiXTEX 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.

26.2.3 Additional documentation

- `sottieng` : Notation mistakes, provided by Jean-Pierre Coulon.

26.3 Compiling `musixdoc.ltx`

This manual is an excellent example of a primarily text document with embedded musical excerpts. For this reason, it is a LATEX document and must be compiled with `latex` rather than `etex`. Those wishing to combine text and musical excerpts should carefully study how it is done here.

Before compiling or recompiling `musixdoc.ltx`, you should remove every auxiliary file `musixdoc.[mx1|mx2|aux|toc|ind|idx|ilg|out]` if they are present. Then the following command sequence will produce `musixdoc.ps`:

```
latex musixdoc
musixflx musixdoc
latex musixdoc
makeindex musixdoc
latex musixdoc
latex musixdoc
dvips -e0 musixdoc
```

The initial three steps `latex`→`musixflx`→`latex` build up the basic appearance of the document including musical examples. The `makeindex` step produces the database for the index. After that, `latex` must be run at least twice to complete cross referencing. Finally, `dvips` converts the `.dvi` file into `.ps`; the `-e0` option prevents `dvips` from making pixel adjustments intended for letters in words. To produce `musixdoc.pdf`, use `ps2pdf` or the Distiller component of Adobe Acrobat, or open `musixdoc.ps` in GSview, go to File|Convert, select `pdfwrite` at 600 dpi resolution, and click OK.

Alternatively, if you have the `musixtex` script, just run `musixtex -x musixdoc` (where the `-x` option is only needed to create a new index database).

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

'A 'B 'C 'D 'E 'F 'G

'a 'b 'c 'd 'e 'f 'g "a "b "c "d "e "f "g """a """b """c """d """e

a b c d e f g h i j k l m n o p q r s t u v w x y z

Notes and Accidentals

\longa \wq \wqq \wh \hu \hl \qu \ql \cu \cl \ccu \ccl \cccu\cccl \ccccu\ccccl\grcu\grcl

\maxima \breve

Accidentals: > \cdsh ^ \csh = \cna _ \cfl < \cdfl

\dqu¹²³ \yqu¹²³ \dcqu² \dhqu² \doqu² \xqu² \oxqu² \roqu² \tgqu² \kqu² \squ³ \lsqu³ \rsqu³ \cqu⁴ \cql⁴ \chu⁴ \chl⁴

Pointed Notes and Accents

\lpz \upz \lsf \usf \lst \ust \lppz \uppz \lsfz \usfz \lpzst \upzst \downbow \upbow \flageolett \whp \qupp

Accent on beam with prefix b and beam reference number instead of the pitch

Clefs

\trebleclef \bassclef \smallaltoclef \drumclef² \gregorianFclef³

\altoclef \smalltrebleclef \smallbassclef \gregorianCclef³ \oldGclef⁴

Rests

\qqs \hs \qs \ds \qp \hp \qqsp \hsp \qsp \dsp \qpp \hpp \pause \pausep \PAuse \PAUSe \Hpause⁴

\lifthpause \liftpause

Miscellaneous Symbols

\mordent \turn \backturn \Shakesw \Shakenw

\tr\Trille \trille \shake \Shake \Mordent \Shakel \Shakene \trt

tr tr

\allabreve \meterC \reverseC \meterplus \duevolte \l[r]par

\reverseallabreve

3+2+3
8

♩ = 99

\metron \setvoltabox\setvolta \coda\Coda\segno \Segno

I. 2.

\PED \sPED\pdld \pdlud \pdlu\DEP \sDEP

\caesura \cbreath

\fermataup \arpeggio d5 \uptrio \octfinup \slide \boxit A \circleit B

\Fermataup \bracket

3

8

A B

\Fermatadown \downtrio \octfindown \leftrepeat \rightrepeat

\fermatadown 8ᵃ bassa \leftrightrepeat

¹musixdia.tex ²musixper.tex ³musixgre.tex ⁴musixlit.tex

Index

155