PMX – a Preprocessor for MusiXTEX

Version 2.2 – March 2000

Don SIMONS
Dr. Don's PC and Harpsichord Emporium
Redondo Beach, California, USA.
dsimons@logicon.com

Contents

1 Introduction **2**
Conventions for This Manual . . . 2
Setup . . . 3
Basic Operation, by Example . . . 3

2 Elements of PMX **4**
2.1 Setup Data in the Input File . . . 4
2.2 Structure of the Body of the Input File . . . 5
Notes . . . 6
Rests . . . 8
Chords . . . 8
Grace notes . . . 9
Ornaments . . . 9
Editorial accidentals . . . 9
Slurs . . . 9
Ties . . . 10
Beams . . . 10
Clefs . . . 11
Arpeggios . . . 11
2.3 Commands That Affect All Voices . . . 12
Repeats, double bars, forced single bars . . . 12
Voltas (first and second endings) . . . 12
Meter changes . . . 12
Transposition and key changes . . . 12
Text . . . 13
Page numbering, centered header text . . . 13
Overriding defaults for accidentals, dot positions, spaces at start of bar and between staves . . . 13
Extra hardspace, horizontal shifts . . . 14
Minimum spacing between notes in crowded systems . . . 14
Page size . . . 14
Line, page, and movement breaks . . . 14

Fractional bars 15
Stem direction of bass notes 15
2.4 Putting TEX Commands into the .pmx File 15
2.5 Figured Bass 16
2.6 Macros 16
2.7 Batch Processing 16
2.8 Lyrics 17

3 Making Parts from a Score **17**

4 Making MIDI Files **18**

5 Limits **19**
Limits under direct user control 19
Limits not under immediate user control 20

6 Closing Notes **20**
About the Example Files 20
A Benign Bug 21
Where to Get **PMX** 21
Acknowledgments 21

1 Introduction

PMX can help you typeset a musical score if you have a reasonably up-to-date version of MusiXTEX. The goal of **PMX** is to facilitate the efficient production of scores and parts that have an almost professional appearance. It can do *all* the work involved in setting up `\notes-\enotes` groupings, selecting groups of notes to be beamed, defining beam heights and slopes, spreading the entire piece evenly over specified numbers of systems and pages, and inserting extra spaces where needed to make room for accidentals, flags, dots, and new clefs. The input language for **PMX** is much simpler than MusiXTEX. You can enter note values and rests from 64ths to double whole notes (*breves*), ornaments, slurs, and limited text strings. Every voice in every bar must have exactly the correct number of beats in the current meter, but you may change the meter at the beginning of any measure, with or without printing the new time signature. Before making a TEX file, **PMX** checks these timings and other aspects of the input. **PMX** has special features for dealing with baroque chamber music, including the ability to notate figured bass below the bottom staff in each system. If **PMX** hasn't yet learned to do something you want to do, you can usually work around the problem by inserting literal TEX strings in the **PMX** input file.

You can automatically create parts from a score using **scor2prt**. This auxiliary program generates a set of `.pmx` input files, one for each part, from a single `.pmx` file for the score. You can control the appearance of the parts with special commands in the main file, thereby making it possible to include within a single input file all the information that defines the score and the individual parts.

The native language of **PMX** is FORTRAN and its home port is DOS. The basic distribution contains the FORTRAN sources, and binaries that will run in a DOS window on a PC with 2.0
WINDOWS95 or higher. Thanks to some very helpful individuals, ports are available for Macintosh and UNIX (see section 6 on page 21).

Conventions for This Manual

Hey, this is boring stuff, but if you take a minute to understand the typographic conventions and a little jargon, it may avoid some confusion down the road.

The typewriter typeface always indicates verbatim text as it would be input to a computer. This includes file names, MusiXTeX tokens, and **PMX** commands, e.g., `barsant.pmx, \internote, c44`.

Bold is used for program names (e.g., **pmxab**), or when applied to a single letter, to relate a **PMX** command to its meaning (e.g., "`e` signifies a l**e**ft shift").

Italics may mean several different things depending on the context: simple emphasis, or the first appearance of *jargon* (buzz-words that need to be explicitly defined), or finally to represent input variables for which some verbatim text would need to be substituted. In the latter case the variable will be surrounded by square brackets, e.g., [*basename*], but the brackets are not to be included with the verbatim text.

Speaking of jargon, there are several special words that have very specific meanings here: A *staff* is one set of 5 lines (plural *staves*), a *system* is a group of staves, *voice* is the same as *staff*, and *line of music* refers to one of the one or two simultaneous allowable sequences of notes in a voice. Note that this definition of voice is a little unconventional, since a single voice may have more than one line of music.

Setup

This section describes the setup for the DOS version or for those compiling the FORTRAN source. Other versions should be accompanied by their own setup instructions.

After decompressing the distribution file if necessary, you should have these files: `pmxab.for`, 2.0
`scor2prt.for`, two corresponding DOS executables, several sample `.pmx` files, `pmx.tex`, `ref220.tex` (TeX source for a command summary), `pmx220.tex` (TeX source for this file), and Postscript images of the latter two files. If necessary, compile the FORTRAN programs. I have tried to keep the source code as generic as possible, but minor modifications may be needed for FORTRAN-to-C translation and/or other compilers.

Once you have assembled a full set of files, put the executables somewhere in the path or in your working directory, `pmx.tex` into the texinput directory, and the sample `.pmx` files in your working directory (the one from which you will run **PMX**).

Basic Operation, by Example

Edit the 15th line of `barsant.pmx` to contain the path to the texinput directory; **PMX** will write the `.tex` file there. For example, if you want this to be the same as the working directory, type `.\` for DOS, or `./` for UNIX.

Execute **PMX** by typing `pmxab barsant` . Or, for backward compatibility, you may type `pmxab` 2.0
`<return>` and you will be prompted for a jobname. **pmxab** will always generate two files in the working directory: `barsant.pml` is a log file, and `pmxaerr.dat` contains a single integer, 0 if the run was successful, otherwise the line number in the `.pmx` file of the fatal error (useful for batch processing). Also, on successful completion, `barsant.tex` will be placed in the path specified in the setup.

Now you are right where you would be after entering, debugging, and rough-editing the `.tex` file manually. To see the results, process `barsant.tex` just as you would for any MusiXTeX file, running all three passes, and view the `.dvi` file. To make separate parts, run **scor2prt** by typing
`scor2prt barsant` . The program will create a new `.pmx` file for each instrument, in this case 2.0
`barsant1.pmx` and `barsant2.pmx`. You may then process these files like you did the original one to create separate parts.

2 Elements of PMX

2.1 Setup Data in the Input File

To see how the input file is put together, we'll look at `barsant.pmx`. For reference, here are are the first few lines:

```
%----------------%
%
%  barsant.pmx   Revised 31 August 1997
%
%----------------%
%
% nv,noinst,mtrnuml,mtrdenl,mtrnmp,mtrdnp,xmtrnum0,isig,
  2    2      4       4        0      6      0       0
%
% npages,nsyst,musicsize,fracindent
    1      7     20         0.07
Basso
Recorder
bt
./
```

The lines with `%` in column 1 are comments. Some special handling of comment lines will be discussed in the section on creating parts from a score. The rest of the lines in this example are the *setup data.*

Starting in the first non-comment line above,

`nv` (integer$\leq$12) is the total number of **v**oices, or sta**v**es per system. [We will use the terms *voice* and *staff* (plural *staves*) interchangeably]. Each voice or staff may contain either one or two lines of music. 2.0

`noinst` (integer$\leq$`nv`) is number of *instruments*. Each instrument has a unique name (see below), and any instrument with more than one voice will have its staves joined with a curly bracket. Usually there is only one voice per instrument and `noinst=nv`. There are two ways to assign more than one voice to one or more instruments. If only the first (lowest) instrument has more than one voice, such as in a score for piano and a solo instrument, simply make `noinst<nv` and any difference will show up in instrument 1, the bottom one in each system.. For a more general distribution, put a minus sign in front of `noinst`, and follow `noinst` with the number of voices in each instrument in succession, separated by spaces. These numbers must add up to `nv` or your computer will explode. For a typical example of keyboard music, see `mwalmnd.pmx`, in which `nv=2` and `noinst=1`, producing two staves per system with a curly bracket at the left.

`mtrnuml` is the *logical* numerator of the meter, or the number of beats per measure; `mtrdenl` the denominator. Please note the special considerations in the paragraph after the next. If `mtrnuml` is divisible by 2 or 3, beam grouping will be automatic; otherwise you will have to force all beams using `[...]` as described in subsection 2.2 on page 10.

`mtrnmp` and `mtrdnp` are the *printed* numerator and denominator. These determine the appearance of the meter in the printed output but have no effect on the internal timing analysis. If `mtrnmp>0` then it and `mtrdnp` are printed literally as the numerator and denominator of the time signature. Please note the special considerations in the following paragraph. If `mtrnmp<0`, then the numerator is abs(`mtrnmp`) and the entire time signature will be printed with a vertical slash through it. If `mtrnmp`=0, then `mtrdnp` determines the printed meter as follows:

`0`	No meter is printed (*blind* meter change)
`1, 2, 3, or 4`	A single digit, between the 2nd and 4th lines
`5`	Cut time (alla breve)
`6`	Common time
`7`	Numeral 3 with a vertical slash

There are special considerations for n/16 and n/1 time signatures (where the latter "1" normally means a whole note). To get n/1 time, use `0` (zero) for `mtrdenl` and `1` for `mtrdnp`. To remember this rule, recall that the printed denominator is taken literally, while the logical denominator can always be represented by the same single digit used for the corresponding time value when entering ordinary notes (see section 2.2 starting on page 6). So for n/16 time, use `1` for `mtrdenl` and `16` for `mtrdnp`.

If the first bar is a partial bar containing a pickup, `xmtrnum0` is the number of beats in it; otherwise set it to 0. It need not be an integer. The first bar is the *only* bar that can have a different number of beats than the current value of `mtrnuml` (Later we'll see how to change the meter).

`isig` is the key signature, positive integer for sharps, negative for flats.

If `npages`>0, it is the number of pages and `nsyst` is the total number of systems in the entire piece. **PMX** will spread the entire piece horizontally over this number of systems, and vertically over `npages` pages. For proper vertical spacing there should be from about 9 to 16 staves per page. If you specify too many staves for the number pages, one or more staves will spill over onto an extra sheet. If this happens it will only become obvious when you preview the `.dvi` file. However, with practice, you can spot the problem by watching the page numbers printed to the screen on the first pass through TEX. The most common solution is either to increase `npages` or decrease `nsyst`.

If `npages` is set to 0, then `nsyst` is interpreted as the average number of measures per system. This is useful while building up a file a little at a time. **PMX** will decide how many pages to use.

`musicsize` is either 20 or 16, the height of a staff in points.

`fracindent` is the indentation of the first system from the left margin, expressed as a decimal fraction of the total line width.

Next come the names of the `noinst` instruments as you want them to appear within the indentation in the first system, one per line, starting with the *bottom* instrument. If you've set `fracindent`=0 and don't want instrument names to appear, you must still leave `noinst` blank lines here. Next comes a single string of `nv` letters or numbers for the clefs, again starting with the bottom voice: `b, r, n, a, m, s, t, f` or digits 0-7 respectively for **b**ass, ba**r**itone, te**n**or, **a**lto, **m**ezzo-soprano, **s**oprano, **t**reble, or **F**rench violin clef. The last line of setup data contains the path 2.2
to the texinput directory, where you want the `tex` file to go when **PMX** creates it. The one here, `./` , represents the current directory in UNIX and some versions of DOS. The path must terminate with `/` or `\` .

2.2 Structure of the Body of the Input File

The rest of the `.pmx` file is the *body* of the input. The basic unit of input from here on is called an *input block* or just *block*, each one representing an integral number of bars except possibly the first, which may include a pickup. There will usually be 4 to 8 bars in a block. 15 is the most allowed. It is good practice to separate the blocks with comment lines showing which bars they represent, as I've done in `barsant.pmx`. It is also advisable, although not required, to separate the bars with the symbol `|`, whose main functions are to provide visual separation in the input file, and to help isolate input errors: if you put one anywhere except at a bar-end, **pmxab** will stop and show you where it detected the timing error. Otherwise, with several minor exceptions, `|` has no effect.

At the start of each block there may be a few special symbols (described in section 2.3 starting on page 12). Next come the input data for the selected number of bars of the first (lowest in the system) line in the first staff, followed by either `/` to move to the next staff, or `//` to move to the next line of music on the same staff. Each new line of music should start on a new line in the input file, i.e., there should be no further data on the same input line after `/` or `//` . Continue entering other lines of music, each with *exactly* the same number of bars as the first, terminated by `/` or `//`, until the last (topmost in the system) ends with a `/` and the block is finished. Within a block every line must have the same number of bars, but every block needn't have the same number of bars as other blocks. The number of lines in a voice can only be 1 or 2, and cannot change within a block, but may vary from block to block.

The data for each line of music in each voice are a sequence of *symbols* containing one or more adjacent characters. Symbols are separated from each other by spaces. The line-terminating symbols `/` and `//` should also naturally be preceded by a space.

Notes

Symbols for notes always start with a lower-case letter and end at the first space. The first letter is the note name (`a-g`). The rest of the characters can be in any order with only a few restrictions. The first digit defines the *basic time value* of the note: `9, 0, 2, 4, 8, 1, 3` or `6` respectively for double-whole, whole, half, quarter, eighth, sixteenth, thirty-second, and sixty-fourth notes. The second digit sets the octave (for reference, octave 4 runs from middle C to the B above). Certain letters may appear after the initial one: `d` for **d**ot; `dd` for double dot; `f, n,` or `s` for **f**lat, **n**atural, or **s**harp (repeat the letter immediately for a double); `u` or `l`, which force the stem direction of any un-beamed note; `e` or `r` to shift the notehead l**e**ft or **r**ight by its own width; and `a` (for **a**lone) which inhibits beaming for this note. Other characters allowed in note symbols are `+`, `-`, `.`(period), `,`(comma), `x`, and several special characters following `x`, all to be described below. Between the first letter and the end or `x` if present, non-digits can be in any order with respect to each other and to the digits, with minor exceptions involving shifting dots and accidentals.

To move a dot from its default location, simply follow the `d` with one or two decimal numbers, each predeced by `+` or `-`. The first is the vertical shift in `\internote`s, the second, the horizontal shift in notehead widths.

Accidentals can be shifted too. One way is to enter `+` or `-` right after the accidental character, then an integer for the vertical shift, then another `+` or `-` followed by the horizontal shift in notehead widths. If you use this method, you *must* enter both numbers. Or, to just shift horizontally, use `<` or `>` followed by the shift in notehead widths. When shifting a sharp to avoid another sharp, a left shift of 0.85 is usually best. When shifting a flat to avoid a flat above it, a left shift of 0.3 is suggested.

Dots and accidentals always have to be entered when and if a note calls for them. i.e., they are never carried over from previous notes. On the other hand, the octave only needs to be entered if the note is more than a fourth away from the most recent note in the same voice. This feature lets you go for long stretches in a voice before needing to enter the octave. An alternate way to jump more than a fourth but less than a twelfth is to type `+` or `-`. In other words, these symbols mean to put the note an octave higher or lower than it otherwise would have gone. Two `+`'s will raise the pitch two octaves above what it otherwise would have been, and so forth. The basic time value is also carried over from the past if it is not re-entered, except for the first note or rest in each line of music in an input block, for which it `must` be entered. Therefore, when the melody jumps more than a 4th, using `+` or `-` is often more convenient than using a digit. This is because in order to use the digit, you must first enter the basic time value whether it changes or not.

For example `c44 d e f g a b c c0-` is an ascending quarter-note scale starting on middle C, followed by an octave jump down to a whole note middle C.

Starting with version 1.2, octave numbers can be combined with one or more `+` or `-` . In earlier versions, `+` or `-` was ignored if an octave number was specified. This is a slight backward incompatibility; **pmxab** prints a warning when it happens.

The first note symbol of each line of music in a block must contain at a minimum the note name or `r` for a rest (see below), and a basic time value. An option in setting the pitch level of the first note in a line in any block except the first, is to rely on some fairly obvious inheritance rules from the end of the prior block. However, if the number of lines of music in a staff has changed from the prior block, it is safest to reset the octave at the start of a new block. Duration is never inherited and must be set at the start of each input block.

Dots can be a little tricky, because even though they affect the actual time value, they don't affect the basic time value, and it is only the latter that is "sticky". Therefore, if a note is dotted, you always have to enter a `d` (or a period, see next paragraph) somewhere within the symbol, after the note name, even if the actual time value and octave are the same as the prior note. But the *basic* time value need not be re-entered if it hasn't changed (unless the note is more than a fourth from the prior note *and* you have for some strange reason elected to indicate the octave with a number rather than `+` or `-`). So for example, consecutive dotted half notes, each within a fourth of the previous one, could be most cleanly entered as `cd24 ed gd ed`, whereas `cd24 e` would represent a dotted half note followed by a plain half note (since the basic time value–as defined by the first digit–was a half note all along).

There are two special shortcut rhythmic notations. For normal dotted rhythms (3:1 ratio), if you include a period (`.`) in the note symbol, it will (a) assign a dot to the note just entered, (b) terminate that note, (c) prepare to receive the next note name *without any space*, and (d) automatically assign a time value to the second note equal to one-third of the first one. No time value may be entered for the second note, but octave and accidental data may. Ornaments and slurs (see below) following this symbol will apply to the second member; you can't use it if you need such adornments on the first. However, the main advantage of this shortcut comes if you want to follow one such pair with another of the same rhythm; in that case you needn't enter any further time value for *either* member of the second pair. This is only possible because after using the shortcut, the basic (inheritable) time value is set to that of the *first* note in the pair, without the dot.

For paired notes with 2:1 rhythmic ratios, the symbol `,` (comma) behaves similarly to the `.` (period) for 3:1 rhythms.

Xtuplets can have from 2 to 24 notes or rests all of equal duration. The symbol for the first note of an xtuplet begins exactly like a note or rest symbol, with the name of the first note in the xtuplet, or `r` if it starts with a **r**est (see next subsection), and an optional time value. However, the actual time value (including a dot if present and a basic duration that may have been inherited from the prior note) now represents the *total* duration of the xtuplet. Next (with no space, as usual) comes `x` followed by a one- or two-digit integer for the number of notes in the xtuplet, then optionally an `n` to control the printing of the **n**umber. A blank following `n` signals that the number should not be printed. On the other hand, if one or two decimal numbers follow `n`, each preceded by `+` or `-`, the first is a vertical shift in `\internote`s, and the second, a horizontal shift in notehead widths. A final option after `n` is `f`, to **fl**ip the number vertically from its default position.

The second through last notes of the xtuplet are each then represented by a separate symbol containing a subset of the characters permitted for ordinary notes or rests: note name or `r` (the only required character), accidental, and octave change character (`+` or `-`). The octave may be given explicitly instead, and any integer will be interpreted as such, as no time values or dots are permitted.

The last note of an xtuplet may not be a rest.

As an example, an ascending quarter-note triplet scale would be notated

```
c44x3 d e f4x3 g a b4x3 c d ...
```

Rests

The symbol for a rest starts with `r`. Then for a normal rest, in either order come a digit for the basic time value (using same codes as for notes, optional if unchanged from previous value), a `d` if the rest is dotted, and a second `d` if double dotted. The basic time value of a rest affects future notes and rests the same as if it had come from a note, i.e., it applies until another value is entered with a subsequent note or rest in the same line of music. The symbol `rp` represents a full-bar rest notated with a *pause* character (whole rest) regardless of the time signature; in this case no other duration information is needed or allowed. `rb`, followed if necessary by a duration specifier, denotes a ***blank*** rest, one that occupies space and time but is invisible. This is most often used when there are two lines of music in a staff and one drops out for some of the duration of the current input block. (See `mwalmnd.pmx` for examples). The option `o` (for **o**ff-center) suppresses centering a full bar rest. If you don't exercise this option, then *all* full-bar rests will be horizontally centered between bar lines, including pauses (`rp`) as well as normal rests that fill the bar. `rm` followed immediately by an integer will generate a ***multi-bar*** rest, a special combination of symbols between two bar lines with an integer above representing two or more bars of rest. This symbol will generally only be used in separate parts after having been automatically generated by **scor2prt**. 2.0

The default vertical position of a rest depends on whether there are one or two lines of music in the staff. For one line it is just the MusiXTEX default (approximately centered on the middle line). On the other hand, in the lower line of music in a two-line voice, the rest is lowered `4\internote`, while in the upper line it is raised `2\internote`. The **PMX** default can be manually overridden by appending `+` or `-` and an integer representing the offset from the *middle* line of the staff (not from the **PMX** default if there are two lines in the voice!). So for example, in a single staff in 3/4 meter, two lines of music each with a half note followed by its own quarter rest would be either

```
c24 r4 //
c25 r4 /
```

or equivalently

```
c24 r4-4 //
c25 r4+2 /
```

while

```
c24 r4+0 //
c25 r4b /
```

would produce two notes followed by a single, vertically centered rest.

Chords

Chordal notes, which always share a stem and the same time value as the prior note, are symbolized with `z` followed by a note name and optionally an accidental, `+` or `-` as octave indicator, and `e` or `r` for a l**e**ft or **r**ight shift by one notehead width. No basic time value is allowed. If the main note is dotted, then the chordal note will appear with a dot regardless of whether a `d` is entered. The only time a `d` is required in a chordal note symbol is if the dot's position is to be adjusted; in this case the `d` is required, followed by one or two decimal numbers, each preceded by `+` or `-`. The first is the vertical shift in `\internote`s; the second, the horizontal shift in notehead widths. Any number of chordal notes can follow a single main note. The stem direction of a chord is controlled by the main note, but may be manually overridden with `u` or `l` in the main note symbol.

When chordal notes are beamed together, the default height and angle of the beam will be determined by the main note on each stem (the one without `z`). If a beam joining chordal notes looks bad, you can usually fix it either by changing which note acts as the main one, or by fine-tuning the beam parameters as described starting on page 10.

Grace notes

A grace note symbol starts with a `G`. It is entered in its natural order, normally before the main note, but sometimes after. After `G` and before the note name, comes any combination of the following options: a single digit representing the number of notes in the grace (default is 1), `m` and a digit for ***m**ultiplicity* (number of flags or beams, default is 1, 0 is allowed), `s` for **s**lur (joining all notes of the grace to the main note; no other `s` is needed on the main note), `x` for a slash (only for single graces), `l` or `u` to force the direction of the stem(s), `A` (for **A**fter) or `W` (for **W**ay-after) to associate the grace note with the *prior* note. Next comes the only required character, the first note name. No time value can be entered, but if needed, the octave or an accidental can be given as in a normal note. Second and later notes must follow immediately in sequence, set apart by spaces, likewise without any time value, and without any intervening symbols.

Normal or after-graces will be placed *immediately* before or after the main note; way-after's, as far to right as possible before the next note or bar line. If either type of after-grace is slurred, the slur will start on the main note and end on the last one in the grace.

Ornaments

Symbols for ornaments are entered *after* their associated note symbol. The ornaments now available are shake (`ot`), **m**ordent (`om`), "x"- or "+"-shaped ornament symbols (`ox, o+`), pizzicato (`ou`), strong **p**izzicato (`op`), left parenthesis before notehead (`o(`), right parenthesis after notehead (`o)`), upper **f**ermata (`of`), **d**own **f**ermata (`ofd`), staccato (`o.`), tenuto (`o_`), se**g**no (`og`), arbitrary-length wavy-line **t**rill with *tr* (`oT`), arbitrary-length wavy-line trill without *tr* (`oTt`), sforzando (`o>`), and duncecap (`o^`). All except the parentheses, staccato, tenuto, and down fermata will appear above the staff; the parentheses appear at the level of the note head, and staccato and tenuto just above or below depending on the stem direction. (The only difference between staccato and pizzicato is the vertical positioning of the dot.) The trill and segno symbol may have additional optional characters. Either trill symbol may include a decimal number to specify the length of the printed symbol in current `\noteskip`s; the default is 1. Thus `oT0` represents *tr* with no wavy line. A segno can only appear in the first (lowest) voice. It may be immediately followed by a positive or negative integer, which indicates a number of points that it will be offset horizontally; and it will appear above every system.

Once the ornament type has been specified, most of them can be raised or lowered from their default position by appending a signed integer to the symbol, representing the vertical offset in `\internote`s.

If an ornament is to be repeated on a series of consecutive notes, end the first ornament symbol with `:` . The every note in that line of music will then have the same ornament until a note is followed by the repeat terminator `o:` .

Editorial accidentals

To place a small sharp, flat, natural, or question mark above the staff, after the affected note enter `oe` followed by `s, f, n` or `?`. You may also put a question mark right after the accidental. 2.2

Slurs

The normal symbols for slurs are `(` placed with a space before a note, and `)` placed after. The symbol `s` is equivalent to *both* of them (!), except that it always follows the affected note. A fourth symbol `t` is also equivalent to `s` and `)` but with one minor difference to be explained later. In truth there are all *toggles*, turning a slur off if it's already on and starting one otherwise.

The first character is optionally followed by a single-character ID code (`0-9` or `A-Z` , then by other options described below. ID codes are only needed if two or more slurs are open at the same time within one line of music, such as when several chord notes are tied. Using ID codes in such cases tells **PMX** which open slur to close.

The rules for finding the default direction and position of the a slur are complex; many factors enter into defining visually pleasing values. But there's no need for gory details here; the result will usually satisfy, and if not, all can easily be tweaked. Default direction can be overridden with `u` (**u**pper), `l` (**l**ower), or equivalently `d` (**d**own). Starting or ending position can be shifted from its default by entering one or two explicitly signed numbers. The first, which must be an integer, represents the vertical shift in `\internote`s; the second, which may be decimal, the horizontal offset in notehead widths.

The shape of the slur may be altered as well, although the results may be less than fully satisfying due to fundamental limitations of MusiXTEX. At the slur termination only, one or three more parameters may follow the two just described. The first, a signed, nonzero integer, is a vertical adjustment to the mid-height of the slur in `\internote`s. The next two, integers between 1 and 7 following a ":", are alterations to the starting and ending slopes. These numbers are passed directly as arguments of the MusiXTEX macros `\midslur` (if only one is given) or `\curve` (if there are three). 2.1

A dotted slur is activated by including the option `b` (for **b**roken) in the symbol that starts the slur.

Slurs involving grace notes are specified within the symbol for the grace.

The unique aspect of `t` slurs is that if one starts or ends on the same note as an `s` slur, the former will be moved away from the notehead to avoid a collision. *This only works if neither slur has an ID code.* This feature is retained mainly for backward compatibility.

The available options should cover most circumstances, but if not, the TEX macros `\isu` etc, defined in `pmx.tex`, can be entered as in-line TEX commands (see section 2.4). These commands have three arguments: slur number, vertical position (pitch, or offset from bottom staff line in `\internote`s), and horizontal offset in notehead widths. When using these commands, begin slur numbering at 0 to avoid conflicts with **PMX**'s automatic slurs, which are numbered from 11 downward. Also, remember that non-spacing in-line TEX commands such as this one must come *before* the note they apply to, in contrast with the **PMX** slur toggles which may come after. 2.1

Ties

The only difference between ties and slurs is the default positioning. Ordinary slur ends are centered horizontally above or below the notehead, while tie ends are shifted inboard and closer to the midheight of the notehead. To specify a tie, use a slur symbol and include the option `t` in it, somewhere after the initial `(` , `)` , `s` or `t` .

Beams

For the most part, **PMX** automatically takes care of the details of defining beams: selecting which notes are beamed together, and setting the angle, direction, height, and *multiplicity* (the number of bars along the top or bottom). However, one may define a *forced* beam – which overrides **PMX**'s selection of which notes are beamed together – by surrounding the included notes with `[` and `]`, being certain to separate these symbols and their options from the included note symbols with spaces. One may also wish to edit certain features of a beam even when **PMX**'s grouping decision would otherwise be acceptable; here again the beamed notes must be set apart with `[` and `]`. The `[` may optionally be followed immediately by several options.

`u` or `l` will override **PMX**'s selection of the direction of the beam.

`j` **j**oins the beam grouping to a prior one started in another system (see below).

One, two, or three consecutive integers, each preceded with `+` or `-` , will affect the beam's appearance. The first integer is an adjustment to the starting level (in `\internote`s) and may range from -30 to 30; the second is a slope adjustment with the same permissible range; the third is an alternate adjustment to the starting level (in beam thicknesses) and may only range from 1 to 3, always acting to increase the stem length. The latter may be used to align consecutive horizontal beams which have internal multiplicity changes. For example, in 2/4 time, `c84 c1 c c c c8` would cause two beams but the first one would be lower than the second; `[+0+0+1 c84 c1 c ] c c c8` would align the tops of the beams with each other. Due to the complexity of **PMX**'s beam analysis procedures, these editing commands may sometimes produce unexpected results, and some iteration may be required to get exactly what you want. For example, `[+0+0+3 cd8 c3 c6 c ] c c c3 cd8` will not produce two aligned beams as desired, because when **PMX** analyzes the first beam, it automatically raises the starting level a bit for another reason, namely, to avoid too short a stem on the 64th notes at the end of that beam. In this case, the user could counteract **PMX**'s internal adjustment by using `[-1+0+3 cd8 c3 c6 c ] c c c3 cd8`.

The character `m` followed by a digit 1-4 forces the **m**ultiplicity of the beam, the number of stem-joining bars.

The option `h` forces the beam to be **h**orizontal.

By default, xtuplets are set apart with their own beam. To beam an xtuplet together with other non-xtuplets, just include it with the other notes in a forced beam.

Rests may also be included within forced beams, provided they are shorter than quarter rests, and of course that they come *between* the first and last notes under the beam.

Some users may wish to define beamed groupings with subgroups joined by a single beam. The symbol `][`, standing alone between two note symbols in a forced beam, causes the multiplicity to decrease to unity and immediately increase to its natural value for the next note. For example, `[ c14 c c c ][ c c c c ]` will generate two doubly-beamed groups connected by a single beam.

If there are large jumps in pitch between notes in a beam within a single staff, as a matter of taste you may wish to start the beam for example as an upper one and end it as a lower. **PMX** will never do this automatically, but you can accomplish it by forcing the beam with appropriately modified up/down-ness, starting level, and slope. If you use this technique, there are two details to note: (1) if there are any intermediate multiplicity changes, they will only be handled properly if the initially specified up-down-ness is consistent with the vertical position of the intermediate notes involved, and (2) for proper appearance in crowded scores you may wish to insert hardspace or shifts as described in section 2.3 on page 14. Some examples are included in `most.pmx`.

Beams cannot normally jump staves. But if that is desired, start the beam normally in one voice, and terminate the part of the beam in that voice with `... ]j` . Then resume the beam in the new voice with `[j ...` . For staff-jumping beams, it's OK to have just a single note inside one or both of the members. Some adjustment of the beam height and slope may be required. Sometimes the ending section's up-downness must be overridden; you will know this is so if the ending is shifted horizontally from its proper position. Each line of music must still have the right number of beats, so you will probably need to fill time with blank rests after the first member of the beam and before the second. In version 2.1 we have removed certain restrictions on the multiplicity and jump direction for staff-jumping beams. 2.1

Clefs

A clef change is signaled by `C` followed by a single lower-case letter using the code specified at the end of section 2.1 on page 4. Numbers may also be used as defined in the MusiXTEX documentation. If clefs come out at the wrong vertical position, refer to the note in `pmx.tex`.

Arpeggios

To set an arpeggio (a vertical wavy line), simply place the symbol `?` after the symbols for both the first and last note.

2.3 Commands That Affect All Voices

Most commands that affect all the voices can only appear in the first (lowest) line of music in the first (lowest) staff. Most such commands will automatically be transferred from score to parts when separate parts are generated by **scor2prt** (see section 3, page 17).

Repeats, double bars, forced single bars

Repeat signs and double bars are signaled by `R` followed by `l, r, lr, d, D,` or `dl` for **l**eft repeat, **r**ight repeat, **l**eft-**r**ight repeat, thin-thin **d**ouble bar, thin-thick **D**ouble bar, or thin-thin **d**ouble bar followed by **l**eft repeat. `Rb` forces a single bar before a movement break (page 14), where otherwise by default there is a double bar. That can be useful for example if you change the number of instruments (via an option in the movement-break command) in the middle of a movement. These symbols must be in the first voice. It is best only to place them before the first note in an input block or if necessary after the last one; otherwise **scor2prt** may behave erratically.

Using two separate `R..` symbols in succession will cause unpredictable results.

If `Rlr` falls at a system break, **PMX** will automatically split it in two. The symbol `Rdl` will likewise be split at a system break, but if not at a system break, the `d` will be ignored.

Voltas (first and second endings)

Beginnings and ends of first and second endings are signaled by `V` (for ***v**olta*). If it's the *end* of the volta, enter `b` (for ***b**ox*) or `x` for *no bo**x***. If it's the *start* of a volta, you can optionally enter any text at all that doesn't include a space and doesn't start with `b` or `x` (most commonly `1` or `2`). A period will automatically be appended to the text. If one volta ends and another starts right away, only a single `V` is needed. Voltas must only be entered in the first voice. If separate parts are to be created from a score using **scor2prt**, then only a single volta may appear in any given input block, and it must be at the beginning of the block.

Meter changes

Meter can only be changed at the beginning of an input block. A ***m**eter change* symbol starts with the letter `m`. There are two different ways to complete the symbol.

Method 1. Enter 4 numbers with no intervening spaces. The four numbers are `mtrnuml`, `mtrdenl`, `mtrnmp`, `mtrdnp` as defined in section 2.1, with the following exceptions for this method only: You must use `o` to represent the number 1; if you enter the digit `1` then **PMX** will interpret that digit and the next as a 2-digit integer, between 10 and 19 inclusive. 19 is the largest number that can be entered with this method. Note that `mtrdenl=0` still represents a whole note.

Method 2. Enter the four numbers in the same order as described above, but separate them with slashes (/).

Transposition and key changes

To transpose an entire score, at the beginning of the first block enter `K` (for ***K**ey*) followed by two explicitly signed digits. The first is the distance to transpose (in `\internotes`); the second is the new key signature. When transposing, you should always use relative accidentals, activated by the separate symbol `Ar` at the start of the first input block (see below). For example, to transpose a piece in C major to E major you would enter `Ar K+2+4` at the beginning of the first block.

A key change can be signalled at any time in the first voice, and will affect all voices. Use the command `K` with `+0` as the first argument and the new key signature as the second.

Text

The symbols `h` or `l`, when placed in the first column of an input line and followed by a blank or, for `h` only, by a signed integer, stand for ***h**eader* and ***l**ower text*. They will put a text string above or below the *top* staff in the *first* bar of the block where they are entered. The text string must be on a line of its own, immediately following the symbol. The integer is a vertical shift in `\internote`s.

A *title block* with up to three elements can be defined at the beginning of the first input block. `Tt` signals that the text *on the following line* is to be set as a **t**itle for the whole piece, and it will be centered. `Tc` similarly indicates a **c**omposer's name, to be set below the title and right justified. `Ti` likewise stands for an **i**nstrument name, which will be set above the title, left-justified. `Ti` will automatically be invoked by **scor2prt** when it generates parts from a score. Extra vertical space can be added between the title block and the top system by appending to `Tt` a one- or two-digit number representing the space in `\internote`s. This only works if `Tt` is the *final* title block element entered.

Page numbering, centered header text

If you want pages to be numbered at the top left or right, place the symbol `P` anywhere within the **PMX** code that represents the first page to be numbered (usually the first or second). `P` can be followed optionally by the starting page number and/or by `l` or `r`, the latter overriding the default locations of odds on the right and evens on the left. There is also a special option `c` for **c**entered header text. It must be the *last* option in the `P` symbol. It will define text to be printed at the top of every page *after the first*. If a blank follows `c`, the default header text will be the instrument name entered with the symbol `Ti` . If any non-blank character except `"` follows `c`, the header text will start with that character and end at the next blank. If `"` follows `c`, the header text will be everything between that and the next `"` (this permits headers containing spaces). The `P` symbol and its options will be ignored when making parts from a score (since page numbering will usually be different in the score than in the parts), but page numbering (and centered headers) for parts can be still be initiated independently, for example with `%!P2` or `%1P2r` (see section 3, page 17). 2.0

Overriding defaults for accidentals, dot positions, spaces at start of bar and between staves

The symbol `A` can be used to override a grab-bag of default settings:

`b` makes all accidentals **b**ig, `s` makes them all **s**mall. By default, big ones are used unless unaltered spacing doesn't provide enough space. Thus the default behavior may cause a mixture of big and small accidentals, and in fact is not recommended.

If transposing, then the *relative* accidental convention should be used, indicated by `r`. The default is the normal, absolute convention.

In staves with two lines of music, `d` causes dots in the lower line to appear on or *below* center, in contrast with the default.

Use `a` followed by a decimal number to override the default setting for `\afterruleskip`, the space before the first note in a bar. The default in **PMX** is `1\elemskip`, 20 percent smaller than MusiXTEX's.

If **PMX**'s vertical spacing between staves within a system is not pleasing, use `I` or `i` , followed by a decimal number, to apply a scale factor to `\interstaff` . `I` affects all pages, `i` only the current one. Shrinking the space between staves within each system will cause the space between systems to increase, and conversely.

Extra hardspace, horizontal shifts

Despite the author's best intentions to relieve you of the chore of adjusting *any* horizontal spacing by hand, there may be some occasions where you will want to do it. A symbol starting with `X` initiates one of two types of horizontal adjustment: A *shift* moves one or more characters but does not affect any other spacing anywhere; a *hardspace* inserts a fixed amount of space at a particular time and affects the horizontal positions of everything in all staves in the system. If the symbol includes `S`, it is a ***s**ingle* shift and affects only the next note or rest. If it includes a `:` it either starts or terminates a *group* shift. All `X` symbols except group shift terminations must include a decimal number; in fact it is the absence of a number (and presence of `:`) that signals a group shift termination. If the number is immediately followed by `p`, then the number represents points, otherwise, notehead widths. Group shifts symbols must occur in start/terminate pairs, and group shifts cannot extend across a bar line.

An `X` symbol with no `S` or `:` is automatically a hardspace.

Because horizontal spacing in parts will usually differ from that in the score, by default the hardspace command will *not* be copied into parts by **scor2prt**; however the shift commands will be copied. These behaviors can be overridden using the methods to be described in section 3, Alternatively, to help keep **PMX** score files neat and readable, the character `B` can be used within the `X` symbol to signify that it applies to **b**oth score and part, or `P` for **p**art only.

Minimum spacing between notes in crowded systems

PMX does some special, complex analysis to adjust horizontal spacing in crowded systems. By default, the minimum space between consecutive noteheads is 0.3 notehead widths. If you want to change 0.3 to some other fraction, enter `W.` (decimal point is required) followed by `1-9` to represent the number of tenths of a notehead width to be used as the minimum spacing.

Page size

The default page size is 740 by 524 pt (10.3 by 7.3 in). To change the height or width, use the special symbols `h`[*n*][*u*] or `w`[*n*][*u*] at the beginning of the first input block. Here *n* is a decimal number for the new dimension and *u* defines the units; `i` for inches, `m` for millimeters, and `p` or nothing for points. This command can be used together with `%%` or `%!` (see section 3, page 17) to give the parts made by **scor2prt** different page sizes than the parent score.

Line, page, and movement breaks

It is possible to force line, page, or movement breaks anywhere. For a line break, just enter `L`[*n*] at the start of an input block (in the first voice only), and the *n*-th system will start there. To start page *m* at line *n*, enter `L`[*n*]`P`[*m*]. You can't force a page break without first forcing a line break.

To force a movement break, you must first force a line break as above, then enter `M`. If a page break also occurs here, the `P` must precede the `M`. Options following `M` are `+`[*integer*] to insert vertical space in `\internote`s before the break, `i`[*decimal number*] to reset the first-line indentation as a fraction of the line width, and `c` to **c**ontinue bar numbering rather than resetting the bar number to 0. Also, to change the **n**umber of instruments, enter `n`[*integer*], then the number of each instrument in their new order, then a clef symbol for each staff of each instrument. (An instrument's number is simply its position in the original sequence.) There can never be more than the original number of instruments.

Another option after `M` is `r+` or `r-`, which either forces or suppresses **r**eprinting the instrument names. The default is to print them if the number of instruments changes, but otherwise not.

Immediately after a movement break, any desired meter changes, key changes, or text can be entered in the normal way.

Fractional bars

Often if a piece starts with a pickup, the last bar may not be complete. In such cases, it is usually possible to place the last bar in an input block by itself, headed by a *blind* meter change. For example, if the meter had been 4/4 and there was a quarter note pickup, leaving 3 beats in the last bar, the last bar might be coded `m3400 cd24 /`.

Stem direction of bass notes

By default **PMX** makes stems go up for middle-line D's in bass clef, but down for notes on the middle line of all other clefs. If you want middle-line bass-clef notes also to have downward stems by default, enter a `B` near the beginning of the file.

2.4 Putting TEX Commands into the `.pmx` File

There are five ways enter TEX commands into the .pmx file. Four of them are *in-line*, where the commands are entered directly; the fifth is by way of an external file.

The four categories of in-line TEX strings differ mainly in where they will appear in the `.tex` file. (A TEX *string* consists of a starting character, a sequence of TEX commands, and a terminal character). In the `.pmx` file, only type 4 TEX strings may wrap over line breaks. All in-line TEX must adhere to the 128-character limit per line, but each line can have more than one TEX command. Type 1 begins with a single `\` and will appear in the `.tex` file right before the TEX command for the next note or rest in the `.pmx` file. Starting with version 2.1, multiple type 1 strings associated 2.1
with the same note or rest are allowed, although the total length may not exceed 128 characters (so there is generally no reason not to combine all TEX commands for a single note into a single type 1 string).

A type 2 string begins with `\\` and will appear at the top of the `.tex` file, right before `\startmuflex`, regardless of where it appears in the `.pmx` file. A type 3 string starts with `\\\` and will appear right before the `\xbar` or `\alaligne` at the beginning of the current input block, before the first barline of the block. While individual type 2 and 3 strings may not wrap over line breaks, strings of like type entered on consecutive lines will appear together in the `.tex` file. Types 1, 2, and 3 strings must end with `\` (backslash-space). This means that they may not contain the TEX macro `\` (backslash-space). Finally, each type 2 or 3 string should be isolated on a line of its own, and should be started in column 1.

Type four permits multiple lines of arbitrary text to be entered at the top of the `.pmx` file; they will be transferred literally to the top of the `.tex` file. Type four is initiated with `---` alone as the top line of the `.pmx` file. Then follows any text on any number of lines, until the next line starting with `---` terminates the block to be transferred.

The only other distinction among the types of in-line TEX strings arises when **scor2prt** is used to make separate parts (see section 3, page 17): types 2-4 will be copied into all parts, while type 1 only goes into its original part.

If you should want to enter a type-1 (note-based) string longer than 128 characters, you could use a series of type-2 or -3 strings to define a TEX macro containing the desired commands.

PMX provides one further option for entering an unlimited set of TEX commands just before `\startmuflex`, and before any Type 2 in-line TEX strings. Simply put the commands into a text file named [*basename*]`.mod` in the texinput directory. It will then automatically be entered with an `\input` command. This feature is retained mainly for backward compatibility; it has been essentially replaced by the various options for in-line TEX strings.

2.5 Figured Bass

Figure symbols are entered *after* their associated note symbols. They only work in the first (lowest) voice. Enter the characters as they would appear from top to bottom, and as you might pronounce them, e.g., `64` or `73`. Flats here are `-` (minus), sharps are `#`, and naturals `n`, *before* the number (if there is a number) (notice the characters are different here than in notes). So for example *sharp third* is `#3`, just a sharp is `#`, *six (over) flat five* is `6-5`, and *sharp six (over) 4* is `#64`. The program positions all the figures for each system below the lowest staff of that system, with their tops aligned, and just low enough to clear the lowest beam, notehead, or stem that could interfere. If you want a figure to align horizontally in the second tier, insert the placeholder figure `_` (underscore) before the one you want lowered.

Sometimes you may need to enter a figure when there's no bass note sounding. To do this, just after the most recent bass note enter `x`, followed by a two single digits (the first is a repeat count; the second a time value, i.e., `2,4,8,1,` or `3`), immediately followed by a figure symbol as defined in the previous paragraph. This will offset the figure from the associated note by the specified time value. For example, if the lowest voice contained `c03 x3465`, there would be a whole-note c, and 3 quarter notes later a figure 65 below the staff.

There is also a *continuation* symbol, a zero followed by another digit. This produces a horizontal line under the bass note, starting just to the left and extending to the right by the given number of `\noteskip`s. The height and length of the line are set by the current note's level and `\noteskip` respectively. If `\noteskip` changes or a note drops below the starting level before the line ends, it is possible to trick **PMX** by entering separate `0`[*n*] symbols under each consecutive note; **PMX** will automagically join them together at the same height (thanks to Werner Icking for this idea). **PMX** can't directly handle continuation symbols above or below other figures. If you want to do that you could look for the macros `\Figu` and `\Cont` in `pmx.tex` and work out how to use them manually as in-line TeX commands.

If there are figured bass commands in a `.pmx` file but you want them to be ignored, then enter the symbol `F` at the beginning of the body of the file. This feature would most often be used in the form `%1F` (see section 3, page 17), which make a separate bass part with no figures.

Figured bass symbols will not be altered in any way under transposition. There is no universal set of interpretations of figured bass symbols, so no automatic transposition is possible.

2.6 Macros

A **PMX** macro is a single symbol that stands literally for any any string of characters that may occur in the input file (sorry, no variables). It may be useful if you need to repeat the same string later. There is no practical length limit.

To **r***ecord* a macro, type `MR`*n* where *n* is between 1 and 20. Everything you then type will be processed normally as well as stored, until you enter the symbol `M`. The next time you need to enter the same string, just type `MP`*n* to **p**lay back the macro.

To just **s***ave* a macro without having **PMX** process it as you enter it, start it with `MS`*n*.

Macros can be redefined at will. **PMX** will print a warning whenever this occurs.

If you use macros and want to make separate parts, some care is necessary. **Scor2prt** will only transfer `MR` macros into the part where they originated, but will transfer `MS` macros into all parts.

2.7 Batch Processing

Due to the number of different programs that must be run in sequence to produce a printed sheet 2.0
of music with the MusiXTeX system, most users prefer to use a batch file to control the process. Since batch commands are platform-dependent we will not provide examples here, but will mention several **PMX** features that can facilitate batch processing.

First, whenever **pmxab** terminates due to a syntax error, the exit code is set to 1. There are various ways of detecting this with batch commands, then acting accordingly. Second, **pmxab** always writes a file `pmxaerr.dat` containing a single number: 0 if it exited normally, otherwise the line number in the `.pmx` file where the syntax error was. With advanced batch programming techniques, this file can be opened and read, and if there was an input error, a text editor can be opened and the input point placed on the line with the error.

There have been several requests to allow **PMX** to keep running even after it detects an input error. This has not been done because in most cases, any error messages after the first one would be meaningless, or worse, uncorrected errors could cause crashes. In any event, all the output from **pmxab** will be stored in the log file [*jobname*]`.pml`.

2.8 Lyrics

PMX has no special provisions for lyrics. One way to include them is by using the macro package `musixlyr.tex` developed by Rainer Dunker. It introduces lyrics into TEX more easily than with MusiXTEX's own facilities. The macros could be entered as in-line TEX directly into the `.pmx` file, but most would prefer the convenient interface to `musixlyr` via the program **M-Tx** developed by Dirk Laurie. It is a pre-preprocessor which produces a `.pmx` file containing the proper in-line TEX commands. Its input language is similar (but not identical) to **PMX** and includes most **PMX** functionality as a subset. Both **M-Tx** and `musixlyr.tex` are available on `ftp.gmd.de` (see section 6).

3 Making Parts from a Score

Separate parts can be made by running **scor2prt** and entering the basename when prompted. The program will create `noinst` separate `.pmx` files, one for each instrument. The files will be named [*basename*][*n*]`.pmx`, where [*n*] is the number of the voice.

In this section we describe how to control the layout of the parts separately from that of the score, but by using commands that are placed in the `.pmx` file for the score. This eliminates the need for ever editing the `.pmx` files for the parts separately. You can make all corrections in the file for the score, and then re-run scor2prt.

Normally all lines starting with `%` in the parent `.pmx` are transferred into all the parts. However, if a line has `%%` in columns 1-2, both it *and the following line* will be ignored when making parts. If the ignored line contains only `h` or `l` in the first column, then one additional line will be ignored.

Conversely, if a line begins with `%!` then it will be ignored as usual in creating the parent `.tex` file, but after stripping the first 2 characters the rest will be put in the `.pmx` file for *all* the parts.

To enter a line into the score file that is only to be transferred to one part, begin the line with `%`*h*, where *h* is the hexadecimal digit representing the part number (`1, 2,..., 9, a, b, c`). The 2.1
first two characters will be stripped and the rest transferred to the desired part. For example, to force a line break to system 15 and a page break to page 2 in part 11 only, enter `%bL15P2`. The use of the hex digits `a-c` creates a potential incompatibility with prior versions. To minimize this, the character after "`%`" will *only* be interpreted as a part number if it represents a number less than or equal `noinst`; otherwise the entire line will be treated as an ordinary comment and transferred to all parts as a comment.

Although only permitted in the first voice in the score, the following symbols with all their options will automatically be copied into all parts (unless the preceding line has `%%`): `m, V, R, A, h, w, K`. Literal TEX strings of types 2-4 will also be copied into all parts, while type 1 will only go into its original part.

User-defined hardspaces (`X` without `:`) are handled specially. By default they are not copied into parts. There are two ways to circumvent this. One way to insert hardspace *x* into part *n* is to place in the score, on a line of its own, the symbol `%`[*n*]`X`[*x*] . The other way is with options in the `X` command in the score: `B` causes the hardspace to be used in **b**oth score and parts; `P` puts it into the **p**art but not the score.

Lateral shifts (`X`[. . .]`:`) will be handled normally, staying with their original line of music.

By default the total number of systems in each part will be the same as in the score. If you want to override this, there is a symbol `S`[*n*] (where *n* is the desired number of systems), which can only appear at the beginning of the first input block. This can be used after `%!` to affect all the parts, or after `%`[*k*] to affect just part *k*. **Scor2prt** will also compute how many pages it thinks each part should have, and enter that in the startup data for that part. If you wish to override that, then in the `.pmx` file for the score, insert for example `%3S14P2` to force the third part to have 14 systems and 2 pages (you cannot override the number of pages without first overriding the number of systems).

A musicsize of 20 is the default in all parts. This may be overridden with the option `m` in the symbol `S`; e.g., `%2S15m16` . 2.0

As already noted, a `P` symbol for page numbering in the parent file is ignored when making parts. To initiate page numbering in the parts, use for example `%!P` anywhere within the **PMX** code representing the first page of the parts (from TEX's standpoint the command must occur between the beginning and end of the page on which the numbering is to begin). It will often be useful in this case to use the option `c` , which by default causes the instrument name to be centered in small type at the top of every page after the first. 2.0

Note the distinctions among the various usages of `P`: as an option with `S`, it sets the total number of pages in a part; as an option with `L`, it forces a page break; and as a command on its own, it controls page numbering and centered headings.

MIDI commands, i.e., those starting with `I`, will never be copied into parts, unless they are in a special comment line as just described. 2.2

One function of **scor2prt** is to condense consecutive bars of rest into a single group of special printed characters with a number above it. The symbol `rm` defines such a **m**ulti-bar **r**est as described in section 2.2. **Scor2prt** will automatically insert `rm` symbols into the `.pmx` files for the parts where appropriate. However, for this feature to work, the *first* full-bar rest in the sequence *must* have its duration explicitly defined in the parent `.pmx` file, either with a digit or with `p`. I.e., the feature will not work if the first rest in the sequence inherits its duration from the previous note.

Using the special **PMX** commands listed in this section, augmented where needed with literal TEX commands, it is possible to store *all* the information for both the score and the parts in a single `.pmx` file. This greatly simplifies the editing process, since both the score and the part can be corrected at once, and parts need not be re-edited each time they are regenerated from the score.

4 Making MIDI Files

2.2

Entering the command `I` before any notes have been entered will cause a file [*jobname*]`.mid` to be written in current directory. Options may follow, without spaces. They are defined in the following paragraphs. Multiple options can be combined in one `I` command. `I` commands can appear later in the file as well, but only at the start of an input block. Sometimes the order of the options matters, determining for example whether or not a user-defined pause is included inside a macro block.

`t`*x* sets the tempo to *x* quarter notes per minute. Default is 96. You can change tempos as often as you like, but only at the start of an input block (as with all MIDI commands).

`i`*i1i2...in* assigns MIDI instruments *i1,i2,...,in* to the respective **PMX** instruments. The default is harpsichord, of course. If you use this option, you must specify *all* instruments. Each *in* is either a 2-letter abbreviation or an integer between 1 and 255. Acceptable abbreviations are listed below. Numbers and pairs of letters may be mixed, but consecutive pairs of numbers must be separated by : (colon) . This option can only be exercised once per file. Also, the number of instruments cannot change during a piece.

`M` initiates a macro operation. This is used for repeats, da capo's, etc. Macros must have ID numbers between 1 and 20. Operations are start record macro i: `MR`i ; end recording: `M` ; and playback (insert) macro i: `MP`i . Only one macro can be active at a time, recording or playing but not both. If you try nesting or overlapping macros, your computer will become psychotic.

`p`x inserts a pause of x quarter notes. Decimals are allowed, but will be rounded to the nearest sixteenth note.

`g`i Sets the MIDI gap to i MIDI clock tics. This is a silence inserted at the end of every note, while decreasing the sounding duration by the same amount. The default is 10, which is 2/3 of a 64th note.

The module does not recognize graces, ornaments, repeats, voltas, or segnos. The only ties that are recognized are those using `s` or `(` alone, with no explicit ID number. Key signatures, time signatures (meter) and instrument names will be written into the MIDI file, the latter as track names. This will have no effect whatsoever on audible output but will affect on-screen appearance of some MIDI file players and editors. Location of the **PMX** key-change and meter-change commands relative to MIDI macro delimiters in the source will affect (in the obvious way) how these data are passed to such programs.

The MIDI file generator does not yet support changing the number of instruments in midstream. Doing so will cause unpredictable results.

The instruments are a subset of "The General MIDI Instrument Specification." Of course how they sound depends on your hardware and software. Instruments not listed below can still be used but must be specified by number. The numbers listed here are from the 1-128 range; when passed to the MIDI file they are reduced by one.

`pi`	Acoustic Grand Piano (1)	`va`	Viola (42)	`al`	Alto Sax (66)
`rh`	Rhodes Piano (5)	`vc`	Cello (43)	`te`	Tenor Sax (67)
`ha`	Harpsichord (7)	`cb`	Contrabass (44)	`bs`	Baritone Sax (68)
`ct`	Clavinet (8)	`vo`	Synth Voice (55)	`ob`	Oboe (69)
`ma`	Marimba (13)	`tr`	Trumpet (57)	`ba`	Bassoon (71)
`or`	Church Organ (20)	`tb`	Trombone (58)	`cl`	Clarinet (72)
`gu`	Acoustic Nylon Guitar (25)	`tu`	Tuba (59)	`fl`	Flute (74)
`ab`	Acoustic Bass (33)	`fr`	French Horn (61)	`re`	Recorder (75)
`vl`	Violin (41)	`so`	Soprano Sax (65)		

5 Limits

For simplicity in writing the program, **PMX** has numerous variables with fixed dimensions. In most cases there are no checks against these limits (hey, I've got more important things to program), so occasionally there may be hangups due to exceeding a dimension. Any of these can be increased on request. However, before making such a request, it will usually be possible to work within existing limits by breaking the input into smaller blocks.

Limits under direct user control

128 characters per input line.
12 voices (staves). 2.0
2 lines of music per staff.
12 lines of music. 2.0
125 systems.

500 bars.
40 forced line breaks.
10 forced page breaks.
18 key changes.
20 pages.
200 notes per input block.
15 bars per input block.
101 slurs per input block.
74 figures (figured bass) per input block.
37 grace note groups per input block.
74 notes in grace note groups per input block.
52 literal TEX strings per input block.
6 voltas per input block.
18 trills per input block.
62 chordal notes (non-spacing) per input block.
8 beams per line of music per bar.
20 forced beams per line of music per input block.
10 clef changes per line of music per input block.
24 notes per beam.
24 notes per xtuplet.

Limits not under immediate user control

2000 `\notes` groups.
20 `\notes` groups per bar.
20 inserted standard anti-collision spaces (not xtuplet or end-of-bar) per bar.
20 inserted anti-collision spaces within xtuplets per bar.
19 inserted anti-collision end-of-bar hardspaces per system.
83 inserted anti-collision end-of-bar hardspaces.
400 inserted standard anti-collision spaces per system.
100 inserted anti-collision spaces within xtuplets per system.
1000 inserted standard anti-collision spaces.
200 inserted anti-collision spaces within xtuplets.
24576 bytes of MIDI output data per line of music. 2.2

6 Closing Notes

About the Example Files

`most.pmx` contains examples of most of the **PMX** commands, and a few programming tricks, including examples in the last line of beam groups whose notes vary widely in pitch. The printed output displays the **PMX** commands near to the resulting typeset characters. It is more useful to look at the printed output rather than the source file, since the file is littered with in-line TEX needed to output the text strings representing the **PMX** commands. **WARNING:** Do not try to play this music; it could be hazardous.

`barsant.pmx` contains the first movement of a recorder sonata by the Italian Francesco Barsanti (1705-1765). It demonstrates many of **PMX**'s strong points in a "battlefield" situation: figured bass, complex beaming patterns, xtuplets, and automatically adjusted horizontal and vertical spacing in crowded scores. In fact, this single-page score is at the limit of vertical crowding. If you added anything that occupied additional vertical space, e.g., by putting a header on the first system or inserting a title, there would not be enough vertical space available on the page and the last system would spill over. Some vertical space could be freed up by switching to 16-pt staves.

`rtsg.pmx` is a trio sonata by Johan Helmich Roman (1694-1758), "the Swedish Handel". It demonstrates how to set movement breaks for the score, and how to use this command and some others to control the appearance of the parts independently.

`mwalmnd.pmx` is an Allemand for harpsichord by the German Matthias Weckmann (1616-1674). It uses many techniques peculiar to keyboard scores.

A Benign Bug

When TEX'ing the output of **PMX** you will usually get an `Underfull \vbox` message at the end of each page. This is due to my using `\eject` at the end of every page, which automatically spaces the systems vertically without having to fiddle with `\staffbotmarg`. As far as I know, the warning is benign, and may be ignored.

Where to Get PMX

The main home of **PMX** on the internet is the GMD server at `http://www.gmd.de/Misc/Music/` , or if you prefer FTP, `/ftp.gmd.de/music/` . **PMX** files on this server will propagate to the CTAN sites at `www.ctan.org`, `ftp.tex.ac.uk`, and `ftp.dante.de`, and their mirrors.

As of the date of this writing, the Macintosh and UNIX ports of this version of **PMX** are not yet available, but previous ones are, and the new ones may appear soon at the following sites or on the GMD server.

The previous Macintosh port by Eric Petersen is available at the GMD server or at Eric's very spiffy web site `http://www.aem.umn.edu/people/students/epeterse/PMX/` .

Stefan Evert has translated the previous version of **PMX** into C and prepared a user-friendly distribution which should be compatible with most UNIX systems. It is located at `ftp://ftp.mathematik.uni-stuttgart.de/pub/software/pmx/` .

Dirk Laurie's lyrics pre-preprocessor **M-Tx** is available from the GMD server.

Acknowledgments

To Daniel Taupin, Ross Mitchell, and Andreas Egler for creating MusiXTEX, to Olivier Clary for suggesting a crucial modification in the note-entry scheme, to Robin Fairbairns and Werner Icking for including **PMX** in their archives, to my colleague John DiPol (a non- musician!) for the idea of using binary masks to define beam groupings, to Dirk Laurie, Eric Petersen, Johan Tufvesson, and Stephan Evert for beta-testing and for porting **PMX** to other platforms, and to Dirk Laurie for making **PMX** accessible to vocal music by creating **M-Tx**. Special thanks and credit go to Werner Icking for exhaustive beta testing, uncanny bug-finding, and continuing encouragement.