PMX – a Preprocessor for MusiXTeX

Version 1.1 – February 3, 1997

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

Contents

1 Introduction

PMX can help you typeset a musical score if (a) you have MusiXTEX version 0.396T or later installed and (b) your score has from one to seven staves (or voices) per system. Executables are available for DOS, Macintosh, and Linux, as are C and FORTRAN source code. 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 a specified number of systems, 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 dotted whole notes to 64ths, trills, mordents, slurs, and limited text strings. Every voice in every bar must have exactly the correct number of beats in the current meter. You may change the meter at the beginning of any measure, with or without printing the new time signature. On the first pass, **pmxa** 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 using **PMX**'s facility for handling 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 score. You can control the appearance of the parts with special commands in the main input file for the score. It is thus possible to include in a single input file all the information that defines the score and the individual parts.

Conventions for This Manual

Sure, 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., `barsanti.pmx, \internote, c44`.

Bold is used for program names (e.g., **pmxa**), 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.

Setup

This section describes the setup for the DOS version or for those compiling the FORTRAN source. Other versions will be accompanied by a readme that describes peculiarites for that platform.

After decompressing the distribution file if necessary, you should have these files: `pmxa.for`, `pmxb.for`, `scor2prt.for`, three corresponding DOS executables, several sample `.pmx` files, `pmx.tex`, `ref11.txt`, `pmx11doc.tex` (TeX source for this file), `pmxdoc.sty`, and `pmx11doc.ps`. If necessary, compile the FORTRAN programs. I used Microsoft FORTRAN vintage 1987 to develop the code, and have checked that it works on an IBM RISC 6000 when compiled with **xlf**. I have tried to keep the code generic enough that it will work with any reasonable FORTRAN compiler, although I can't say for certain whether it adheres to any particular standard.

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**).

Quick Start

Edit the 6th line of `barsanti.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 the first pass by running **pmxa**. You will be prompted for a *basename*, which is the name of the input (`.pmx`) file without the extension, in this case `barsanti`. If **pmxa** runs successfully it will generate two temporary files in the working directory, `pmxtex.fig` and `pmxtex.dat`. Then run **pmxb**. This will create `barsanti.tex` in the texinput directory. Now you are right where you would be after entering, debugging, and rough-editing the `.tex` file manually. To see the results, process `barsanti.tex` just as you would for any MusiXTeX file, running all three passes, and view the `.dvi` file. To make separate parts, run **scor2prt** and enter the basename when prompted. The program will create a new `.pmx` file for each instrument in [*basename*]`.pmx`, named [*basename*]`1.pmx`, etc. 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 `barsanti.pmx`. For reference, here are the first 8 lines:

```
2 2 4 4 0 6 0 0
1 7 20 0.07
Basso
Recorder
bt
./
%read(10,*)nv,noinst,mtrnuml,mtrdenl,mtrnump,mtrdenp,xmtrnum0,isig,
%     *           npages,nsyst,musicsize,fracindent}
```

The first 6 lines in this example are the *setup data.* The lines with `%` in column 1 are comments. You can only put comments *after* the setup data. Some special handling of comment lines will be discussed in the section on creating parts from a score.

Starting in the first line of the example above,

`nv` (integer<8) is the total number of voices, or staves per system. (We will use the terms *voice* and *staff* interchangeably). Each voice or staff may contain either one or two lines of music.

`noinst` (integer<8) is number of *instruments*. Usually `noinst=nv`. Only the first instrument – the lowest in each system – can have more than one voice. If `nv`>`noinst`, then the lowest instrument will have `nv-noinst+1` voices (staves), and they will be set off with a curly bracket at the left. If `nv`>1 and `noinst=1`, then all voices will be bracketed together. 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. Some rare meters may cause problems with the beam grouping algorithm, but all common ones should work.

`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 the numerator and denominator of the printed time signature. 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

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 tell by watching the page numbers printed to the screen on the first pass through TeX. The 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 `nv` instruments as you want them to appear within the indentation in the first system, one per line, starting with the *bottom* voice. If you've set `fracindent`=0 and don't want instrument names to appear, you must still leave the proper number of blank lines here. Next comes a single string of `nv` letters for the clefs, again starting with the bottom voice: `b, r, n, a, m, s, t` respectively for **b**ass, baritone, te**n**or, **a**lto, **m**ezzo-soprano, **s**oprano, or **t**reble. Line 6 in this example is the last of the setup data, the path to the texinput directory, where you want the `tex` file to go when **PMX** creates it. It 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, usually 4 to 8 (except possibly the first, which may include a pickup). It is good practice to separate the blocks with comment lines showing which bars they represent, as I've done in `barsanti.pmx`. (You can't enter the whole piece as a single block because if any block is too big, the code will choke due to dimension limits). There is also the optional symbol |, whose only functions are to separate measures visually in the input file, and to help isolate input errors. If it appears anywhere except at a bar-end, **pmxa** will stop and show you where it detected the timing error. Otherwise, | has no effect. Its use is strictly optional.

At the start of each block there may be a few special symbols (described in section 2.3 on page 8). 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 on the same staff. Each new line of music should start on a new line in the file. Continue entering other lines of music, each with *exactly* the same number of bars, terminated by `/` or `//`, until the last (topmost in the system) line 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 line in each voice are a sequence of *symbols* of one or more adjacent characters, separated by spaces. The line- terminating symbols `/` and `//` should also 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: `0, 2, 4, 8, 1, 3`, or `6` respectively for 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; `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 **le**ft or **r**ight by its own width; and `a` (for **a**lone) which inhibits beaming for this note. The only other characters allowed in note symbols are `+, -,` and `x`, to be discussed below. After the first letter, except for `x`, non-digits can be in any order with respect to each other and to the digits.

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. 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. And using `+` or `-` is the only legal way to change octaves in xtuplets, to be described below.

For example `c44 d e f g a b c` is an ascending quarter-note C scale.

The first note symbol of each line of music in a block must contain at a minimum the note name 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 use the fact the pitch is remembered from the corresponding line at the end of the prior block (duration is not). If the number of lines per voice has changed from the prior block, it is safest to re-enter the octave at the start of a new block.

Dots can be a little tricky, because even though they affect the actual time value, they don't affect the basic time value. Therefore, if a note is dotted, you always have to enter a `d` somewhere 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).

Xtuplets can have from 2 to 24 notes; the notes must be of equal duration. The initial symbol for an xtuplet begins exactly like a note symbol with the name of the first note in the xtuplet. However, the actual time value (including a dot if present) now represents the total duration of the xtuplet. After this, with no space, comes an `x` followed by a one- or two-digit integer for the number of notes in the xtuplet, then optionally an `n` to suppress the printing of the number, then a space. The second through last notes are each then represented by a separate symbol containing a subset of the characters permitted for ordinary notes: note name (the only required character), octave change character (`+` or `-`), and accidental. No time values or dots are permitted, nor are numbers permitted as octave indicators.

For example, an ascending quarter-note triplet scale would be notated `c44x3 d e f4x3 g a b4x3 c d` ...

Rests

A *rest* symbol has one to three characters, starting 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) and a `d` if the rest is 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 in keyboard music where 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). `rm` followed immediately by an integer will `mwalmnd.pmx` for examples). over it. This symbol will generally only be used in separate parts after having been automatically generated by **scor2prt**.

PMX uses a default vertical position for rests which is different if the voice has one or two lines. For one line it is just the MusiXTEX default (approximately centered on the middle line). On the other hand, in the lower line of a two-line staff 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 necessarily from the **PMX** default).

This completes the description of the basic scheme for entering notes and rests. It can become very intuitive, allowing you to enter a sequence of notes almost as you would read it. For example. as you are typing `cd45 ef8 r` you can be saying to yourself

"**c** **d**otted 1/**4**-note in **5**th octave, **e**-**f**lat **8**th note, (8th) **r**est."

Chords

Chordal notes, which 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 time value or `d` is allowed. Any number of chordal notes can follow a single main note.

Grace notes

A grace note symbol starts with a `G`. Next 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 *multiplicity* (number of flags or beams, default is 1, 0 is allowed), `s` for slur (joining all notes of the grace to the main note), `x` for a slash (only for single graces), and `l` or `u` to force the direction of the stem(s). Next comes 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 be preceded by spaces and likewise must not include a time value.

Ornaments

Symbols for ornaments or slurs 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 or staccato (`ou`), strong **p**izzicato (`op`), right parenthesis (`o)`), se**g**no (`og`), arbitrary-length wavy-line **t**rill with *tr* (`oT`), and arbitrary-length wavy-line trill without *tr* (`oTt`). All except the right parenthesis will appear above the staff; the right parenthesis appears to the right of the note head. 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. 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.

Slurs

To allow for overlapping slurs, there are two types, denoted by the symbols `s` and `t`, placed after the note. These are toggles, i.e., they start a slur if one of that type is not already turned on in the current voice, otherwise they terminate it. There is no distinction between ties and slurs. There are only two minor differences between `s` and `t` slurs. First, if one of each type starts or ends on the same note, then the `t` slur will be moved vertically away from the note a little bit to avoid a collision. So it's best to use `s` for most slurs, and save `t` for slurs that surround `s` slurs. Second, only `s` slurs are permitted in xtuplets.

The default direction or starting or ending level of a slur can be overridden by appending `u` or `d`, or `+` or `-` followed by an integer for the number of `\internote`s to raise or lower the position. **PMX** will automatically adjust the horizontal position of the start or end if necessary to avoid overlapping a stem, but there is currently no **PMX** command which lets the user control or override this. However, the macros `\isu` etc˙ defined in `pmx.tex` can be entered as in-line TEX commands (see below) if such horizontal offsets are needed. When this becomes necessary, as for example in chords with all notes tied, begin numbering the manually entered slurs at 0 to avoid any possible conflicts with **PMX**'s automatic slurs, which are numbered from 8 downward.

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 `]`. One may also 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 then optionally be followed immediately by several options: (1) `u` or `l` to override **PMX**'s selection of the direction of the beam; and (2) one, two, or three integers, each preceded with `+` or `-`. The first integer is an adjustment to the starting level (in `\internote`s) and may range from -15 to 15; 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 common 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`.

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.

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 3. 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`.

2.3 Commands That Affect All Parts

Commands that affect all the parts can only appear in the first (lowest) line of the first (lowest) voice. Most such commands will automatically be transferred from score to parts when separate parts are generated by **scor2prt** (see section 3, page 11).

Repeats and double 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. These symbols must be in the first voice, and must be followed in the same block by at least one note or rest, except at the very end of the piece. Unfortunately, this requirement can cause a little confusion for right repeats, since they have to appear at the start of the next bar, which is usually in the next input block.

If `Rlr` falls at the end of a system, **PMX** will automatically split it in two. The symbol `Rdl` may *only* come at the beginning of system and will always be split, leaving a double bar at the end of the prior system and a left repeat to start the new one.

Voltas (first and second endings)

Beginnings and ends of first and second endings are signaled by `V` (for *volta*). 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

A *meter change* symbol is a letter `m` followed by 4 numbers with no intervening spaces. The four numbers are `mtrnuml`, `mtrdenl`, `mtrnmp`, `mtrdnp` as defined above. Meter can only be changed at the beginning of a block. When changing meters, you must use `o` to represent the number 1; if you enter the digit `1` then **PMX** will append the following digit to it to make a number between 10 and 19 (inclusive).

Transposition

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 `\interlignes`); the second is the new key signature. When transposing, you should always use relative accidentals, signaled by the separate symbol `Ar` (see below). For example, to transpose a piece in C major or E major you would enter `Ar K+2+4` at the beginning of the first block.

Text

The symbols `h` or `l`, when placed alone in the first column of an input line, 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.

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 `\internotes`. This only works if the `Tt` is the final title block element entered.

Page numbering

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. This command 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 for parts can be still be initiated independently, for example with `%!P2` or `%1P2r` (see section 3, page 11).

Overriding defaults for accidentals, \afterruleskip

The symbol `A` controls the default sizes and operation of accidentals, and the size of `\afterruleskip` (the gap between a bar line and the first note). Its options are `b` to make all accidentals **b**ig, `s` to make all accidentals **s**mall, `r` for **r**elative accidentals (which should always be used if transposing), and `a` followed by a decimal number to override the default setting for `\afterruleskip` (which is 1`\elemskip`, 20 percent smaller than MusiXTEX's default). If no specification is entered for accidentals, the defaults will be absolute (not relative), and the size will be big except in places where a big accidental won't fit without adding extra space; in the latter case **PMX** will convert it to a small accidental and add extra space if necessary.

Minimum spacing between notes

PMX does some 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 11) to give the parts made by **scor2prt** different page sizes than the parent score.

Line and page breaks

It is possible to force line breaks or page 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).

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 d's on the middle line of 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 Literal TeX Strings (In-line TeX Commands)

PMX will accept three types of literal TeX strings, differing only in where they will appear in the `.tex` file. In the `.pmx` file, type 1 begins with `\` and will appear in the `.tex` file right before the TeX command for the next note in the `.pmx` file. *Each note or rest may have only one type-1 literal TeX string.* Type 2 begins with `\\` and will appear at the top of the `.tex` file, right before `\startmuflex`, regardless of where it appears in the `.pmx` file. Type 3 starts with `\\\` and will appear right before the start of the first `\notes` group in the current input block.

Literal TeXstrings may contain more than one TeXcommand, but must not exceed 80 characters in length, and may not wrap from one input line to the next. They end with `\` (backslash-space). This means that they may not contain the TeX macro `\` (backslash-space).

If you should want to enter a type-1 (note-based) string longer than 80 characters, you could use a series of type-2 or -3 strings to define a TeX macro containing the desired commands. Type-2 or -3 strings are also restricted to 80 characters, but you can enter several in a sequence on consecutive lines, and they will stay in the same sequence in the `.tex` file.

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) (perversity or absentmindedness is why they're different here than in note symbols). 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 prints 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.

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 `\noteskips`. The height and length of the line are set by the current note's level and `\noteskip`. 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 for some reason 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 conjunction with `%1` (see section 3, page 11) if you want to make a separate bass part with no figures.

2.6 Global TeX Modifications

PMX provides the option of entering an unlimited set of any legitimate TeX commands just before `\startmuflex`, and before any Type 2 literal TeX strings. Simply create a text file named [*basename*]`.mod` with the commands and place it 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 ability to handle literal TeX strings already described in section 2.4 on page 10.

3 Making Parts from a Score

This is accomplished by running **scor2prt** and entering the basename when prompted. **Scor2prt** will create `noinst` separate `.pmx` files, one for each instrument, with names [*basename*][*n*]`.pmx`, where [*n*] is the number of the voice.

In this section we describe how to control the appearance of the parts independently from 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. Lines beginning with `%`[n] will be transferred into the `.pmx` file for part n only. For example, to force a line break at line 15 and a page break at page 2 in part 3 only, enter `%3L15P2`.

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`.

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 `%!` or `%`[k] to affect all the parts or 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).

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).

One function of **scor2prt** is to condense consecutive bars of rest into a single symbol, a heavy horizontal bar with a number above it. The symbol `rm` defines such a **m**ulti-bar rest 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 judicious combinations of literal TEX commands and the special **PMX** commands listed in this section, 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.

4 Closing Notes

About the Example Files

`mostpmx.pmx` contains examples of most of the **PMX** commands. The printed output displays the **PMX** commands next to the resulting typeset characters. It is most useful if you look at the printed output rather than the file itself, 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; it could be hazardous.

`barsanti.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.

`jhrtsg.pmx` is a trio sonata by Johan Helmich Roman (1694-1758), "the Swedish Handel". This score demonstrates some tricks used to transition between the first and second movement (perhaps soon to be incorporated in **PMX**). It also has many examples of controlling the appearance of the parts with commands in the score.

`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 `ftp.gmd.de/music/musixtex/software/pmx`, or point a web browser to `ftp://ftp.gmd.de` and follow the links. **PMX** files on this server are supposed to propagate to the CTAN sites at `ftp.tex.ac.uk` and `ftp.dante.de`. The alleged American CTAN mirror at `ftp.shsu.edu` seems to be very unreliable. The following site may be a useful alternative: `ftp://ftp.cdrom.com/pub/tex/ctan/support/pmx/`.

As of the date of this writing, the Macintosh and UNIX ports of this version of **PMX** ore not yet available. The previous Macintosh port by Eric Petersen is available at `ftp.gmd.de` and at Eric's web site `http://www.aem.umn.edu/people/students/epeterse/PMX/`.

Stefan Evert has translated **PMX** into C and prepared a distribution which should be compatible with most UNIX systems. A beta version of **PMX** 1.1 is located at `ftp://ftp.mathematik.uni-stuttgart.de/pub/software/pmx/`.

Johan Tufvesson has posted binaries for Amiga, Sun Solaris, and Linux at `http://www.lysator.liu.se/ tuben/pmx/`

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, and to Eric Petersen, Johan Tufvesson, and Stephan Evert for beta-testing and for porting **PMX** to other platforms. Special thanks and credit go to Werner Icking for exhaustive beta testing, uncanny bug-finding, and continuing encouragement.