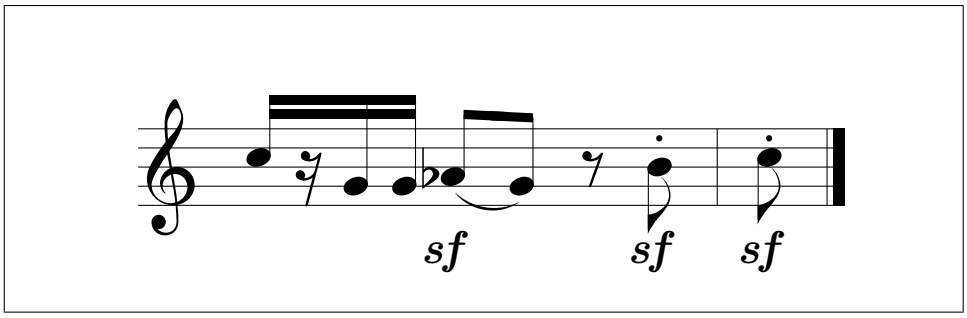
M-Tx: Music from Text

Version 0.60

User's Guide

Dirk Laurie
dlaurie@na-net.ornl.gov

16 March 2005



Contents

1 Music from Text

Written music has a distinctive appearance, very unlike written text. But if we want to enlist the aid of the computer to print our scores, we either need a sophisticated graphical interface[1], or we must describe music in terms of what we are able to type. Text input is not going to look much like printed music, but we would like it to resemble printed music closely enough that we can see the correspondence.

M-Tx is a language for typed music aimed at people who are not experts. Later in this document a full description of ***M-Tx*** is given, but to start with, here is a sample of what ***M-Tx*** input looks like.

```
c2+       e4    g   | b4d-  c1 d c2        |
c8 g+ e g c- g+ e g | d g f g    c- g+ e g |
```

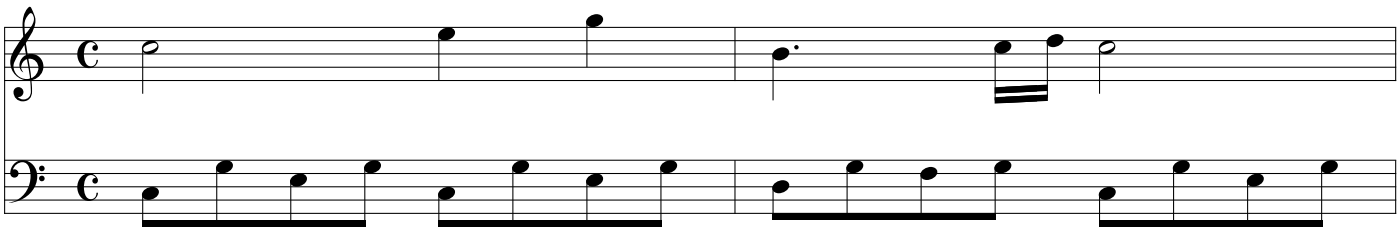
Before showing you how it comes out, let's observe and discuss it. There are two lines of input, which means that there are two voices of music, one for each line. (Line boundaries are important in ***M-Tx***.) The lines read from top to bottom as printed music reads from top to bottom.

Each voice contains a number of 'words' separated by blanks. In this example, each word stands for either a note or a bar line. Bars are also important in ***M-Tx***, although the example is simple enough that I could have omitted them. For greater readability, I have typed extra blanks so that notes and bars that are aligned in the printed version also line up here, but that is not necessary. What is necessary, though, is that each line ends at the end of a complete bar.

The notes are written in the **PMX** language designed by Don Simons. Each note starts with one of the letters `a` to `g`, which mean exactly what you think. Lower case is easier than capitals, because it is quicker to type. Some notes have suffixes after them. A digit indicates the duration (1/2, 1/4, 1/8, 1/16, 1/32, 1/64).[2] When a note has no digit, it means that the duration is the same as that of the previous note in the same voice. Some note names have a `+` or `-` after them, which means that it is not the nearest note of that name, but that there is a jump of at least a fifth up or down from the previous note. One note is dotted, which is indicated with a `d`. It cannot be confused with a note name, because it is not the first letter of its word. A double-dotted note would have `dd`.

The first note in each voice of the piece doesn't have a previous note. Since you haven't specified anything else, it is assumed that the top line is in the treble clef and its notes come in the octave running from middle C upwards; and the bottom line is in the bass clef and its notes come in the octave just below the one for the treble clef. The first note of the top voice is the C above middle C, so we need a `+` for it. If ever again you prefer the initial octave of a voice to wherever the previous note puts you, use a `=` after the note name. This can be combined with `+` or `-`.

Here is the printed version of the above excerpt.



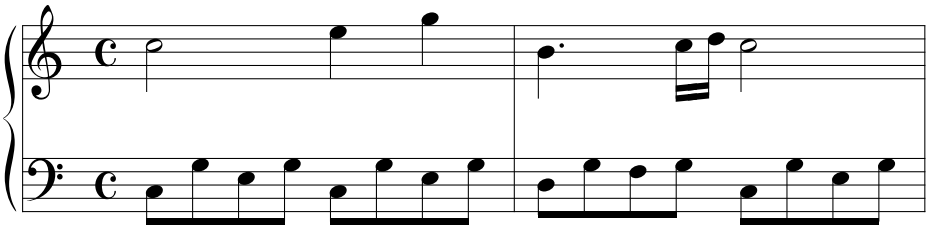
[1] One such is `noteedit` by Jörg Anders obtainable from the Archive — see Section C.1.

[2] Yes, it is more logical to think `1` should mean a whole note, but we don't want to type two digits for something so common as a semiquaver. A whole note is denoted by `0`, which looks a bit like a printed semibreve.

Do you think this looks good? Much better than handwritten? You are right, but look at the following version:

Riff in C

W. A. Mozart (1756–1791)



There is a heading, the notes are not stretched out so much, and the two staves of the piano are grouped together. To achieve this, the input looks as follows:

```
Title:  Riff in C
Composer: W. A. Mozart (1756--1791)
Style: piano

%% w120m
c2+       e4    g   | b4d-  c1 d c2        |
c8 g+ e g c- g+ e g | d g f g    c- g+ e g |
```

The notes are exactly the same as before, but there is an introductory paragraph, or preamble, that specifies what to do with them. A paragraph in ***M-Tx*** is a group of consecutive lines set apart by one or more blank lines. Most of the paragraphs contain notes, but the printed music will not necessarily be broken at the same places as the paragraphs.

Only in the simplest cases (every music line in the paragraph starts with what might be a note) are you allowed to omit the preamble.

The preamble lines in this example are self-explanatory. The one mysterious line is the one starting with %%at the top of the music paragraph. The present implementation of ***M-Tx*** acts as a front end to the **PMX** program. Things that can be done easily in **PMX** are not re-invented, but instead the %% facility is provided to pass information directly to **PMX**. The **PMX** command `w120m` means 'width 120 millimetres'. A short tutorial on **PMX** appears in Appendix B.

A few words on technical terms, to make sure that we know what we are talking about:

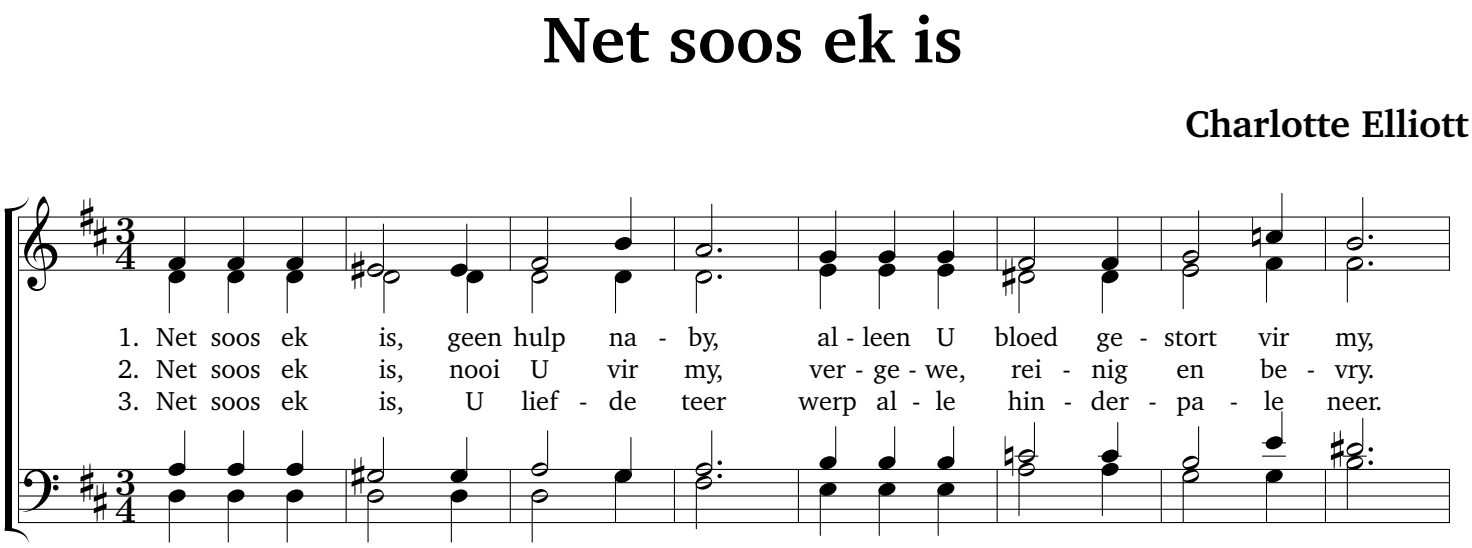
- A **line** is a text line of typed music.
- A **word** is a string of consecutive non-blank characters, separated by blanks from other words on the same line.
- A **stave** is the familiar group of five closely spaced parallel lines on which music is written.
- A **voice** is a melodic strand of music, of which there may be either one or two per stave.[3]

[3]Experienced **PMX** users please note: the **PMX** manual uses 'voice' as a synonym for 'stave', and 'line of music' for what I call a 'voice'.

- An **instrument** corresponds either to a single stave or to a group of two or more adjacent staves linked together with a brace {, usually representing a single instrument like a piano.
- A **system** is a group of staves for the various voices that are heard at the same time.

1.1 Lyrics

I wrote the ***M-Tx*** converter because lyrics are not part of the **PMX** design. It used to be[4] a major effort to obtain pretty output like the following:



The lyrics are in Afrikaans, which maybe looks incomprehensible, but knowing as much as you do already, it should be quite easy to understand the following input:

```
Title: Net soos ek is
Composer: Charlotte Elliott
Style: SATB
Sharps: 2
Meter: 3/4
Space: 9

%% w190m
f f f | e2s e4  | f2 b4 | a2d | g4 g g | f2 f4  | g2 c4n | b2d  |
d d d | d2r d4r | d2 d4 | d2d | e4 e e | d2s d4 | e2 f4  | f2d  |
L: 1. Net soos ek is, geen hulp na-by, al-leen U bloed ge-stort vir my,
L: 2. Net soos ek is, nooi U vir my, ver-ge-we, rei-nig en be-vry.
L: 3. Net soos ek is, U lief-de teer werp al-le hin-der-pa-le neer.
a a a | g2s g4  | a2 g4 | a2d | b4 b b | c2n c4 | b2 e4  | ds2d |
d d d | d2 d4   | d2 g4 | f2d | e4 e e | a2 a4  | g2 g4  | b2d  |
```

There are a few new things: `Style: SATB` to indicate four-voice choral music on two staves, `Sharps` for the key signature (of course there is `Flats` available too), `Meter` for the time signature, and `Space: 9` to indicate that there should be nine interlines of extra space between the

[4] Rainer Dunker's `musixlyr` package, on which ***M-Tx*** depends, can now be used directly with **PMX**.

staves.[5] If there are more than two staves, you can prescribe a different extra distance for each on the `Space` line. This is the one preamble command that may also appear later in a music paragraph.

Among the notes, you notice suffixes `s` and `n` for 'sharp' and 'natural', and `r` for 'displace notehead to the right'.

But the main novelty is the lines of lyrics. The rule is: each lyrics line is identified by the label `L:`, followed by lyrics of the verse in question. Syllables are indicated by hyphens, and you don't use bar lines on lyrics lines. If more than one note belongs to a syllable, you need not worry, as long as the notes in question have been slurred or beamed together — each lyrics line is synchronized to the immediately preceding voice. There may be a set of lyrics lines for each stave in the score: if a stave has two voices, only one of them can have lyrics specified by `L:` — see Section 1.4 for a feature that allows both voices to have lyrics.

Note the verse numbers. The rule is: if you code a number followed by a point and a blank at the start of a lyrics line, this number will not be treated as a syllable but will be printed to the left of the first lyrics syllable: the position of that syllable will be the same as if the number was not there. This may not be just what you want: see the `musixlyr` reference manual (Section A) for commands that can modify this behaviour.

1.2 Bars and meter changes

The notation for bars is taken from Chris Walshaw's `ABC2MTEX` music language, with one small modification.

`|`	Normal bar line
`|]`	Thin-thick double bar line (as at the end of a piece)
`||`	Thin-thin double bar line (to separate sections)
`:|`	Left repeat
`|:`	Right repeat
`:|:`	Left-right repeat

Bar lines make life easier, but are normally optional. Only the normal bar line actually implies a bar separation: the double line and repeat signs may appear in mid-bar. The fancy bars must appear in the bottom voice and are optional elsewhere.

- If you do put them in, ***M-Tx*** will check that there are no shorter bars than the currently defined one. If a too-short bar occurs at the end of the final paragraph, ***M-Tx*** assumes you know what you are doing, and automatically redefines the meter without printing a new time signature. If it appears elsewhere, it is an error.

- If your piece starts with a pickup, it is defined by a bar line at the end of "bar 0", i.e. the incomplete bar containing the pickup. Even when there is a repeat sign after the pickup, so that you don't actually see a bar line in the printed music, you still need a bar line before the repeat sign, otherwise ***M-Tx*** cannot know where the pickup stops. The bar line defining the pickup is compulsory in the first voice found, and optional but recommended in the others. When there is a pickup, the meter is not redefined.

- If you do not explicitly end the piece with one of the other types of bar line, a thin-thick bar line is automatically provided: you don't need to code it.

- **PMX** allows you to put two voices on a single stave by giving the notes of the lower voice first, followed by `//` and the notes of the upper voice. This construct works in ***M-Tx*** too

[5] One interline is the distance between two lines in a stave. The normal distance between staves is five interlines, but sometimes extra space on a page is distributed between the staves: you may need to experiment.

when only one voice for the stave was specified in the style, but for just one bar at a time; such a bar *must* be terminated by a bar line. Use it only for very short stretches. You will probably need to set the octave in the first note using =.

- If you are using **PMX** features that are not recognized by ***M-Tx*** but simply passed through, you may need a bar line to tell ***M-Tx*** to which bar they belong.

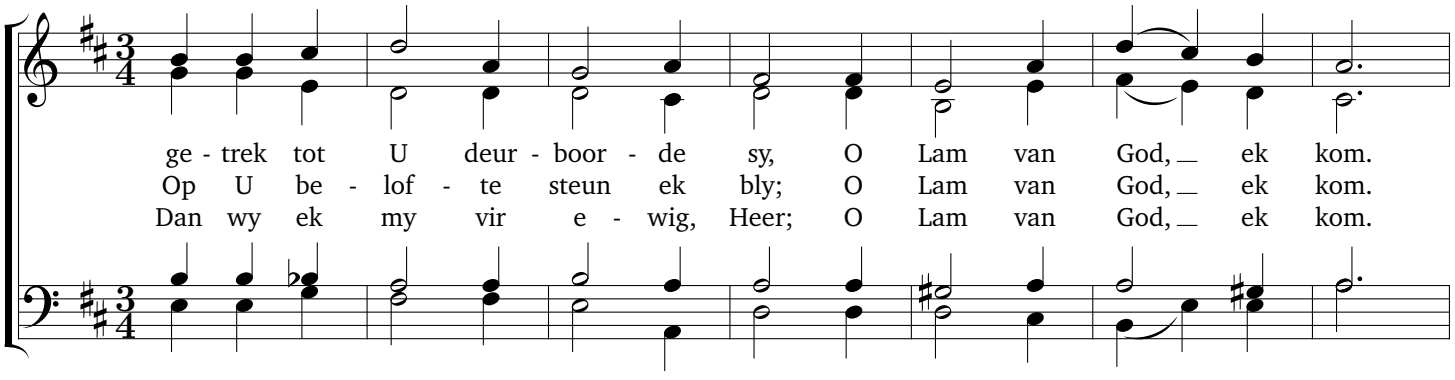
Apart from the final 'blind' meter change that is automatically made if required, you can also define other meter changes. The rules are:

- The meter change may only occur at the *very* beginning of a bar after a full bar of the meter previously in effect, and must be made in all voices. In particular, any key change, volta indication etc. must come *after* the meter change; also, the bar following the pickup may not contain a meter change but must have the length specified in the preamble.
- The normal meter change word looks like a fraction, e.g. `3/4`. This always gives a visible meter change.
- If the printed meter symbol looks different from the logical one (you want a big 'C' instead of '4 over 4', for instance), you may use a **PMX** formatted meter change word. E.g. `m3/4/0/0` gives a blind meter change — one that takes effect but is not printed. See the **PMX** manual. This is also legal in the preamble `Meter:` command. You may not introduce a meter change directly to **PMX** via a %% command — ***M-Tx*** must know about it too.

1.3 Beams and slurs

Thanks to **PMX**, beams in instrumental voices are automatic. ***M-Tx*** does, however recognize that it is customary in vocal music to use beams only when the notes in question are sung to the same single syllable. So if you select a style like `SATB` or `Singer` that involves voices, notes will normally appear unbeamed, except those that appear under slurs. To override this, you need to use the **PMX** "forced beam" feature, e.g. `[ c d ]`, or disable the `unbeamVocal` feature (Section 1.6). See Section 2.3 on how to fine-tune beams, and Appendix D.4 for some examples involving lyrics.

The slur notation is similar to forced beams, but uses parentheses. Look at the second system of the same song:



Omitting the preamble and **PMX** lines, this is coded as:[6]

[6]The coding is for this morsel only, not an extract out of the coding for the whole piece. The octave of the first note here is the default, not deduced from a previous note.

```
b4 b c  | d2  a4 | g2 a4  | f2 f4 | e2  a4 | ( d4 c ) b | a2d |
g4 g e  | d2  d4 | d2 c4  | d2 d4 | b2  e4 | ( f4 e ) d | c2d |
L: ge-trek tot U deur-boor-de sy, O Lam van God, ek kom.
L: Op U be-lof-te steun ek bly; O Lam van God, ek kom.
L: Dan wy ek my vir e-wig, Heer; O Lam van God, ek kom.
b4 b bf | a2  a4 | b2 a4  | a2 a4 | g2s a4 |   a2   g4s | a2d |
e4 e g  | f2 f4  | e2 a4- | d2 d4 | d2  c4 | ( b4 e ) e | a2d |
```

Apart from `f` for 'flat' you should notice the left parenthesis `(` to start a slur and the right parenthesis `)` to close it. So ***M-Tx*** knows that e.g. the `c` in the top voice falls under the slur, and does not use up a syllable of lyrics. The default direction of a slur is up for the upper and down for the lower voice in a stave, but you can override that: see Section 2.3.

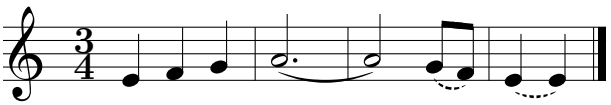
You are allowed to have a slur under another slur, e.g. `( c8 d e ( f g ) f d b ) c2`.

Occasionally a slur ends at a note and the next one starts immediately. For such a slur continuation you code `)(` as a single word after the note.

If you want a tie rather than a slur, use braces `{ }` instead of parentheses. A continuation tie is of course `}{`.

In obscure situations you may want absolute control over slur stops and starts. To do this, put an identifier, which must be a non-zero digit or a capital letter, immediately after the slur parenthesis or brace, before any adjustments. Of course, any labelled slur of either kind that you open must be closed again by its exact partner. If you really must have interlocking slurs, `(1 c8 d e (2 g g )1 f d b )2 c2` is the way to get them.

It may happen that you do not wish ***M-Tx*** to put more than one note to a syllable even though the notes are under a beam or slur. For example, you may wish to use phrase marks that only look like slurs, or you may be writing music that has so many consecutive short notes that the beat becomes difficult to follow without beams. The way to do this is to double the first symbol of the word starting the forced beam or slur. E.g. `{{+10` or `((` may be used to start a phrasing slur, and `[[` may be used to start a forced beam that will still keep one note to a syllable. The terminator for the beam or slur should not be doubled. A variation of this feature occurs mainly when you have multi-verse lyrics and there should be a slur in one verse but not in another. Start the slur with `("` or `{"` to get the same effect as `((` or `{{`, except that the slur symbol itself is no longer solid, but broken like a dotted line. Synchronize lyrics by using void syllables and extension rules, e.g.



She is a dan - gerous woman
You'll get a bro - - ken heart

```
e f g ( a2d a ) (" g8 f ) (" e4 e )
L: She is a dan-ge-rous wo-man
L: You'll get a bro--ken heart_
```

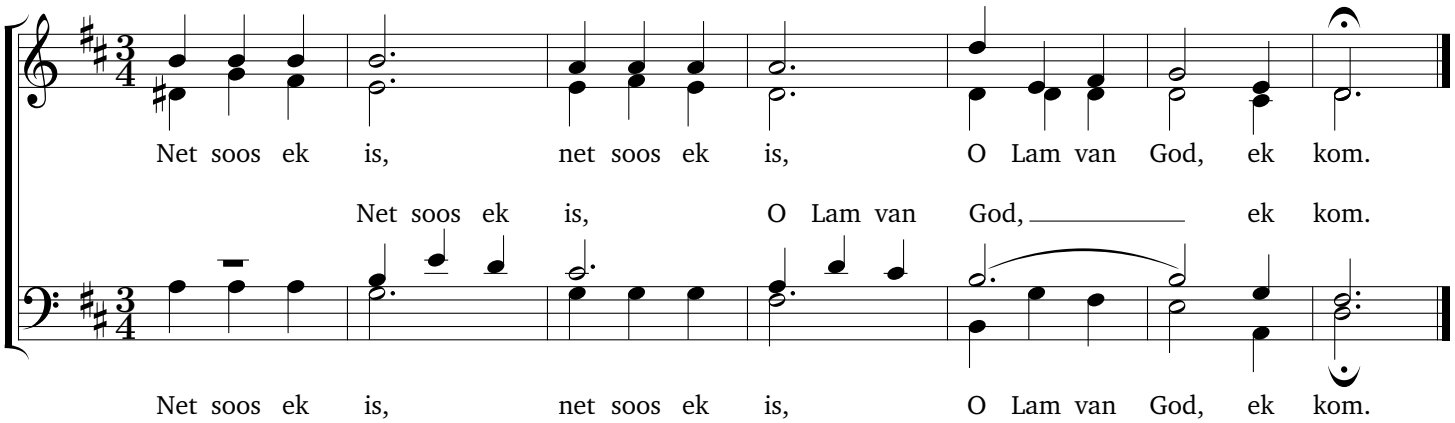
The trailing underscore on "heart" takes the place of a syllable of lyrics, and on a longer note would produces a lyrics rule that extends to the end of the melisma. You can use as many underscores as required; for very long melismas, you can code e.g. `Ah_6` instead of `Ah______`.

The converse of this situation is that you want ***M-Tx*** to make a melisma, but you don't want a visible slur. Code the slur beginning as `(~` or `{~` and the slur end as `)~` or `}~`. This gives a "blind" slur: invisible, but having the same effect on lyrics as a normal slur. Blind slurs should not have any other tuning marks.

1.4 More complicated lyrics

Lines of lyrics that start with `L:` are set in the middle between two staves (or below the bottom stave) and are aligned with the notes of the voice under which they appear in the input file (in this case, the alto). Such lines are referred to as *normal* lyrics lines.

Sometimes different words apply to different voices, as in the last line of the song:



This is coded as:

```
@+5 b4 b b  | b2d    | a4 a a | a2d     | d4 e- f  | g2 e4    | d2d of  |]
L: Net soos ek is, net soos ek is, O Lam van God, ek kom.
    d4s g f | e2d    | e4 f e | d2d     | d4 dr d  | d2 c4    | d2d     |]
@^+5   rp+6 | b4 e d | c2d    | a4 d c  | ( b2d | b2 ) g4    | f2d     |]
LT:           Net soos ek is, O Lam van God, ek kom.
    a4 a a  | g2d    | g4 g g | f2d     | b4- g+ f | e2 a4-   | d2d ofd |]
L: Net soos ek is, net soos ek is, O Lam van God, ek kom.
```

Some interesting **PMX** features in this piece are `of` for 'ornament: fermata' and `rp+6` (rest: pause) for a full-bar rest moved up six interlines. For the meaning of the `@...` words, see the next section.

The label `LT:` (Lyrics: Tenor) marks the line as belonging to a particular voice. Such a line is referred to as an *auxiliary* lyrics line, and may only appear on a stave that already has a normal lyrics line. Labels for voices are defined in the `Style`, or may simply be numbers: the voices are numbered 1,2,... from the top line downwards. Auxiliary lyrics for the top voice on a stave (soprano and tenor) will be set above the stave, and those for the bottom voice (alto and bass) below. Their input lines may appear anywhere in the paragraph, since the labels show where they go.

You can also use voice labels without a leading `L` to identify the voice lines. This is necessary if some voice is omitted from the paragraph (***M-Tx*** will fill its stave with rests) or if you prefer to have the voices in a different order (e.g. bottom-up) than top-down. You need only label a voice line if it does not belong to the voice immediately following the previous one.

1.5 Preamble commands

Here is a complete list of built-in preamble commands, with examples and explanations. Case in preamble commands is immaterial: all command and style names are translated to uppercase internally.

`Part: Recorder`	Part name (set flush left above title)
`Title: Clarinet Quintet`	Title of piece.
`Composer: Mozart`	Name of composer (set flush right below title)
`Poet: Rellstab`	Name of poet (set flush left below title)
`Meter: C/`	*Alla breve* meter: another notation for `2/2`.
`Flats: 3`	Key signature has three flats.
`Sharps: 2`	Key signature has two sharps.
`Space: 6 0 3`	Extra interlines of space below staves.
`PMX: h10i`	**PMX** command in preamble. Rather like %% feature.
`Options: x`	Forces the `x` option to be in effect, despite the command line.
`Enable: pedanticWarnings`	Enables the `pedanticWarnings` feature.
`Disable: unbeamVocal`	Disables the `unbeamVocal` feature.
`Pages: 2`	Set the piece over two pages.
`Systems: 11`	Use a total of eleven systems.
`Bars/line: 4`	Try to use on the average four bars to a line.
`Size: 16`	Size of music in points: default is 20.
`Style: Singer Piano`	The piece is for a singer with piano accompaniment.
`Name: Dietrich Gerald`	Names of instruments, performers etc.
`Indent: 0.10`	Indent first system by 10% of the music width.
`Start: @+1;@-3`	Put specified items at the start of voice lines
`Octave: 4 4 3 3`	Specify initial octaves for each stave

The `Bars/line` command should only be used when you do not yet know what `Pages` and `Systems` should be.

The style line may contain several style elements (instruments etc.). At present, ***M-Tx*** knows the following style elements:

```
SATB:    Voices S,A T,B; Choral; Clefs G F
SATB4:   Voices S A T B; Choral; Clefs G G G8 F
SINGER:  Voices S; Vocal; Clefs G
PIANO:   Voices RH LH; Continuo; Clefs G F
ORGAN:   Voices RH LH Ped; Continuo; Clefs G F F
SOLO:    Voices V; Clefs G
DUET:    Voices V1 Vc; Clefs G F
TRIO:    Voices V1 Va Vc; Clefs G C F
QUARTET: Voices V1 V2 Va Vc; Clefs G G C F
QUINTET: Voices V1 V2 Va Vc1 Vc2; Clefs G G C F F
SEXTET:  Voices V1 V2 Va1 Va2 Vc1 Vc2; Clefs G G C C F F
SEPTET:  Voices V1 V2 Va1 Va2 Vc1 Vc2 Cb; Clefs G G C C F F F
```

`Voices` gives the labels of the voices, later used to identify lyrics lines and out-of-sequence music lines. You should not use labels like `L`, `U`, `C`, `1` or `L1` that conflict with ***M-Tx***'s use of labelled lines for chords, lyrics etc. Labels separated by blanks belong to voices on different staves; labels separated by a comma, to voices on the same stave. `Clefs` define the clefs, one for each stave. You can use any PMX-supported clef symbol in addition to G and F — see Appendix B.3. The symbol `8` or `G8` indicates music written in the treble clef but sounding an octave lower, as is usual for the tenor voice in choral music.

`Vocal` means that the voices will be treated as vocal for the purpose of beams and lyrics, and `Continuo` means that the staves belong to a single instrument, and will be grouped together by a brace. You may not have too many voices or staves — see Appendix C.3.

`Choral` means that the voices form a choir. This implies that each voice is vocal, and their

staves will be grouped together by a bracket. You can get the bracket for instrumental voices too, by using `Group` instead of `Choral`.

You may define your own style elements, e.g.:

```
Honky-Tonk:  Voices R L; Clefs G F; Continuo
```

The whole preamble is read, and new style elements saved, before any of the commands is interpreted. So it does not matter in which order the commands come. The preamble may be spread over more than one paragraph. But if you issue a command more than once, only the last instance counts.

The `Start` command is mildly convenient when you are experimenting with fine-tuning, since it allows small additions affecting the beginning of each voice to be collected in one place. It comes into its own when combined with the multi-score option (see Section 3.8). Items for each voice are separated by semicolons and may contain blanks. These are prepended to the corresponding line of the first paragraph, so the *combined* length of the starting item and the line in question should not be too long (see Section C.3).

You don't normally need to specify `Octave`, since most of the time the octave implied by the clef is correct.

1.6 Customizing *M-Tx*

You have a lot of control over how ***M-Tx*** does its translation. The old way of customizing ***M-Tx*** was by compiler options and the `Options:` preamble command. That method uses one-character switches, which is only slightly mnemonic and anyway the letters get used up. The new way is to use the `Enable:` and `Disable:` preamble commands to select the features you want.

Here is a list of switchable ***M-Tx*** features that are enabled by default:

`splitShortcut` Separate 2:1 and 3:1 shortcuts into two words, and explicitly code their durations.

`newWordShortcut` Allow 2:1 and 3:1 shortcuts to be entered as separate words.

`multiFile` Take `Include:` lines into account, otherwise ignore them.

`doChords` Take `C:` lines into account, otherwise ignore them.

`doUptext` Take `U:` lines into account, otherwise ignore them.

`doLyrics` Take `L:` lines and inline lyrics changes into account, otherwise ignore them.

`unbeamVocal` Unbeam non-melismatic notes in vocal lines.

`hideBlindSlurs` Hide `{~` and `(~` slurs.

`interpretSlurs` Translate slur signs to PMX `s` and `t` slurs.

`uptextOnRests` Synchronize uptext with notes and rests, not only with notes.

Here is a list of switchable ***M-Tx*** features that are disabled by default:

`solfaNoteNames` Use tonic sol-fa note names `d r m f s l t` (for *do, re, mi, fa, sol, la, ti*). This will be translated to the usual note names `c d e f g a b` in C major. Use the **PMX** transposition feature to obtain the desired key. Only the basic note names are supported:

you should not use the sol-fa chromatic note names (e.g. *re, ma,*) but the basic sol-fa note names with **PMX** accidentals (e.g. `ds`, `rf`).

This feature conflicts in some cases with other uses of these letters. To retain the standard meaning in sol-fa mode, precede the word with a `"`, e.g. use `"r` for a rest since `r` translates to a `d`.

`pedanticWarnings` Suppress pedantic warnings, i.e. cases where you omitted something but the default action is probably what you want.

`ignoreErrors` Do not stop on the first error. (Errors are messages clearly marked with `ERROR` — any other message printed is informative or a warning only.)

`instrumentNames` Indent first system and print default instrument names.

`beVerbose` Verbose progress report. Default behaviour is to print out errors, warnings, and very little more.

`debugMode` Give messages that may be helpful in finding bugs in the `prepmx` program.

You can put as many features as you like on the `Enable:` and `Disable:` lines, just separate them by spaces. You can also have as many `Enable:` and `Disable:` lines as you want. For example, to imitate the way ***M-Tx*** 0.54c did things, put

```
Enable: interpretSlurs
Disable: newWordShortcut multiFile
```

2 Fine-tuning the printed output

The decisions made by **PMX** represent a good compromise in the majority of cases, but sometimes the first printout is disappointing. You can influence the details of the layout in various ways. You will probably need to look at the first draft, make fine-tuning changes, look at the second draft, . . . several times, if you are a real perfectionist.

2.1 General vertical and horizontal spacing

It is best to use `Bars/line` in the preamble for the first draft. You can then decide how many systems in total are required for proper horizontal spacing, and over how many pages they need to be spread for proper vertical spacing. On the later drafts you will therefore use `Pages` and `Systems` in the preamble.

There are **PMX** commands to put line and page breaks where you like — see Appendix B.

You can use `Space` to control the vertical space between the staves. There may be a number for every stave: the last number is interpreted as extra space below the bottom stave. You will probably need to specify it if you have lyrics or very low notes down there. This command can be issued in any music paragraph, not only in the preamble. It will take effect at the start of the next system. See Appendix D.1 for an example of how this command interacts with lyrics adjustments.

You can specify different sizes for the instruments, e.g. `Size: 13 16` might be appropriate for the main score of a violin sonata, with the notes of the violin smaller than those of the piano. Valid sizes are 13, 16, 20, 24 and 29. Size is a delicate issue: see Appendix D.2.

You should be aware of what the **PMX** manual calls a “benign bug”. Pages more than approximately half full get filled out by distributing the vertical space among the staves. This means that the appearance of the score may drastically change when you add one more stave. For this

reason it is best to leave space adjustments till last, when you already know what values to give for `Systems` and `Pages`.

Occasionally **PMX** underestimates the total height of a page and you end up with one system less than you wanted, which then appears by itself on a page of its own. This is particularly likely if you asked for extra space between the staves, or are using different-sized staves. A dirty trick that fixes this problem is to specify a little more height directly to MusiXTEX than you told **PMX**: see Appendix B.

2.2 Lyrics placement

Sometimes the default position for placing lyrics causes ugly clashes with note stems, beams, slurs etc. You can move lyrics up or down by inserting a word starting with `@` in the music line to which that lyrics line is attached. For example, `@-2` ('at minus two') indicates that the lyrics should be moved down two internotes. One internote is the vertical distance between e.g. a B and a C, i.e. half of one interline. The change takes effect at the start of the bar in which it appears, and remains in effect until further modified by another shift. Each voice has its own associated shifts: one for normal lyrics and a different one for auxiliary lyrics.

You can also say `@^-2` ("put the lyrics above the stave at two internotes below the default position") or `@v` ("put the lyrics below the stave at the default position"). These commands discard any shift that may have been in effect for that set of lyrics before.

Be warned that the default position itself is not fixed relative to the stave but depends on the general vertical layout. So if you change a layout parameter after having placed the lyrics 'perfectly', they may need a further adjustment.

Occasionally you need a syllable spanning two words. You can code a hard blank to string the words together, e.g. `Egit-to~ad I-si-de`, but it also looks good to use an underscore which will be translated to a link, e.g. `Egit-to_ad I-si-de` will become: Egit-to ad I-si-de. If the link gets in the way of letters with descenders, you can lower it, e.g. `pretty\_and smart` becomes: pretty and smart.

2.3 Beam and slur placement

The decisions made by **PMX** in choosing the height and direction of beams and slurs may differ from what you would like. The slur start character may have the suffixes `u`p, `d`own or `l`ower (the last two mean the same) to control the direction of the slur, and a suffix `+#` or `-#` to raise or lower the start of the slur by `#` internotes. The slur end character may also have an adjustment suffix. Remember to fine-tune both end points when you wish to move the whole slur up or down!

It is also possible to fine-tune the horizontal position. The most common case occurs when two notes on the same pitch are linked by a tie. In that case `{...}` give a neater result than `(...)`. See the **PMX** user's manual for details on other horizontal adjustments.

In the case of a slur continuation you should code the height adjustment *between* the two parentheses, e.g. `)-3(`. You must not have two adjustments.

Fine-tuning beams is all done on the beam start character. It may have the suffixes `u`pper or `l`ower for direction, and up to three `+#` or `-#` suffixes. The first raises or lowers the beam by so many internotes, the second makes the beam more or less steep, and the third also raises or lowers the beam, but this time in units of beam thickness.

2.4 Instrument names

Some scores have indications like **Violin** to the left of the first system. The easiest way to do this is to enable the `instrumentNames` feature, which uses the line labels as names, but you may not

like the result.

To do it yourself, use the `Name:` command in the preamble, e.g. `Name: Violin Piano`. The names can be any valid TEX expressions that have no spaces, such as `{\it{French~Horn}}`. Remember, it's one name per instrument, not one name per stave. If you have too few names, the remaining instruments will not be named. You can also use a single hard space (`~`) when some instrument in the middle should not have a name.

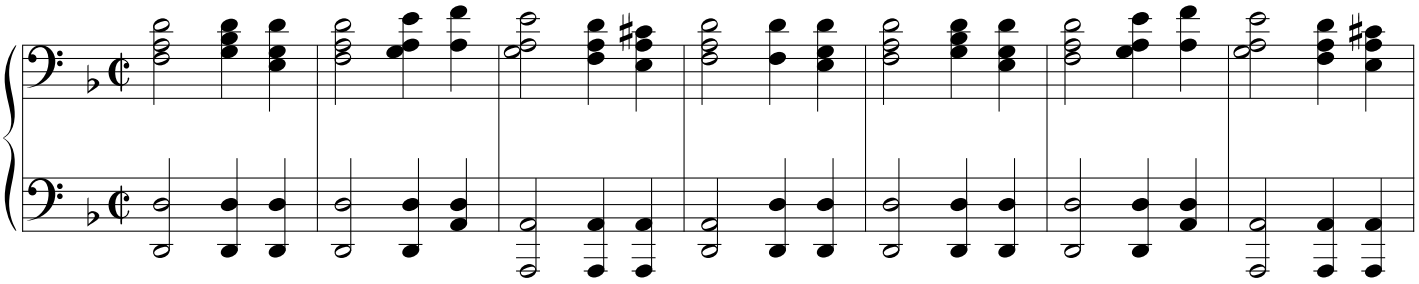
You will probably also need the `Indent:` command to specify the indentation required to provide space for the names. E.g. `Indent: 0.12` says the indentation should be 12% of the music width.

3 Shortcuts

This section introduces shortcuts for things that are perfectly possible in other ways, but a little clumsy.

3.1 Chords

An isolated chord in the middle of a melodic line is best treated by the **PMX** construction of chordal notes with the `z` prefix, e.g. `c ze zg zc` for a C major chord. Sometimes a whole string of successive chords appears, as in the introduction to Schubert's song *Der Tod und das Mädchen*. You can see that the coding for this can become quite cumbersome. Therefore ***M-Tx*** allows you to put the chordal notes on a second line, as shown below.

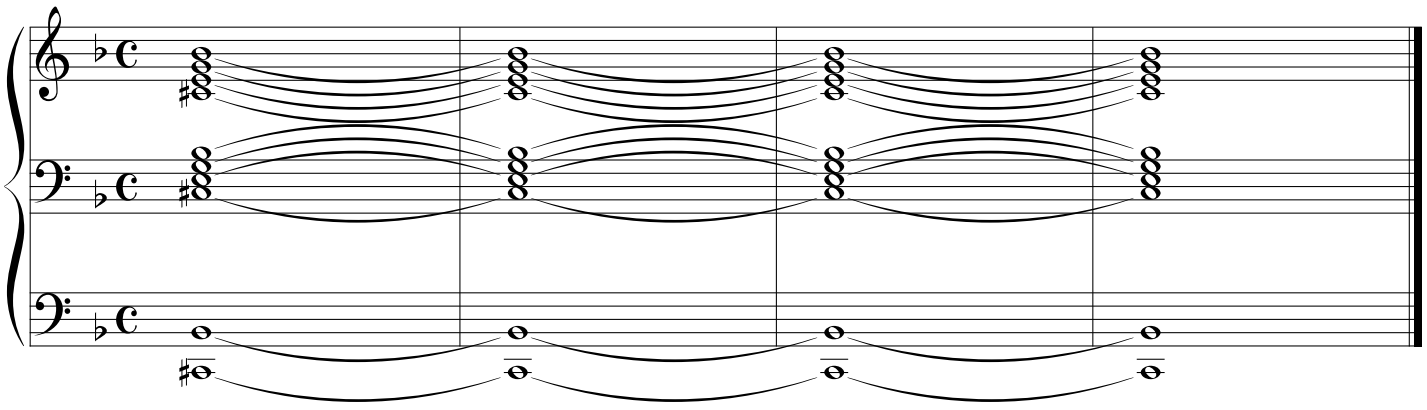


```
Style: PianoBass
PianoBass: Voices RH LH; Clefs F F; Continuo
Meter: C/
Flats: 1

   f2 g4 e   | f2 a4    a | a2  f4 e | f2 f4 e | f2 g4 e | f2 a4    a | a2  f4 e |
C: ad bd gd+ ad gle+ f+ gle+ ad acs ad d+ gd+ ad bd gd+ ad gle+ f+ gle+ ad acs
   d2 d4 d   | d2 d4    a | a2  a4 a | a2 d4 d | d2 d4 d | d2 d4    a | a2  a4 a |
C: d- d- d-    d- d-    d   a-  a- a-  d- d- d-  d- d- d-  d- d-    d   a-  a- a-
```

- There is a basic melodic line, with a separate line for the chordal notes, labelled like a lyrics line, but with `C` instead of `L`.
- For each chord there is one word on the chord line. If the chord contains more than one chordal note, the notes follow with no spaces in between. The notation is the usual **PMX** notation, with two changes: fla`t` and notehead `l`eft. These changes are used only on chord lines, and are unavoidable because `e` and `f` are needed for note names.

- The note chosen as melodic note on the main music line must not be a shifted note (i.e. one with displaced notehead): all shifted notes must be on the chord line.
- The stem direction of the chord is what that of the melodic note would have been in the absence of chordal notes.
- Pitch on the melodic line (i.e. the choice of the closest note with the given name) is not affected by the chordal notes. Each chord has its own local frame of reference, starting anew from its melodic note. This differs from the **PMX** `z...` notation, where the pitch of each note is determined by the previous note, whether chordal or not.
- Sometimes in the middle of a chordal passage there might be an isolated melodic note that does not have any chordal notes. To indicate this, the chord line must contain a spacer, which is a one-character word containing just a tilde (`~`).
- Sometimes you only need a few isolated chords. A one-character word consisting of a bar symbol `|` on a chord line indicates that no more chords appear in the current bar even though one or more melodic notes are still left. In that case you do not need a `~` for the 'missing' chords. Don't use bar symbols on a chord line just to separate bars when there are no missing chords.

  Two or more consecutive such words indicate one or more bars with no chords. If the very first word on a chord line is a bar symbol, the first bar has no chords.
- You can get an arpeggio sign (wavy line) in front of the whole chord including the melodic note by starting the chord word with a question mark, e.g. `?ad` instead of `ad` in the first chord above would have put an arpeggio sign in front of all three notes. For more complicated arpeggio constructions, you need to put `?` signs on the music lines themselves: consult the **PMX** manual.
- You can tie notes in chords. The notation is more compact than ties in the music line itself: put a `{` before each note that should be tied, no blanks, but don't put anything to close the tie. It is an error if you open a tie and there is no note at that pitch in the next chord. All the notes in that chord that close ties should appear before all the notes that open ties, e.g. `{c c{e` is OK but `{c {ec` is not. 0.60



```
Style: Organ
Meter: C
Flats: 1
```

```
{ cs0  c }{ c }{ c }
C: {e{g{b {e{g{b {e{g{b egb
{ cs0  c }{ c }{ c }
C: {e{g{b {e{g{b {e{g{b egb
{ cs0-  c }{ c }{ c }
C: {b+ {b+ {b+ b+
```

3.2 Expression marks and other annotations

Annotations and interpretation marks (***p***, ***f***, *rit.*, *cresc.* and so on) are inserted using lines like lyrics lines, but with `U` (for Uptext) instead of `L`. Note, though, that uptext lines are synchronized with rests as well as notes, whereas lyrics are synchronized with notes only.

The normal position for an uptext line is *above* the music line to which it applies: if it appears anywhere else, the label must include the voice name or number. I'll speak of an *uptext voice* to refer to the totality of uptext lines (no more than one per paragraph) associated with a particular voice.

- Spacers `~` and bar symbols `|` work for uptext in exactly the same way as for chords — see Section 3.1.

- Vertical adjustment signs starting with `@` like `@^` `@v` `@=4` `@+4` `@-4` may appear in uptext lines, and have the same effect on the position of the uptext as in the fine-tuning of lyrics — see Section 2.2. The only difference is that these signs appear in the uptext line itself, whereas the lyrics adjustments appear in the corresponding music line. A line starting `U: @v` is actually a downtext line! Each uptext voice retains its own adjustments which remain in effect even after the paragraph in which they appear.

- Uptext lines may also have horizontal adjustment signs, namely `@<` and `@>`. There are three possible positions for aligning uptext to a note: left, right and centre. The default position is to align to the right, but the horizontal adjustment shifts the position in the indicated direction, e.g. one `@<` causes centred uptext and another `@<` causes uptext extending to the left of the note. Yet a third `@<` will be ignored. Here, too, each uptext voice retains its own adjustments which remain in effect even after the paragraph in which they appear.

- Uptext lines are useful for indicating guitar chords. To make this easier, the sharp character `#` may be used on an uptext line as a normal text character (usually in TEX you need `\#` to get it) and the `%` character is used to indicate a ♭.

- The preprocessor tries to guess whether the word is a dynamic indication like ***mp***, ***sf***, ***rfz*** etc. and will use the MusiXTEX `\ppff` font for it. The current algorithm is highly unsophisticated: if all the letters in a word come from the list `fmprsz` it is diagnosed as a dynamic indication.

- A more permanent font change is indicated by a word like `!bf` or `!it` (no backslash — the `prepmx` program will put it in). Every word of that uptext voice (except `\ppff` words) will be set in the indicated font. Each uptext voice retains its own font which remains in effect even after the paragraph in which it appears.

- You can indicate crescendo and decrescendo signs in uptext by two methods:

  1. Put `<` or `>` where the sign starts, and `<.` or `>.` where it ends. This feature gives unpredictable results when you try to use it in more than one voice at the same time.

2. Put e.g. `<18` or `>7` where the sign starts. The number gives the length of the crescendo in elementary skip units or "elemskips".[7]

3.3 Lyrics paragraphs

You may prefer to have all the lyrics for a particular voice together in one paragraph. To do this, the first line of the paragraph consists of a name for the group of lyrics in braces, e.g. `{verse1}`.

When you later wish to use these lyrics, you put a list of names on a lyrics line where you would normally have put the lyrics, e.g. `L: {verse1,verse2}` replaces:

```
L: Lyr-ics for the first verse
L: Lyr-ics for the second verse
```

You do not repeat this instruction in the next paragraph, but if the voice concerned no longer has lyrics attached to it, you should indicate that by a blank lyrics line, i.e. `L:` by itself.

A particular lyrics paragraph name should not be used in more than one place. If two voices have exactly the same lyrics, you need only one lyrics paragraph, but more than one name. E.g. `{soprano}={bass}` makes two identical sets of lyrics, one to be used by `L: {soprano}` and the other by `L: {bass}`.

One useful property of lyrics paragraphs is to allow you to switch lyrics in the middle of a line instead of between music paragraphs. This is only allowed if the voice in question already possesses lyrics: it cannot be used to replace the `L:` line. E.g. `a b {chorus} c d` means that starting at the note `c` lyrics will be taken from the lyrics paragraph labelled `{chorus}`, and `a b {} c d` means that lyrics are switched off at that point.

3.4 Long/short note combinations

Passages containing dotted notes require duration changes for every note:

```
c8d g1 c8d g1 c8d g1 c8d g1 c4
```

There is a shorthand notation for this, namely:

```
c8.g c.d c.g c.d c
```

The first note in a dotted group may have an explicit duration, which then becomes the default duration, but the second must not. There is a similar shortcut `c..d` for double-dotted notes.

When the dotted note appears not in a group but on its own, as in triple and compound time (3/4, 12/8 etc.), you must use a `d` to indicate the dot, not a period.

Similarly, in compound meter passages containing long and short notes require frequent duration changes: `6/8 c8 g1 c8 d1 c8 g1 c8 d1 c8d` The shorthand notation for this reads: `6/8 c8,g c,d c,g c,d cd`

3.5 Barless music

Many old hymn tunes do not fit into the modern pattern of regular bars. You could get this effect by carefully counting the length of each music line and issuing your own blind meter change instructions.

Instead, specify in the preamble that you have zero beats per bar. E.g. `Meter: 0/4` means that each paragraph of music should be treated as a single bar, and that the basic counting unit is a quarter-note. You need not issue any other meter change instructions. ***M-Tx*** will check that each of the lines in the paragraph have the same total duration, which must a multiple of the specified counting unit.

[7] It is quite technical to define precisely how long an elemskip is. The important thing about them is that they stretch or shrink with spacing of the notes, so that if say a half-note is followed by two elemskips in one place, it will be followed by two elemskips everywhere.

3.6 Sticky ornaments and suffixes

You may need to mark a whole run of notes as staccato. To do so, put `o.:` instead of `o.` after the first staccato note and `o:` after the last one. This method also works with tenuto and all the other ornaments in B.3. The stickyness of ornaments lasts only till the end of the bar.

On notes and rests there are many possible suffixes, some of which you have already encountered and some of which are given in Appendix B. There are situations where you might wish to have several consecutive notes or rests with the same attribute. The method is similar: put a colon (`:`) after the letter involved on the first note of such a sequence, leave out the letter on all the notes except the last, on which you again put the letter. For example, in `12/8` meter you might have a passage moving at four beats to the bar, normally coded `c4d cd ed ed gd gd c2d`, which becomes `c4d: c e e g g c2d` when you use this shortcut. The stickyness of note and rest suffixes lasts until cancelled.

Each voice is treated on its own, and notes and rests are considered independently, so if you make the `d` sticky on a note it will not affect rests. My favourite application of this feature occurs in homophonic music with two voices per stave: I prefer to have only one rest per stave rather two rests, one on top of the other, and so I make all the rests in one of the voices blank by marking the first rest of that voice.

3.7 Multi-bar rests

It may happen that all the instruments in your score simultaneously have a rest of two or more bars. In that case a succession of empty bars may be difficult to count, and the conventional notation is to put all the rests in a single bar, with the total number of bars printed above them. The word for a multi-bar rest is e.g. `rm8` for a rest of eight bars. You code this for one voice only, and there may be no other notes or rests on that line. The same rest will be printed on all staves.

You can tune the appearance of the rest by appending a signed integer to the word, e.g. `rm19+18`. Roughly speaking, this moves the front end of the rest to the right by 18 points. Since multibar rests can have several shapes and may even contain as many as three separate symbols (in the case of `rm7`) it is difficult to describe precisely what happens and you are advised to experiment.

3.8 Skipping and including portions of a score

To read in another file at the start of a paragraph, use the "Include:" directive, e.g. `Include: mylayout.mti`. The effect is exactly the same as if those lines had been part of the original input file. The file thus included may also contain "Include:" directives; there is no restriction on the number of levels. You may re-include a file that was included before, but you may not include any file that has already been included but has not yet been fully read.

You may have various reasons for wanting ***M-Tx*** to omit one or more lines of your score. There are several ways to do so, of which the simplest is the comment symbol % that you have met before, which skips that one line.

A line starting with `SUSPEND` all in capitals, at the start of a paragraph, tells ***M-Tx*** to discard all following paragraphs until a line starting with `RESUME` all in capitals is found at the start of a paragraph. The material in between must still be organized into paragraphs that are not too large (see Appendix C.3) but need not in any other way conform to ***M-Tx*** input rules.

A line starting with `Only:` defines a subset of line numbers, e.g. `Only: 1,3-7,9` defines the set {1,3,4,5,6,7,9}. Only the corresponding lines from all non-lyrics paragraphs after the paragraph containing the `Only` are processed until another `Only` is found. Any paragraph containing an `Only` is always fully taken into account: it is not subjected to the masking action of any `Only`. *WARNING: this feature is deprecated and may behave differently in future versions.*

You can also conditionally ignore portions of the score depending on a compiler variable. The compiler options `0` to `9` identify ten special cases of a score. Together with the basic unnumbered version of the score, you can therefore have up to eleven versions. A line starting with `Case:` at the start of a paragraph gives a list of those cases that include the paragraph in question, e.g. `Case: 137` identifies a paragraph that is only included when one of the options `1`, `3` or `7` was selected, either as a command option or on and `Options:` line.

All paragraphs headed by `Case:` occurring *before* the first music paragraph are treated as extensions of the preamble. The very first music paragraph may therefore not start with `Case:`.

For example, the score `cervus.mtx` of Palestrina's four-part setting of Psalm 42 (see the `source` directory in the ***M-Tx*** distribution) is set up to give by default a score for the conductor with all voices in 16-point type. When one of the `1`, `2`, `3` or `4` options is specified, you get a score with one voice in 20-point type and the others in 13-point. When the `0` option is given, 20-point is used throughout and the score is spread over three pages.

3.9 Macros

0.60

M-Tx offers limited support for the **PMX** macro feature, which is a shorthand notation for pieces of text that appear often in your input. In **PMX**, you define macro number i by `MS`i and end it by `M`, each a separate word, where i is a number from 1 to 20. Everything in between (which may not contain any word starting with `M`) is not acted on, but only stored, but when you later "play" the macro by the single word `MP`i, the effect is the same as if you had typed the text you defined earlier. In other words, a **PMX** macro works as a purely literal text insertion. You can also start a macro definition by `MR`i : this has the effect of immediately playing the macro, not just storing it.

The default behaviour in ***M-Tx*** is to pass along such `M`-words without looking at them, which in expert hands can be quite effective, but for the rest of us is simply a disaster. The problem is that in that case ***M-Tx*** does not know that notes are being inserted, let alone what their duration is. It is in the position of a musician who has lost count.

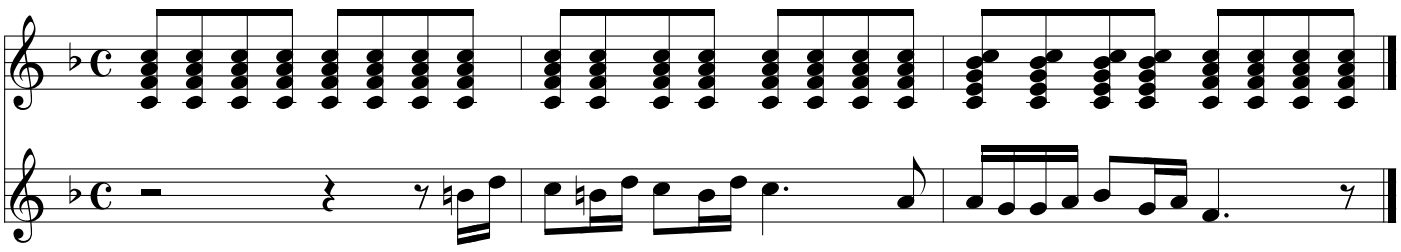
M-Tx offers two solutions to this problem, either of which you can `Enable` in the preamble. In both cases, macros may only appear on music lines.

`expandMacro` All macros are fully expanded at the ***M-Tx*** level. The **PMX** file will not contain any macros. If you use this feature, you can have macros numbered from 1 to 99, and macro definitions may contain plays of other macros. Error messages apply to the expanded text and may be difficult to understand.

`countMacro` `M`-words are still passed along, but when a macro is defined, ***M-Tx*** determines its duration and counts correctly when it is played.

If you have enabled `countMacro`, there is one unavoidable restriction: a macro definition or play may not spill over into the next bar. Hence the total duration of a macro may not be more than what remains of the current bar.

A word of caution: because you cannot see the macro definition when you play it, it is easy to lose track of duration and octave. It is therefore just sound common sense to set duration and octave at the first note in the macro definition, and at the first note after the macro play, to avoid mistakes.



```
Style: Woodwind Violins
Woodwind: Voices FlObCl; Clefs G
Violins: Voices Violini; Clefs G
Flats: 1
Meter: C
Enable: countMacro

MS1 c8+ za zf zc M [u MP1 MP1 MP1 MP1 ] [u MP1 MP1 MP1 MP1 ]
r2 r4 r8 b1n d

[u MP1 MP1 MP1 MP1 ] [u MP1 MP1 MP1 MP1 ]
c8 b1n d c8  b1 d c4.a

[u MR3 c8+ zb zg ze zc M MP3 MP3 MP3 ] [u MP1 MP1 MP1 MP1 ]
a1 g g a b8 g1 a1 f4d r8
```

If you used `expandMacro` instead, the first music line could have read

```
MS1 c8=+ za zf zc M MR2 [u MP1 MP1 MP1 MP1 ] M MP2
```

in which case the first line of the second bar would only be `MP2 MP2`. This is shorter for you to code, and in this case it works. It is not always feasible to use macros within macros: `expandMacro` works by actually inserting the strings into ***M-Tx*** input lines, which may not be longer than 255 characters. Too many macro expansions, and you hit your head against the ceiling.

4 Acknowledgments

I still have the `.tex` and `.dvi` files of my first attempt at setting music (Bach, no less) with the MuTeX package of Andrea Steinbach and Angelika Schofer. The first phrase goes:

```
\4\,\F\~\B\.\c\|\2\A\.\|\4\B\,\E\E\| \2\group{\\\E\\\d}{\\0\\{-1}}\lslur12\go\|
```

In ***M-Tx*** that would be

```
f bd c8 | a2d | b4 e- e | ( e2 d4 )
```

A tantalizing glimpse of a preprocessor appeared in the MuTeX documentation, but the software for it was never released.

None of this would have been possible without the late Daniel Taupin, who was the first to emphasize that printed music has a fundamentally two-dimensional layout, and whose MusiXTeX package is the engine that actually performs the typesetting.

I made a previous attempt to implement something like ***M-Tx*** as a direct preprocessor to MusicTeX (Daniel's first package). My lack of expertise caused that project to die. The appearance in January 1997 of **PMX** 1.1 inspired me to resurrect the idea — thanks to Don Simons. Don has

also made some comments which helped me to improve the ***M-Tx*** input language. Some ***M-Tx*** features that are implemented by using TEX directly, such as multi-bar rests in scores with more than one stave, make use of MusiXTeX code written by Don.

Since version 0.21, ***M-Tx*** uses Rainer Dunker's `musixlyr` package. This gives much neater horizontal spacing of lyrics than the manual system used in previous versions, and virtually eliminates the need for horizontal tuning.

Chris Walshaw's `ABC2MTEX` provided the repeat notation.

The layout of ***M-Tx*** bears some resemblance to a language developed by Miguel Filgueras, not yet publicly available in software, although I did not consciously borrow his ideas.

I had some enlightening e-mail correspondence with the developers of MPP, Jan Nieuwenhuizen and Han-Wen Nienhuys. This package (no longer being maintained, unfortunately: the authors are working on something altogether grander called LilyPond) was the first music package which made me believe that one could write good-looking scores in a reasonable amount of time.

Eric Petersen, Reenen Laurie, Kestell Laurie and Joel Hunsberger found several bugs in earlier versions of ***M-Tx***. Christian Mondrup did this for ***M-Tx*** 0.50 and later. The subscribers to the discussion list `tex-music@sunsite.dk` — too many to name — found many more and contributed to the development by requesting new features. Details of these contribution are given in the `Corrections` file distributed with the `prepmx` code.

The success of public domain music publishing owes an incalculable debt to the late Werner Icking. He maintained an online archive for music software and sheet music, set standards for music typesetting, tested software packages, gave genuinely expert advice to would-be experts, and encouraged newbies with patience and understanding. When he died unexpectedly at a comparatively young age, each of the small but globewide community of users of MusiXTEX lost a personal friend and mentor. His music archive as it was at the time of his death can still be accessed at `ftp://ftp.gmd.de/music` or `http://www.gmd.de/Music`.

Werner's work is still being carried on, but it requires three people to fill his shoes. Maurizio Codogno is the mailing list administrator, Christian Mondrup the maintainer of the software archive, and Don Simons the maintainer of the sheet music archive.

My wife, Trienke, has had to suffer many effectively husbandless evenings and weekends while I hammered the program and documentation into good enough shape to be able to offer it around.

A Where to find help

M-Tx is the tip of an iceberg. Things can go wrong at many levels. Fortunately, here are many ways to get information and help.

- Consult the Table of Contents and *index* in search of a likely-looking topic.
- The annotated examples might be useful.
- There is a file `FAQ` in the distribution that contains the sort of questions that new users often ask.
- The packages on which ***M-Tx*** relies all have their own reference manuals. Look for the following files:

  PMX: `pmx230.pdf`, `ref230.pdf` (these names will obviously be different if your **PMX** is not Version 2.30);

  `musixlyr`: `musixlyr.pdf`;

MusiXTEX: `musixdoc.dvi`.

- Consult the mailing list archive at

  `http://icking-music-archive.sunsite.dk/mailinglist/archive`

- Join the mailing list by sending an e-mail message to

  `tex-music@sunsite.dk`

  with the word `subscribe` as subject — no message body needed.

B A short PMX tutorial

This section describes the parts of PMX that you are likely to need when coding scores in ***M-Tx***. I have left out or simplified features that are not necessary or too advanced. In most cases, you can still use them, as ***M-Tx*** simply passes through anything that it does not recognize.

PMX resembles English in that it is made up of words, which are strings of non-blank characters separated by blanks. Each word, though, has the character of an acronym, in that its letters each stand for something longer, and in fact Don Simons designed the notation so that you can actually say out loud what the acronym stands for.

In the following, the symbol `#` *stands for some number.*

B.1 Words that contribute to the count

Each bar of music has a certain duration, which is is the total of the notes and rests in it. The word for a note that contributes to the count starts with `a` to `g`, which indicates pitch; and the word for a rest starts with `r`.

A note may have a number of suffixes, which may come in any order. The duration is indicated by a digit: from short to long, these are 6 (1/64-th note), 3, 1, 8, 4, 2, 0, (whole note), 9 (double note, also known as 'breve'). Moving a note an octave[8] up or down is indicated by `+` and `-`, repeated if necessary. The other suffixes are the following marked letters: dotted, flat, sharp, natural, stem upper, stem lower, offset to the right, offset to the left, apart (not attached to a beam).

You can fine-tune dot position, e.g. `gd-2` puts the dot two internotes lower than it would normally go, and `gd-0+1` puts it one notehead to the right.

You can also fine-tune accidental position in a similar way. For accidentals (but not for dots) you are allowed to write `gs>1` instead of `gs-0+1`.

A rest may also have suffixes: duration and dot as for notes, pause (duration is the whole bar), blank (rest is counted but not printed). It can be moved up and down by `+` and `-` followed by a number, e.g. `r0-4` is a whole-note rest moved down four internotes. The other rest suffixes mentioned in Section 3.7 are not yet (version 1.3.8) part of **PMX**, except `m`, which is allowed in a score for a single one-stave instrument. Also the 'blind' and adjustment suffixes on the pause `rp` are legal in ***M-Tx***, but not yet in **PMX**.

The duration of a note may be influenced in various ways by a previous note: if unspecified, it is the same as the previous note; or the note may be part of a chord, a multiplet, or a grace note group.

[8] The `=` suffix is not part of **PMX**. Instead, **PMX** offers the ability to specify octaves by numbers. *This feature is not supported by* ***M-Tx***.

B.2 Words excluded from the count

Except for the first note of a chord or multiplet, the following words for notes are excluded from the count, and should not have a duration suffix.

Chords: Each note in a chord except the first has the prefix `z`. It is excluded from the count, and suffixes `a d l u` may only appear on the first, non-`z` note. The "melodic note" of Section 3.1 becomes the first note of the chord when the ***M-Tx*** code is translated to **PMX**.

Multiplets: A normal note may have an extra suffix, after all the others, of the form `x#` or `x#n`, where `#` stands for a number. This means that the note is the first of a multiplet (or x-tuplet) with `#` equal notes in the time of one. Only the duration coded on this first note is included in the count. The appearance of the first note will not be the normal one for its specified duration: it will look like all the others. E.g. `g4x3 a b` defines a triplet of total duration a quarter-note looking like three eighth-notes. The `n` means that the number `#` is not to be printed.

Grace notes: A normal note may have a prefix group starting with `G#` or `G`. This indicates that the note is the first of a grace note group with `#` notes, to be set in a tiny font. All the notes in the group, even the first, are excluded from the count.

B.3 Words for other things than notes

Other things than notes go into a music score: ornaments, clefs, expression markings, changes in key and time signature, etc. **PMX** contains support for many of these. Here I mention only a small selection: see the **PMX** manual for items you need but which are not described here. A word for each such feature has its characteristic first letter followed by a selection of suffixes that qualify it.

Ornament: An ornament is set off-stave in line with its note, and its word, starting with `o`, comes directly after that of its note. The more commonly needed suffixes are `u` (dot for pizzicato), `p` (prime for sharp pizzicato), `f`ermata above, **fd** (`f`ermata `d`own). The suffixes `.` (staccato) and `-` (tenuto) differ from the others by being set close to the note, not off-stave. There are several more in **PMX** and many others in MusiXTEX.

Clef change: You get a small clef at the point where the clef change occurs and a big one at the start of the next line. The clef change word begins with `C`. The suffix `#` places middle C on line `#` of the stave, where `1` is the bottom line and `5` the top. An appropriate clef is selected automatically, e.g. `C0` is the G-clef (treble) and `C6` is the F-clef (bass).

Voltas: These are the labels to mark sections of the score that are not played at every repeat. You put `V1` to indicate that a bracket containing the number 1 should appear above the bar in question. Since **PMX** wants all such indications in the bottom voice only, they are deleted by the ***M-Tx*** preprocessor if found elsewhere. See Appendix D.1.

Key change: Put `K+0-2` in a music line where a key change to two flats should occur. As with voltas, this indication is deleted if not in the bottom voice.

B.4 Useful things to put on %% lines

Most of the following commands should appear in the first music paragraph or on a `PMX` preamble command line. Some of them are actually raw MusiXTEX — consult the MusiXTEX manual

for other interesting commands. %% lines in the preamble are treated a little differently – see Appendix C.4.

Line, page and movement breaks: Put %%L6 above a paragraph to make the 6th system start there, and %%L6P2 to put that paragraph at the top of page 2. You can't ask for a page break without a line break. Use this feature only when necessary. Put L10M+10i0.1 to make a new movement start at the 6th system, with 10 internotes of extra space between movements and an indentation of $0.1 = 10\%$ of the width.

h260m Height is 260 millimetres.

w8i Width is 8 inches.

\\vsize 270mm\ Height given to MusiXTEX is 270 millimetres.

Ar Use relative accidentals. This requires a different way of coding flats and sharps —think of s as meaning 'sharpened' instead of 'sharp'. E.g. if the key signature has one or more flats, then bs means a B natural, bn a B flat, and bf a B double flat. If you need to transpose music often, it is a good idea to acquire the habit of using relative accidentals always. If you don't, only pieces in C major or A minor can be transposed without trouble.

As Always use small accidentals. Normally **PMX** uses big accidentals and only switches to smaller ones if there is a reason to do so. You can also force all big accidentals by Ab.

Ar K+1-3 Use relative accidentals and transpose the piece up one internote, to a key with three flats. You can only transpose correctly when using relative accidentals!

\\stdbarrules\ This will cause bar lines to be drawn solidly from the top to the bottom of each system, maybe cutting through lyrics in the process.

Midi commands **PMX** can make MIDI files. For example, if you put

```
PMX: Ii55:55:55:55t96b56:72:48:80v127:100:73:46
```

in your preamble, a MIDI file will be generated in which all four instruments are played on MIDI instrument number 54 (if you use the default instruments together with Timidity++, this means 'voices') at a tempo of 96 quarter-notes to a minute, with MIDI balances of 56, 72, 48 and 80 (centre is 64) and MIDI velocities of 127, 100, 73 and 46 respectively. More details appear in the **PMX** manual.

C How to get and use *M-Tx*

Sorry, this section assumes that you have some knowledge of computers! Get a TEX-pert to help you if necessary.

C.1 Installation

The primary official source for all the software you need is the MusiXTEX Archive at

```
http://icking-music-archive.sunsite.dk/mailinglist/archive
```

in the /music/musixtex/ directory and its subdirectories. Copies spread from there to other official CTAN (Comprehensive TEX Archive Network) sites. ***M-Tx*** is in the software/mtx subdirectory. You might find there a binary package suitable for your system; if so, you can ignore the part of this section that discusses compilation.

Read the README file that comes with the ***M-Tx*** zip file.

If you are still reading on despite the previous line, here is a general description — but the most recent stuff, the real nitty-gritty, is in the README file.

To use ***M-Tx***, you need a working **PMX** installation (version 2.502 or later, although many scores will work with earlier versions) which is in the software/pmx subdirectory of the Archive. **PMX** itself is a preprocessor to Daniel Taupin's MusiXTEX (Version T.112 was used to prepare this manual), which is in the MusiXTEX Archive directory itself. All of these depend on Donald Knuth's TEX typesetting package. If your system does not already have TEX, put pressure on your system administrator.

You will need Rainer Dunker's musixlyr extension package (version 2.0c or later). This package is available with full documentation in the add-ons subdirectory of the Archive, and you should read that documentation if you need more lyrics features than those accessible from ***M-Tx***.

My ***M-Tx*** translator was originally written in Turbo Pascal, but I use the Free Pascal compiler, available for many systems, for current development work. For the sake of those users whose systems do not yet run Free Pascal, but have a C compiler that accepts ANSI C source code, the Pascal source distribution provides the possiblity of making a C translation, using Dave Gillespie's Pascal-to-C converter p2c and a bit of post-translation hacking.

Once you have all the software, make an ***M-Tx*** file with extension .mtx using your normal text editor. It must be an editor that respects the line boundaries and empty lines that you type in: in other words, something fairly primitive, not an upmarket word processor. Run the program prepmx with argument basename when your file is called basename.mtx. The output will be a file basename.pmx which then goes through **PMX** and TEX. The error detection capabilities of prepmx are reasonable but not extensive: much is passed through unexamined to **PMX**.

C.2 The prepmx command line

You could also supply other arguments on the command line. A complicated command line in Unix (replace / by \ in DOS) might be

```
prepmx -bvn mysong /home/dirk/texinput /home/dirk/mtx/mystyles.txt
```

This line contains switches (recognized by the leading hyphen), a basename, the directory where the final .tex file will be put (if you omit it, files go in the current directory), and the file where I keep my own styles (if omitted, the file mtxstyle.txt in the current directory is used, if there is one). Instead of putting the switches as here on the command line, you can specify them in a preamble Options: line, which takes precedence over the command line.

-b Disable unbeamVocal.

-c Disable doChords.

-D Enable debugMode.

-f Enable solfaNoteNames.

-i Enable ignoreErrors.

-m Disable doLyrics.

-n Enable instrumentNames.

`-t` Disable `doUptext`.

`-u` Disable `uptextOnRests`.

`-v` Enable `beVerbose`.

`-w` Enable `pedanticWarnings`.

`-0`, `-1`, ..., `-9` Select specified case of multiple score. See Section 3.8.

The `prepmx` program exits with return code 0 if no error was found, and return code n if an error was found while processing line n of the input. Return code 10000 means that the input file was empty or could not be opened, or that an invalid option was encountered. In the latter case, a list of available features is displayed — this is a quick way of obtaining such a list.

In some cases, the actual error may be on another line, e.g. when it is found that music lines have a different duration, the first line processed may be in error, but the discrepancy is only found later.

C.3 Bugs, restrictions and incompatibilities

The status of bugs changes too quickly for this document. Please consult the files supplied with the package, in particular `README`, `Corrections` and `Bugs`.

In ***M-Tx*** 0.60 a paragraph may not be more than 100 lines long, an input line (including those in lyrics paragraphs) not more than 255 characters, a lyrics line in a music paragraph not more than 128 characters, and you may define a maximum of 12 extra style elements of your own, whether in the preamble or in the `mtxstyle.txt` file. It is possible to change these easily: see `README`.

You can have up to 15 voices, using up to 15 staves. For each two-voice stave, the number of staves you can have goes down by 1.

Other limitations are described in the **PMX** documentation.

C.3.1 Compatibility

In ***M-Tx*** 0.55 a small but wide-ranging change has been made in `mtx.tex` which affects the `\musixlyr` interface and vertical spacing. The net result is that new scores will need much less tweaking with `Space:` and `@`. On the other hand, old scores that have laboriously been manipulated into correctness will now look wrong unless you put an old copy of `mtx.tex` where it will be found first, preferably in the same directory as the ***M-Tx*** score.

M-Tx 0.55 takes advantage of the new slur syntax of **PMX** 2.502. In some cases the slurs will look a little different, because `{...}` now generates ties.

Scores made using ***M-Tx*** 0.52 and later should still compile under ***M-Tx*** 0.60. Older scores may need to be revised.

Some command line options allowed in ***M-Tx*** 0.52 have been removed.

The ***M-Tx*** 0.52 documentation stated falsely:

> The default duration at the completion of a line is that of the last note appearing at that stage, which in the case of an expanded group is the shorter note. Since this is opposite to what happens during the line itself, a line ending with a group should have an explicit duration code for the first note on the next line for that voice.

Actually, duration is taken from what you code, so e.g. `c4.d` will leave the duration code as `4`. Of course, explicitly coding the duration can never hurt.

C.3.2 Unsupported features of MusixTEX and PMX

Unlike MusiXTEX itself, which is stable and nearly complete, **PMX** and ***M-Tx*** are both still in a stage of growth. Their authors are reasonable people who subscribe to the discussion list `tex-music@sunsite.dk` and are open to suggestions made there. If you have any ideas, please email them to the list, so other users can see them and comment on them.

The implementation of new features depends on three factors: what I need in my own scores; what users of ***M-Tx*** have asked me to do; and how easy and quick it is to do them. But bear in mind that the underlying philosophy of ***M-Tx*** is to be simple and intuitive: arcane features are best left to **PMX** if not MusiXTEX. I don't mind how sophisticated the **PMX** code generated needs to be, but the ***M-Tx*** input language must not acquire a cluttered and obscure appearance.

Support or otherwise of MusixTEX and **PMX** features comes at three levels:

1. The most frequently used features are transparently incorporated into the basic ***M-Tx*** language.

2. The default action of ***M-Tx*** is to pass through unchanged anything it does not recognize. Most **PMX** and MusiXTEX features still work in this case. For example, you can use `\PED\` in front of a note to obtain a pedal annotation below the stave, and `\DEP\` to get the big asterisk for pedal release. This is an example of a "Type 1 TEX string" as described in the **PMX** manual.

3. When a new **PMX** feature affects aspects like count or pitch of which ***M-Tx*** should be aware, it is likely to be incompatible with the preceding release of ***M-Tx***, although I do try to catch up later.

Some pertinent aspects:

1. ***M-Tx*** 0.60 does not recognize the macro features of **PMX**. You should therefore not use macros for notes or anything else that affects the count, the duration, the current octave and other properties that are inherited by a note from the next. But there is no harm in using macros for things like extra space.

2. ***M-Tx*** 0.60 is not aware of movement breaks and the possibility to change the number of voices.

C.4 For PMX- and MusiXTEX-perts only

You can send commands directly to PMX. All lines starting with `%%` in music paragraphs are collected, stripped of the leading `%%` and passed otherwise uninterpreted to **PMX** before anything else from that paragraph is translated. If such a line is found in a preamble paragraph, it is treated as a "Type 4 TEX string", TEX string" which means that it will eventually appear at the top of the TEX file, after `musixtex.tex` and `pmx.tex` have been read in. Such lines should *not* end with `\`. An example appears in Appendix D.5. As in TEX, a line starting with only one `%` is treated as a comment: such lines do not appear in the **PMX** file.

If you ever need to pass through something involving stave or instrument numbers, remember that **PMX** and MusiXTEX number these from bottom to top, not from top to bottom, and that the "first voice" in **PMX** is the *bottom stave* to ***M-Tx***.

Apart from the conversion of lyrics and chord lines into **PMX** commands (which is the main job of ***M-Tx***) the notes are modified in various ways.

- Octave is inserted into the first note of each voice, depending on the `Octave` command, or if none is supplied, on the clef. Absolute octave numbers are not recognized by ***M-Tx*** and

may cause unexpected side effects. You can get something like absolute octave numbers by combining = with - or -. This may cause an irritating but harmless error message about backward incompatibiliy at the PMX stage.

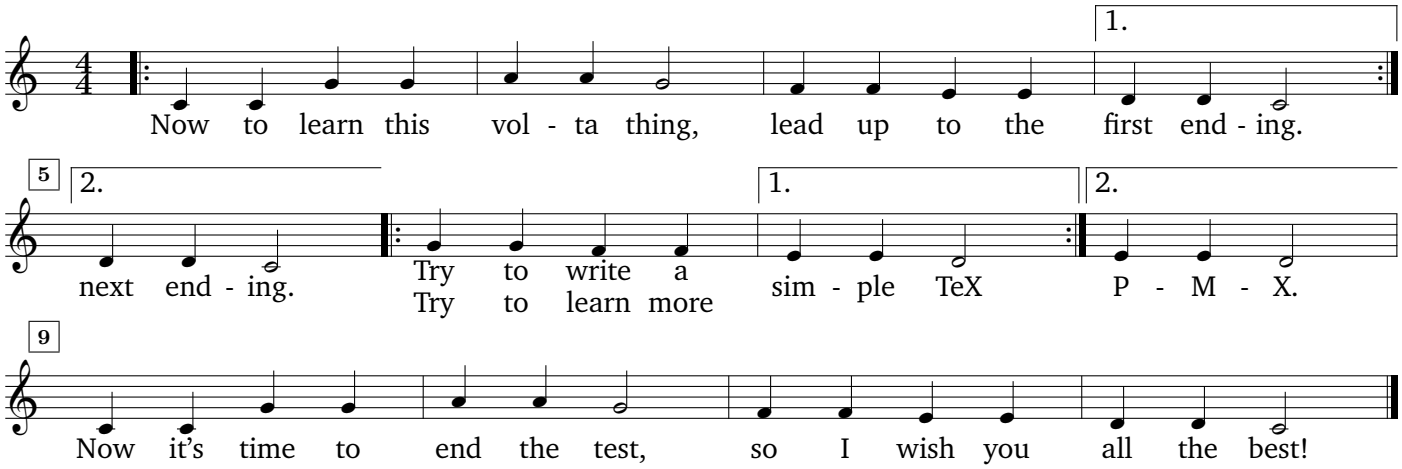
- If a line is vocal, an `a` is inserted into quavers and shorter notes when not under slurs, to protect them from being beamed.
- Duration codes are inserted into all notes and rests before any serious processing is done.
- Long/short note groups are expanded into separate notes. Although **PMX** allows these shortcuts, lyrics cannot be properly synchronized unless the groups are expanded.
- **PMX** syntax allows great freedom in the order in which different parts of a node are coded. ***M-Tx*** is not aware of all the subtleties, but extracts only those parts of which it needs to be aware, and reassembles the note afterwards.
- Labels in the range `I` to `T` are inserted into unlabelled slurs and ties. You should therefore avoid those labels for slurs that you label yourself.
- Chord lines are expanded into standard **PMX** code and inserted into the main line for each voice.

Other ways in which an ***M-Tx*** score differs from a native **PMX** score in more than layout are: a bar line is *required* for a pickup; repeat bars are coded by the ABC2MTEX symbols instead of `Rl` etc.; you don't need to insert a meter change for an incomplete final bar: ***M-Tx*** does it.

All the ability of **PMX** to pass through TEX (including MusiXTEX) commands is still there. You might for example like to read the `musixlyr` documentation for a large variety of ways to fine-tune horizontal lyrics placement, but I can promise that you hardly ever will need to do horizontal tuning except to align multi-verse lyrics as in Section 2.2.

D Annotated examples

D.1 Voltas




```
Meter: 4/4
Style:  Singer
Bars/Line: 4
Size: 16
Space: 6

@+4 |: c4 c g+ g | a a g2 | f4 f e e | V1 d d c2 :|
L: Now to learn this vol-ta thing, lead up to the first end-ing.

Vb2 d4 d  c2  |:
L: next end-ing.

Vx g4+ g f f |
L: Try to write a
L: Try to learn more

V1 e e d2 :|  Vb2 e4 e d2 |
L: sim-ple TeX   P-M-X.

Vx c4 c g+ g | a a g2 | f4 f e e | d d c2 |]
L: Now it's time to end the test, so I wish you all the best!
```

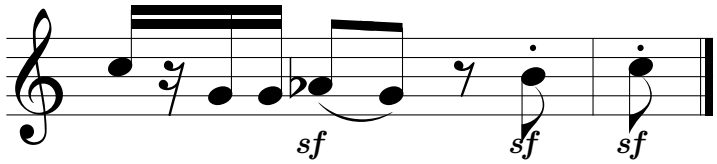
The main rule is: put every `V` command in the *bottom* voice at the start of a measure, after the bar line (if any) but before the first note, and don't try to use more than one at a time: the new volta automatically closes the old one. Volta signs in other voices are silently ignored. When reading the **PMX** manual, remember that *your* last part will become the first part for **PMX**.

In the above example `V1` starts a volta box, but to get the first volta box to be closed at the end, you must later say `Vb`. Also, to get the text `2` into the "second ending" box, you add the `2` after the `Vb` to make `Vb2`. Finally, to make the second box open ended, you must enter `Vx` at the point you want it to be done. (If you forget to enter the `Vx` after your last volta box, you will get a strange unwanted volta box at the end of your piece.)

This example also illustrates the interplay between `Space` and `@` required for proper spacing of lyrics when there is stuff above the staves. — *Contributed by Joel Hunsberger*

D.2 Music size

```
Bars/line: 2
Style: Solo
Meter: m3400
Size: 29pt

%% w3.7i
U: @<v ~ ~ ~ ~ sf ~ ~ sf sf
[ c1+ r g g ] ( [ af8 g ] ) r b o. | c o.
```



The above does not look quite right, for a complicated reason. A piece as a whole has a basic music size, and then each stave can have its own "local" size. The local size affects everything on the stave itself, but not the annotations, ornaments etc. The range of legal basic sizes is quite small: **PMX** allows only 16-point and 20-point. In this example, the basic size is 20-point and the local size 29-point.

To get the perfectly matched annotations displayed on the title page of this manual, the input looks like this:

```
Bars/line: 2
Style: Solo
Meter: m3400

%% w3.7i \\Largemusicsize\
U: @<v-2 ~ ~ ~ ~ sf ~ ~ sf sf
[ c1+ r g g ] ( [ af8 g ] ) r b o. | c o.
```

MusiXTEX has four size commands relative the basic size (called `\normalsize`). These commands affect all music fonts. They are:

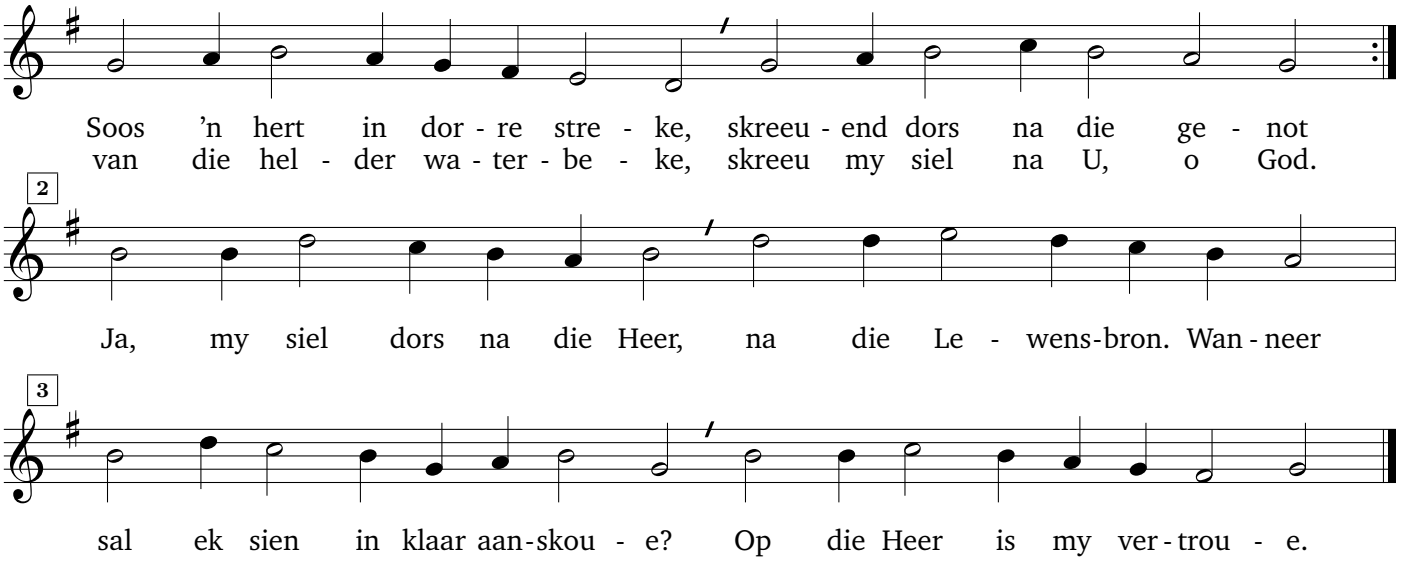
`\smallmusicsize`	13pt	16pt
`\normalmusicsize`	16pt	20pt
`\largemusicsize`	20pt	24pt
`\Largemusicsize`	24pt	29pt

There is no `\tinymusicsize`, although several of the music fonts are individually available in 11-point size.

D.3 A psalm tune

Psalm 42

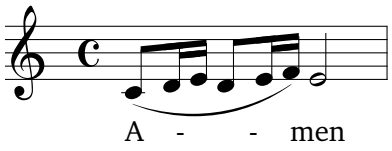
Ainsi que la biche rée



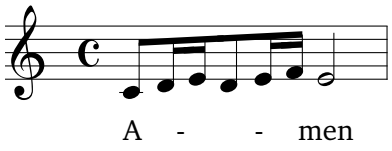

```
Meter: 0/2
Style: Singer
Sharps: 1
Title: Psalm 42\\{\textit{\normalsize Ainsi que la biche r\'ee}}

@0 g2 a4 b2 a4 g f e2 d2 \caesura\ g2 a4 b2 c4 b2 a g :|
L: Soos 'n hert in dor-re stre-ke, skreeu-end dors na die ge-not
L: van die hel-der wa-ter-be-ke, skreeu my siel na U, o God.

@+1 b2 b4 d2 c4 b a b2 \caesura\ d2 d4 e2 d4 c b a2
L: Ja, my siel dors na die Heer, na die Le-wens-bron.  Wan-neer

b2 d4 c2 b4 g a b2 g2 \caesura\ b2 b4 c2 b4 a g f2 g2
L: sal ek sien in klaar aan-skou-e?  Op die Heer is my ver-trou-e.
```

- `Meter: 0/2` indicates counting in half-notes, but no bar lines.
- Line breaks in the `Title` are made by `\\`.

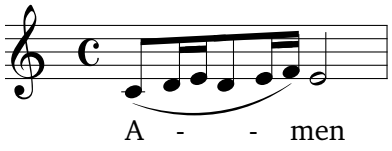
D.4 Beams, slurs and melismas



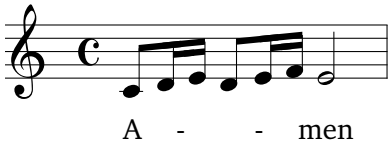

```
Style: Singer
Meter: C
Start: @+2

%% w2i
( c8 d1 e1 d8 e1 f1 ) e2 |
L: A-men
```



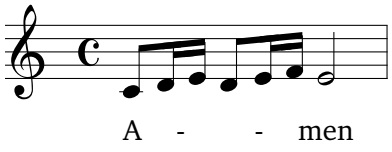

```
[ c8 d1 e1 d8 e1 f1 ] e2 |
L: A-men
```



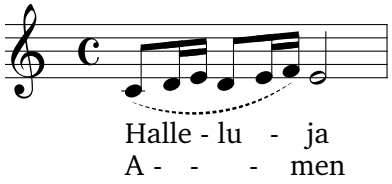

```
( [ c8 d1 e1 d8 e1 f1 ] ) e2 |
L: A-men
```




```
%% \\let\BM\beginmel\let\EM\endmel\
[[ \BM\ c8 d1 e1 ] [[ d8 e1 \EM\ f1 ] e2 |
L: A-men
```




```
[ c8 d1 e1 ] [ \nolyr\ d8 e1 f1 ] e2 |
L: A-men
```

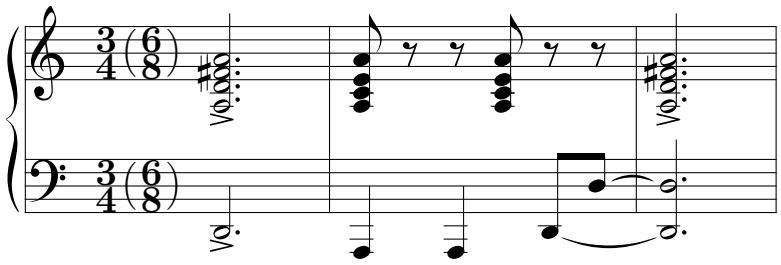



```
@-3 (" [ c8  \lyr\ d1 e1 ] [ d8 e1 f1 ] ) e2 |
L: Hal-le-lu-ja
L: A---men
```

Some points to note: Example 1 uses a slur with default PMX beaming, Example 2 uses forced beaming to beam all notes, Example 3 does both. Examples 4 and 5 violate the ***M-Tx*** convention that notes sung to a single syllable must be slurred or beamed together, so we override this in either of two ways using `musixlyr` commands. Example 4 uses `\beginmel` and `\endmel` to force melismas in a situation where `[[` has suppressed them; this method is good for long melismas over several beamed components. Example 5 suppresses lyrics where the beaming would have put them. Example 6 uses `\lyr` to force lyrics where the beaming would have suppressed them, and shows the use of empty syllables to synchronize lyrics and of broken slurs to indicate different slurring in the two verses. — *Suggested by Stefan Haller*

D.5 Dirty Tricks

```
Style: Piano
Meter: m3406
%% \def\upiii#1{\raise3\internote\hbox{{#1}}}
%% \def\chacarera{\meterfrac34\upiii{\Big(}\meterfrac68\upiii{\Big)}}
%% \let\oldmeterC=\meterC  \let\meterC=\chacarera

%% w100m
a-2d o> a8 r r a r r a2d o> |
C: dfsa cea cea dfsa
d-2d o>-15 a4 a { [h d8 {u d+ ] d2d } zd- } |
```



These are bars 7–9 of the *Credo* from *Misa Criolla* by Ariel Ramírez. It is in the *chacarera trunca* meter, which mixes $\frac{3}{4}$ and $\frac{6}{8}$ time. There is no **PMX** construction for such exotic meters, so we need to be ingenious.

We specify the meter as `m3406` which tells **PMX** to use $\frac{3}{4}$ meter but to print the C symbol for common time. Of course we do not actually want the C symbol, we want something that looks like $\frac{3}{4}(\frac{6}{8})$. Now all that **PMX** knows about the C symbol is that it is called `\meterC` in MusiXTEX. So we define `\chacarera` and assign it to `\meterC` in preamble `%%` commands. These appear in the TEX file after the command to read in `musixtex.tex` but before the commands that actually print something. If later we need the original `\meterC` it can be restored by `\let\meterC=\oldmeterC`.

The excerpt illustrates some sophisticated **PMX** features too. Since the `>` ornament on the bottom stave goes above the stave by default, we need to move it down explicitly. The beamed eighths look better with a horizontal beam, which is enforced by `[h`. Finally, there is a tie involving a melodic note with a note in a chord; to allow this, the chordal note must come on a music line, not on a `C:` line.

D.6 Extra text after the piece

```
Meter: 3/4
Title: Song of May
Flats: 1
Style: Singer
Systems: 2
Space: 10

%% \\let\endpiecesav\endpiece\
%% \\def\endpiece{\endpiecesav\input lyrics \input notes}\
( d4 e ) g | ( d e ) g | ( f e ) f | d2 r4 |
L: In ver-na-lis tem-po-ris
L: Dum re-ces-sum fri-go-ris

( f g ) a | ( a g ) f | g2d | g2 r4 :|: g2 d4+ | d2 d4 | ( d c ) b | a2 r4 |
L: or-tu lae-ta-bun-do, ter-rae, ma-ris, ne-mo-ris
L: nun-ti-at hi-run-do, vi-gor re-dit cor-po-ris,

g2 a4 | ( b a ) g | ( f e ) f | d2 r4 | ( f g ) a | ( a g ) f | g2d | g2 r4 :|
L: de-cus ad-est de-fo-ris re-no-va-to mun-do,
L: ce-dit do-lor pec-to-ris tem-po-re iu-cun-do.
```

When typesetting a vocal composition you may sometimes want to put only the text of the initial stanza into your score and let the remaining part of the lyrics be printed below the score. You might also want to add some notes or comments.

Apparently this is not possible according to the set of commands available for ***M-Tx*** and **PMX**. There is, however, a workaround to do things like that. The trick is to look for a command `\Endpiece` or — as in this example — `\setrightrepeat\endpiece` near the bottom of the MusiXTEX file generated by the **PMX** preprocessing. You may extend the actions of `\Endpiece` (or `\endpiece`) by saving the command with another name (`\endpiecesav`) and redefine `\Endpiece` to first execute `\endpiecesav` and then request the inclusion of one or more files containg plain TEX code. The text of the included file(s) will be typeset on unutilized space on the last page or else on a new page.

MusiXTEX commands like these may — as noted in Appendix B.4 — be put directly into the mtx source file in %%-lines at the head of the first music paragraph.

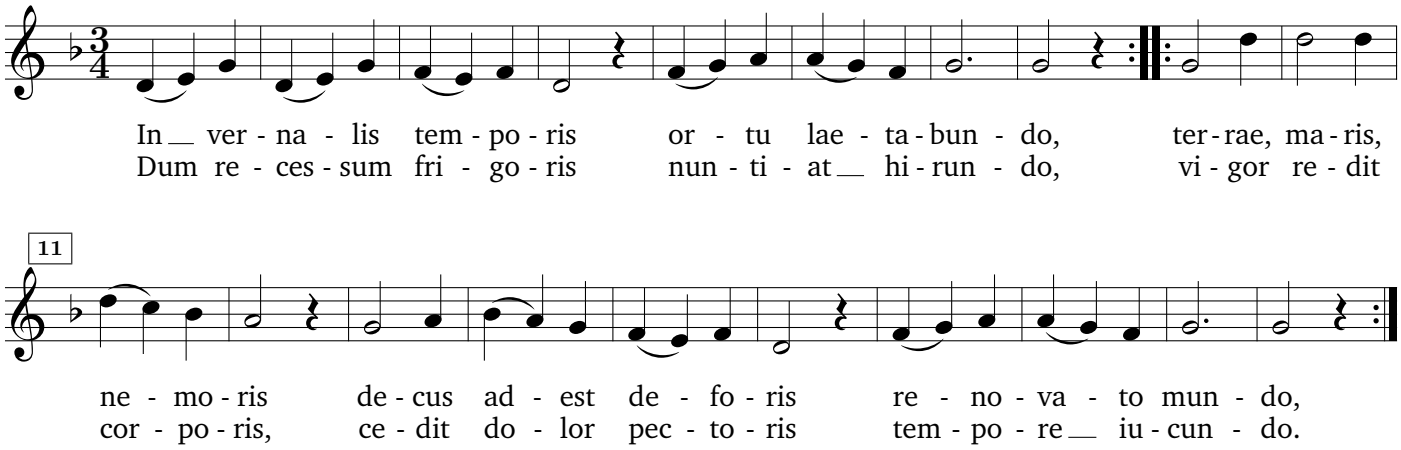
In the current example two files have been included. `lyrics.tex` contains the original Latin lyrics and an English translation. A TEX table[9] consisting of a template line followed by item lines for each lyrics line has been used to set up the text in two columns - one for the lyrics and one in italics for the translation. Notes on the song is included from `notes.tex` and appended below the lyrics. *— Contributed by Christian Mondrup, following a hint by Werner Icking*

```
% ------- file notes.tex ---------------------------------------
\vskip 10 mm
\noindent Lyrics by Morten B\o{}rup, 1446-1526, headmaster at the
grammar school of Aarhus.  The song was published \par
with an anonymous melody in \it Piae Cantiones
\rm by Theodoricus Petri, Greifswald 1582.
```

[9] See Michael Doob: A Gentle Introduction to TEX, section 6.2. The book may be found as `gentle.tex` in the Documentation subdirectory of the TEX CTAN mirror archives.

```
% -------- file lyrics.tex ---------------------------------------
\vskip 8 mm
\halign{\hskip 35 mm # \hfill & \hskip 25 mm \it # \hfill \cr
In vernalis temporis  & In the time when \cr
ortu laetabundo,  & spring rises joyfully, \cr
dum recessum frigoris & the end of frost \cr
nuntiat hirundo,  & is heralded by the swallow, \cr
terrae, maris, nemoris  & earth, sea and grove \cr
decus adest deforis  & is full of beauty \cr
renovato mundo,  & like a renewed world, \cr
vigor redit corporis,  & the body regains strength, \cr
cedit dolor pectoris  & the sorrow of the heart ends \cr
tempore iucundo. & in this playful time \cr
}
```

Song of May



In vernalis temporis	*In the time when*
ortu laetabundo,	*spring rises joyfully,*
dum recessum frigoris	*the end of frost*
nuntiat hirundo,	*is heralded by the swallow,*
terrae, maris, nemoris	*earth, sea and grove*
decus adest deforis	*is full of beauty*
renovato mundo,	*like a renewed world,*
vigor redit corporis,	*the body regains strength,*
cedit dolor pectoris	*the sorrow of the heart ends*
tempore iucundo.	*in this playful time*

Lyrics by Morten Børup, 1446-1526, headmaster at the grammar school of Aarhus. The song was published with an anonymous melody in *Piae Cantiones* by Theodoricus Petri, Greifswald 1582.

E *M-Tx* and LaTeX

This section assumes that you have some knowledge of LaTeX.

The normal output from **PMX** is a stand-alone plain TeX file, which cannot without further ado succesfully be input into LaTeX. The file `mtxlatex.tex` allows you to prepare music scores using LaTeX rather than plain TeX, which is particularly useful when you need the enhanced font handling features of LaTeX, and to produce books combining the outputs of several **PMX** runs into a single document, such as a collection of songs, or a mainly text document (like this one) with many small music excerpts.

Two points apply to all cases:

1. Order of inclusion of packages is important. `mtxlatex` must be the first package included, before any font packages.
2. It is easy to incur `TeX capacity exceeded`. For example, the index to this manual is a separate document because package `multicol` won't load on top of all the others. If you wish to do large scores, it is probably advisable to change `texmf.cnf` so that `save_size = 10000` and run `initex`.

E.1 Collections of complete pieces

Suppose you need to make a document containing several complete pieces, with perhaps a title page, a foreword and an afterword, like a typical published book of piano pieces or songs.

1. Prepare a `.tex` file for each separate piece, using the methods described previously, and be sure that it is fully debugged and looks the way you want it, using the standard `tex-musixflx-tex` compiling cycle.
2. Write a LaTeX file looking something like this:
3. Use a `latex-musixflx-latex` compiling cycle to make the finished document.
4. If you make small changes in the LaTeX part only, a single LaTeX pass is sufficient, unless the page numbering has changed.

You can see the finished product in `halleluja.pdf`. Some points to note:

- The package `mtxlatex` comes with ***M-Tx***. You must put the command `\mtxlatex` in the preamble.
- The package `times` comes with standard distributions. You could use any LaTeX 2$_\varepsilon$ font package here. The `times` font is quite narrow and therefore useful when the lyrics are crowded. If you do use a font a package, load it *after* `mtxlatex`.
- `\pagestyle{headings}` puts page numbers and running headings at the top of the page.
- To change fonts, use pure LaTeX 2$_\varepsilon$ font change commands, as above. Don't try to use `\it`, `\bf` etc. Briefly, subject to what fonts you have installed, you can independently change:

  size `\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge`, `\Huge`;

  shape `\upshape`, `\itshape`, `\slshape`, `\scshape`;

  family `\rmfamily`, `\sffamily`, `\ttfamily`;

  series `\mdseries`, `\bfseries`.

For a full description, see any good LaTeX 2$_\varepsilon$ manual.

- The blank line after `\pagebreak` is essential.
- The environment `Score` is used to include each piece. It takes two arguments: the first will appear in the list of contents and as a running page heading; the second is the name of the `.tex` file. As usual, the extension `.tex` may be omitted. There is also an environment `score` which is less convenient but more flexible: it has no arguments, reads in nothing, puts nothing in the table of contents, and does not change the running page heading. Use it when you need a special lyrics font, e.g.

```
\begin{score} \headingandcontents{Loof nou die Heer}
  \sffamily \input loofnou
\end{score}
```

E.2 Collections of morsels

The situation is a little different when the pieces are so short that more than one of them fit on one page. For these, use the environment `excerpts`, as in the following example:

```
\documentclass{article}
\usepackage{mtxlatex,charter}
\mtxlatex
\pagestyle{empty}
\renewcommand{\writebarno}{\rule{0pt}{12pt}}
\begin{document}
 \begin{excerpts}
\input{dona.tex} \vskip 8mm
\input{sanctus.tex} \vskip 8mm
\input{viva.tex} \newpage
  \end{excerpts}
\end{document}
```

The rest of the procedure is the same as for collections of longer pieces. You can see the finished product in `halleluja.pdf`. Some points to note:

- The `excerpts` environment redefines the heading fonts to more modest sizes.
- A page break should be made before quitting the environment.
- The pieces do not have bar numbers, but space should be reserved for the numbers that show where the voices enter. This is done by redefining the `\writebarno` command.

E.3 Documents with small music excerpts

This manual relies heavily on the `mus` environment. For example, the musical emblem on the front page was set using the commands

```
\begin{center}
  \framebox[5in]{\parbox{3.6in}{
    \begin{mus} \input title.tex \bigskip \end{mus}
  }}
\end{center}
```

Some points to note:

- The excerpts vary in length and some experimentation is required to get the dimensions and positioning just right.
- Putting the `mus` environment inside a `center` environment only works when the music line is not longer than the text. Otherwise, it is useful to use `\hskip`, as in the case of the first example in the manual:

  ```
  \hskip -18mm
  \begin{mus}
  \input mozart0.tex
  \end{mus}
  ```

  For this technique to work properly, it is essential that the above four lines form a paragraph of their own, i.e. they are preceded and followed by blank lines.

Appendix D.4 was produced using the command `\example`. This is a rather complicated macro with eight parameters, which is documented at the end of the file `mtxlatex.sty`.

F Overriding predefined TEX commands

This section assumes that you are an experienced TEX user.

The logical structure of TEX is such that most of the time, a command used in a definition need not exist at the time that the definition is made, as long as it is defined by the time that the definition is used. This behaviour is known to computer programmers as *run-time binding*.

We have already met examples of how to exploit this feature in Sections D.5 and D.6, where respectively `\meterC` and `\Endpiece` were redefined. The macros of the previous section also work in that way, by locally (i.e. inside the various environments) redefining commands that appear in the TEX file generated by **PMX**.

The technique is an extremely powerful and flexible one, and in the hands of an expert can achieve practically any desired result. Here we can only scratch the surface. One point to note is that good programming practice requires that you save the current meaning of the command and restore it later, e.g.

```
\let\commandsave\command\def\command{...}
...
\let\command\commandsave
```

In LATEX, if you redefine a command inside an environment, the saving and restoring is automatic.

F.1 Changing fonts

The font used for lyrics is the default text font of the moment, which usually is ten-point roman. It can be changed globally, e.g. `%% \twelverm` in the preamble will change the font to twelve-point. When using LATEX, you can use font-changing commands inside the environment to change the lyrics font only for that particular piece.

The fonts used for titles and other items in headers are respectively called `\BIGfont` and `\Bigfont`. In `mtxlatex.sty` they are redefined as follows:

```
\renewcommand{\BIGfont}{\Huge\bfseries}
\renewcommand{\Bigfont}{\Large}
```

If you are not using LaTeX, they can be redefined in two ways:

- You can simply assign an existing font, e.g. to get somewhat smaller fonts,
  `\let\Bigfont\bigfont\let\BIGfont\BIgfont`
- You can define the font explicitly, e.g.
  `\font\BIGfont=cmss9 scaled \magstep4`

F.2 mtx.tex

The ***M-Tx*** preprocessor, as far as possible, issues commands starting with `\mtx` instead of direct TeX commands. The main purpose for so doing is that the user can modify their effect by redefining them in the source file. These macros, together with a few others that make life easier, are collected in the file `mtx.tex`.

Documentation of these commands can be found at the end of the file `mtx.tex`.