

Typesetting music with PMX

by

Cornelius C. Noack

noack@itp.uni-bremen.de

— Version 2.8 / March 2012
(PMX features up to version 2.618 included)

Acknowledgement

This tutorial owes its very existence to the work by *Luigi Cataldi*, who a few years ago produced a wonderful manual for **PMX** in Italian. Luigi's manual features many examples which help greatly in understanding some of the arguably arcane **PMX** notation.

Even though the Cataldi manual is, as Don Simons has aptly remarked, “written in the language of music”, it nevertheless seemed useful to have access to it for non-Italian speakers, so Don asked around for help on a ‘retranslation’.

In fact, that is what the present tutorial started out with: essentially a retranslation of the **PMX** part of Luigi's manual back into English, using, where that seemed feasible, Don's original **PMX** manual. I had been thinking for some time of producing some examples (and an index) for the updated (**PMX** 2.40) version of that manual, and now, taking Luigi's Italian version as a basis, it seemed an easy thing to do.

Of course, as such projects go: soon after the first version had appeared in 2002, it tended to get out of hand — Don Simons actively produced one new beta version of **PMX** after the other, and I simply could not keep up with his pace.

So alas: 5 long years went by before the first update of the tutorial – reflecting all **PMX** changes from Version 2.40 to Version 2.514, in one giant step! – had become possible.

But, incessantly, as the development of modern $\text{T}_{\text{E}}\text{X}$ and $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ has moved on in an accelerating pace, and so have $\text{MusiX}_{\text{T}_{\text{E}}\text{X}}$ and **PMX**. This tutorial was also bound to follow suite. So now the current status of **PMX** (2.618) and this tutorial are closely synchronized.

After thanking Luigi, in particular for his kind permission to use his version of the manual and its examples liberally, it behooves me to thank Don even more, not only for originating and continuously maintaining and improving **PMX**; his suggestions and continuous assistance in my work on this tutorial throughout the years were invaluable, as was his conscientious reading and improving, from the first draft to the present version. Many friends (too numerous to mention all of them) kept me busy trying to keep up with their productive suggestions and clarifications on the [\$\text{T}_{\text{E}}\text{X}\$ -music users' list](#) during the past years.

In memoriam Daniel Taupin

Contents

A Introduction	1
How to use this tutorial	1
A 1 \TeX , MusiX \TeX , PMX , M- \TeX	2
A 2 Installation	4
A 3 Authors of the Software	6
A 4 The Werner Icking Music Archive	7
B A PMX Tutorial	9
Conventions for this tutorial	9
B 1 Running PMX	10
B 1.1 Concatenating several files	13
B 2 Preliminary Concepts	14
B 3 Preamble	16
B 3.1 Numerical input	16
Body of the Input File	22
B 4 Commands for the Individual Staves	24
B 4.1 Notes	24
B 4.2 Dotted Notes	26
B 4.3 Stems	26
B 4.4 Other Note Parameters	26
B 4.5 Rests	28
B 4.6 Xtuplets	30
B 4.7 Chords	33
B 4.7.1 Arpeggio	34
B 4.8 Grace notes	35
B 4.9 Ornaments	37
B 4.10 Beams	39
B 4.11 Slurs and Ties	43
Pick a package	43
B 4.11.1 General slur usage	45
B 4.11.2 Invoking and using Type K slurs	47
B 4.11.3 Use of Type M Slurs and Ties	50
B 4.11.4 Special considerations for font-based slurs	50

B 4.12	Dynamic Marks	51
B 4.13	Clef Changes	52
B 4.14	Octavation	54
B 4.15	Figured bass (basso continuo)	54
B 5	Commands That Affect All Voices	56
B 5.1	Single bars, Double bars, Repeats etc.	57
B 5.2	Volta	58
B 5.3	Meter Changes	59
B 5.4	Key Changes	61
B 5.5	Transpositions	61
B 5.5.1	Transposition of an entire score	61
B 5.5.2	Transposition of Individual staves	61
B 5.6	Titles, and text above and below a system	62
B 5.7	Page numbering and page headers	64
B 5.7.1	Page numbering	64
B 5.7.2	page headers	64
B 5.8	Layout: line, page, and movement breaks	67
B 5.9	Bar Numbering	69
B 6	Some general options and technical adjustments	69
B 6.1	Global options	69
B 6.1.1	Accidentals	70
B 6.1.2	General layout	71
B 6.1.3	Layout details	71
B 6.1.4	Vertical spacing	72
B 6.1.5	PostScript type K slurs, ties and hairpins	73
B 6.2	Page Size	74
B 6.3	Stem direction of bass notes	75
B 6.4	Horizontal Spacing	75
B 7	Macros	76
B 8	Inline \TeX commands	77
B 8.1	Including \TeX Commands in the <code>.pmx</code> source file	78
B 8.2	Denoting pitch in inline \TeX	84
B 8.3	Putting \TeX Commands in an external file	84
C	Special Features	85
C 1	Making Parts from a Score	85
C 1.1	Usage	87
C 1.2	The <code>S</code> symbol	88
C 1.3	Other usage rules	88
C 2	Making MIDI Files	89
C 2.1	MIDI macros ¹	91
C 2.2	MIDI only accidentals	92
C 3	Lyrics	94
C 4	PMX and \LaTeX	95

D	Limitations, error messages, and bugs	97
D 1	Limitations	97
D 2	PMX's error messages	99
D 3	Bugs	99
D 3.1	A Benign Bug	99
E	Tricks of the Trade	100
E 1	Simple tricks	100
E 1.1	Special coding in <i>L'Incoronazione di Poppea</i>	100
E 1.2	Text after final system	100
E 1.3	Clef octaviation	101
E 2	More tricks	102
E 2.1	Changing vertical positioning of instrument name	102
E 2.2	Xtuplets ending with a rest	103
E 2.3	Shorthand notation for consecutive quavers	105
E 2.4	Varying the stave sizes	106
E 2.5	Stuff in front of the clefs of the first system	107
F	An Extension of PMX: M-Tx	112
G	Appendix: Examples	113
G 1	Dons Example Files	113
G 2	Full-score examples	113
G 2.1	Dufay, <i>Kyrie</i> (PMX code) :	114
G 2.2	Vivaldi, <i>Mundi Rector</i> (M-Tx code) :	117
G 2.3	Caccini, <i>Amor l'ali m'impenna</i> (M-Tx code)	121
	Index	126

List of Tables

B.1	An example of a batch file for running PMX	12
B.2	PMX source for excerpt of F.J. Haydn quartet	15
B.3	Meter options for <code>mtrdenp = 0</code>	18
B.4	Example of preamble parameters for the F.J. Haydn quartet in Fig. B.1, p. 14	21
B.5	Use of Note Parameters as shown in Fig. B.10	28
B.6	Ornaments	38
B.7	Parameters of the bar symbol R	57
B.8	Symbols beginning with an A (global options)	70
B.9	<code>inlinesample.tex</code> , as produced from <code>inlinesample.pmx</code>	82
C.1	Mnemonics for instruments acceptable in PMX	92
C.2	The General MIDI Instrument Specification	93
D.1	Numerical limits of PMX variables (soft limits)	98
D.2	Numerical limits of PMX variables (hard limits)	98

List of Figures

A.1	W.A. Mozart , <i>Sonata K545</i> , bars 1–2	3
B.1	F.J. Haydn , <i>quartet Op.76, no.2</i> , bars 1–4	14
B.2	C.C. Noack , <i>sonata diabolica, first movement</i> , bars 12–17	17
B.3	I. Stravinsky , <i>agon, first movement</i> , bars 1–2	19
B.4	Result of meter options for <code>mtrdenp = 0</code>	20
B.5	Examples of pickups in 4/4	20
B.6	Notation of the clefs	21
B.7	C. Debussy , <i>Pellas et Melisande</i> (excerpt)	23
B.8	PMX notation for pitch (second digit of note symbol)	25
B.9	Relative Octave Notations	25
B.10	Use of Note Parameters	29
B.11	Rests	30
B.12	Xtuplets	32
B.13	Xtuplets with Sicherman brackets	32
B.14	Chords	36
B.15	Grace Notes	37
B.16	Ornaments	37
B.17	Beams	40
B.18	A staff-jumping beam	42
B.19	An example of bar-crossing beams within a single staff	43
B.20	Three realizations of simple slurs	44
B.21	Three realizations of exotic slurs	44
B.22	E. Bloch , <i>Waves (Poems of the Sea I)</i> , bars 25,26	46
B.23	A dotted slur	47
B.24	Placing the slur ending with a staff-jumping beam	47
B.25	Shape variations in type K slurs	48
B.26	Vertical tweaks of slurs and ties	49
B.27	Dynamic marks	53
B.28	A clef change	53
B.29	A clef change in a staff with 2 voices	54
B.30	Figured bass in C. Monteverdi , <i>L’Incoronazione di Poppea</i> , aria “ <i>Pur ti miro</i> ”	56
B.31	Single bars, double bars, repeats	58
B.32	Volte	60

B.33	Some key changes	61
B.34	Some transpositions of a d minor scale	62
B.35	R. Wagner , <i>Tristan und Isolde</i> , beginning of third act	63
B.36	Titles, and text above and below a system	66
B.37	“ <i>Frère Jacques</i> ” (usage of macros)	77
B.38	A sample for the use of inline \TeX	81
C.1	F.Chr. Bach , <i>quartet B-Dur (p. 1)</i>	86
C.2	The “baroque default”, and overriding it with MIDI only accidentals	94
C.3	Further use of MIDI only accidentals	94
E.1	J.Chr. Bach , Quartet in B Major (beginning of cello voice)	110
E.2	A. Bruckner , <i>Locus iste</i>	111
G.1	G. Dufay , <i>Kyrie</i> (generated by PMX)	116
G.2	A. Vivaldi , <i>Mundi Rector</i> (generated by M-Tx/PMX)	120
G.3	G. Caccini , <i>Amor l’ali m’impenna</i> (generated by M-Tx/PMX)	129

Chapter A

Introduction

How to use this tutorial

The purpose and function of this tutorial is, actually, twofold: for the novice with **PMX**, it should be a readable guide to finding out what **PMX** does (and what it doesn't), how it is sensibly used, and what to do when you run into difficulties. For the seasoned user, on the other hand, it should serve as an extended **PMX** manual,

1. giving a reasonably accurate account of *all* the available features in an order which should make it not too tedious to find an answer to whatever question one might have,
2. indicating some of the more common uses of straight MusiX_{TEX} commands to do things not available in **PMX** directly.

As a consequence of this twofold purpose, the two types of readers should have two very different approaches:

The novice should, after reading Section **A 1**, turn right to Section **B 2** and **B 4.1**. After that you should get your fingers dirty as quickly as possible: keep your first own **PMX** file `my_opus.pmx` as simple as possible, and proudly produce your first print, following the instructions given in Section **B 1**. From thereon follow your interests!

The seasoned user will usually need the tutorial mainly to find out the exact usage of some commands he unforgiveably forgot about, retrieve some rarely used procedure, or find some clever gimmick that he never was aware of. Those people will first scan the whole tutorial quickly in a matter of minutes to see what's there and gloat about what's missing, and later on rely on the index – or look into authoritative source, the [PMX manual](#) by Don Simons.

Of course, once you have gotten acquainted well enough with **PMX**, all you'll ever need is Don Simons' "Quick Reference Table"¹. That summarizes the **PMX** symbols with all their

¹The Quick Reference Table is found in the software section of the **Werner Icking Music Archive** as [ref260.pdf](#).

options and always reflects the latest version. Errors found in the Quick Reference Table should be reported to Don Simons directly, whereas all complaints (outright errors, awkward formulations, missed subtleties) should be sent solely to the author.

This tutorial refers to **PMX** 2.618 , as of March 2012 (formally still a ‘beta’ version). For details on the differences with previous versions consult the [list of changes](#) of **PMX** in the software section of the WIMA (“**W**erner **I**cking **M**usic **A**rchive”) ².

A 1 \TeX , MusiX \TeX , PMX, M-Tx

MusiX \TeX is undoubtedly one of the best programs for typesetting musical scores: it produces ready-to-print output in PostScript and PDF format, it is stable, is continually updated, and is in the public domain and thus *totally free*.

Nevertheless its use seems to be limited, with few exceptions, to musicians coming from the world of science. MusiX \TeX does not ‘look and feel’ very intuitive, and it sometimes requires a familiarity with terms that may seem elementary to software programmers but are tough going for straight musicians. Furthermore, it is not WYSIWYG software (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et). This means that while writing the music you cannot immediately see the score as it will look when printed, because you do not write the music graphically as you do by hand, but rather in terms of a list of symbols that represent it. The system is not interactive; rather, the process of producing a musical score consists of several steps:

1. One writes the symbols in a special version of the \TeX language to a normal text file, using any text editor (such as **Emacs**, **Edit** or **Notepad**),
2. this text file is compiled with \TeX to produce a `.dvi` output file. This `.dvi` file is machine readable and usually can be previewed on the screen,
3. another program, e.g. `dvips`, produces a PostScript file from the `.dvi` file, which again can be previewed on the screen or sent directly to a suitable printer.

This symbolic and non-graphical nature of the program may discourage many people, but it is in many ways an advantage. Being fully aware of the symbolic character of the typesetting process while working on your input lets you concentrate much better on the final typographic output and keeps you from being distracted by the temptation to play games with interactivity.

MusiX \TeX is a collection of ‘macro’s that permits producing musical scores with the help of \TeX , the prestigious electronic typesetting program written by Donald Knuth, which is mainly used in scientific typesetting. In providing \TeX , Knuth not only put it in the public domain (to be freely distributed), but stipulated that any application that uses \TeX should also be in the public domain and freely available, without charge or royalty.

Before \TeX can be used, it has to be properly installed on the computer system at hand. Installing \TeX from scratch is not an easy task; consult Chapter [A 2](#) for the details.

²For details on the staff-wise transposition feature refer to [B 5.5.2](#).

Even after everything is properly installed and running, coding a musical score in the MusiX_{T_EX} language still remains a tedious process. Fortunately, there exist two preprocessors, **PMX** and **M-T_x**, which dramatically simplify the input process; in fact they provide what must be among the simplest possible systems of electronically encoding musical scores.

The first two bars of Mozart's *piano sonata KV 545* illuminate the difference in coding with either MusiX_{T_EX}, **PMX** or **M-T_x**. The codes given below for the three systems of software all produce the (identical) output, shown in figure A.1:



Figure A.1: W.A. Mozart, *Sonata K545*, bars 1–2

MusiX_{T_EX} :

```
\input musixtex
\parindent10mm
\setname1{Piano}
\setstaves12
\generalmeter{\meterfrac44}
\nobarnumbers
\startextract
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\ql l\sk\ql n\en
\bar
\Notes\ibu0f0\qb0{dgi}\qlp i\en
\notes\tbu0\qb0g|\ibb1j3\qb1j\tb11\qb1k\en
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
\endextract
\end
```

PMX :

```
2 1 4 4 4 4 0 0
1 1 20 0.12
Piano
tt
./
% Bars 1-2
c8 g+ e g c- g+ e g | d g f g c- g+ e g Rb /
c2+ e4 g | bd4- c1 d c2 /
```

M-T_x :

```

Style: piano
Piano: Voices MD MS; Clefs G G; Continuo
Name: Piano
Meter: 4/4

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

```

Not only is the MusiX_{TEX} much longer, it undoubtedly is less intuitive and more complicated than the other two.

In the present tutorial we shall describe in detail the usage of **PMX** and, in a very cursory way, that of **M-T_x**.

In the remaining part of this introduction, we treat questions of installation of the entire system of MusiX_{TEX} 1.15 and **PMX**. The whole installation procedure may seem a formidable task to non-experts. But don't despair: your efforts will be rewarded by the best and simplest system of musical typesetting available. And, to paraphrase a famous _{TEX} error message: "If in serious trouble, ask a wizard for help". Such wizards are literally at your fingertips; just sign up and post your problem to the mailing list at WIMA (the **W**erner **I**cking **M**usic **A**rchive), and someone is bound to be in the mood to answer.

A 2 Installation

Recently, there has been a number of major improvements in the basic MusiX_{TEX} 1.15 software (increasing, e.g., the maximum number of instruments allowed), and subsequently **PMX** has also undergone major improvements. As a consequence, **PMX** now is really tuned to MusiX_{TEX} 1.15: with **PMX**(2.6xx) you can make full use of most³ of the new features of MusiX_{TEX} 1.15.

The installation instructions given here refer explicitly to installing MusiX_{TEX} 1.15 and **PMX** under Windows XP; they assume that you have MiK_{TEX}⁴ installed (version 2.6 or higher; preferably the current version 2.9 ; cf. p. 5).

For Unix, good instructions can be found in the software section of WIMA: icking-music-archive.org/software/musixtex/musixtex-for-unix.html.

Installation of MusiX_{TEX} 115 and **PMX** 2.6xx

MusiX_{TEX}

1.15 You can download the basic distribution for MiK_{TEX} 115 from WIMA: icking-music-archive.org/software/musixtex/musixtex.zip.

³An important restriction is: in **PMX**(2.6xx) the maximum number of staves is 24 .

⁴Mik_{TEX} is a _{TEX} system for Microsoft Windows users.

A comprehensive – and very readable – installation instruction manual for MusiX_{TEX} 1.15 has recently been given by Andre van Ryckeghem and Don Simons:

<http://icking-music-archive.org/software/musixtex/mxinsuse.pdf>
on the WIMA software page, which you can download [here](#).

Even if you consider yourself an expert in installing high-level software, it is certainly wise to first read this manual carefully (it’s only 6 pages) before starting the installation this way.

The installation procedure referred to above has the advantage that it does not rely on the TDS-compliant data structure of the CTAN (‘Comprehensive TeX Archive Network’) distribution and, a fortiori, on the MikTeX data system being up-to-date with the development of MusiX_{TEX}. For many years, this was a serious obstacle against making comfortable use of CTAN, and thus of MikTeX as well.

This has changed drastically in the last few years. As of 2012, the MusiX_{TEX} software of WIMA is formally maintained by CTAN as well, and in consequence, MikTeX is now pledged to be updated with the CTAN data within a few months at most. In particular, with its current version (MikTeX 2.9) MikTeX now fully supports **PMX** 2.603.

So if you have MikTeX 2.9 installed, all you need to install MusiX_{TEX} 1.15 is to download

[musixtex/musixtex-texmf.zip](#)

and follow the (terse, but clear) [Installation instructions](#) by Bob Tennent.

To test the MusiX_{TEX} 1.15 installation, download [sample.zip](#) from WIMA to some temporary folder and unzip it, obtaining the 3 files `sample.tex`, `sample.pmx`, and `pmxrun.bat`. Then run

```
etex sample.tex
```

in a command line.

Make sure that when using basic TeX you *always* run `etex.exe` (‘extended TeX’) and *not* `tex.exe` !! Although MiKTeX and L^ATeX now use extended TeX by default, the 2 commands `tex.exe` and `etex.exe` still both exist and *differ*; **PMX** 2.6xx, will not run properly under ordinary `tex.exe`.

PMX

Compared to MusiX_{TEX} 1.15, installing **PMX** 2.6 is a cinch. All you need to do is

1. decide which particular version of the **PMX** 2.6 series you need: 2.603 is by now almost standard; the new version 2.617 has a very important new feature regarding transposing instruments, and 2.618 allows adjusting the size of the **PMX** output automatically to the paper size used⁵,

⁵remember that formally **PMX** is still ‘beta’.

2. download [pmx 2.618.zip](#) (or another 2.6xx) and unzip it,
3. copy `pmxab.exe` and `scor2prt.exe` to `...\localmixtex\miktex\bin`,
4. copy `pmx.tex` to `...\localmixtex\tex`,
5. run the MiKTeX FNDB ('File Name Data Base').

To test the installation, run

```
runpmx sample.pmx
```

in a command line.

A 3 Authors of the Software

MusiX_{TEX}: *Daniel Taupin*

[MusiX_{TEX}](#) evolved from Music_{TEX} by *Daniel Taupin*, a physicist in the Laboratoire de Physique des Solides in Orsay, *Ross Mitchell* (CSIRO Division of Atmospheric Research, Private Bag No.1, Mordialloc, Victoria 3195, Australia) and *Andreas Egler* (Ruhr–Uni–Bochum, Ursulastr. 32, D-44793 Bochum).

MusiX_{TEX} had been maintained actively and conscientiously by Daniel Taupin until his untimely death in 2001 — he died accidentally while climbing in the Alps. He is remembered with a [Memorial](#) on the web page of the Werner Icking Music Archive.

After Taupin's death, MusiX_{TEX} seemed quite stable and could thus, for all practical purposes, be considered 'frozen' for a long time.

However, nothing as lively as MusiX_{TEX} can really remain literally frozen forever. A number of improvements have been continuously added by a group of MusiX_{TEX}pers, and, finally, Hiroaki Morimoto has kindly provided a new version of MusiX_{TEX} which corrects known bugs. Then a new and *strongly enhanced version* of the MusiX_{TEX} manual was prepared by a group led by Don Simons, which has now been officially released: MusiX_{TEX} 1.15. It is located at the [software section](#) of the Werner Icking Music Archive (dubbed 'WIMA').

Finally, here is a list of the authors of essential parts of **PMX** and related software:

PMX: *Don Simons*

E-mail: dsimons@roadrunner.com

M-Tx: *Dirk Laurie*

E-mail: dlaurie@na-net.ornl.gov

Musixlyr and Musixser: *Rainer Dunker*

E-mail: rainer.dunker@web.de

PostScript slur package **K**: *Stanislav Kneifl*

E-mail: stanislav@kneifl.net

PostScript slur package **M**: *Hiroaki Morimoto*

E-mail: CQX05646@nifty.com

A 4 The Werner Icking Music Archive

Werner Icking (June 25, 1943 – February 8, 2001) was the founder of the *GMD Music Archive*, which contained downloadable scores, various MusiX_{TEX} programs, a discussion list, and various other musical resources in the public domain. His contributions to the development and promulgation of this software were peerless. His editions of music (in particular the edition of the complete ‘Urtext’ of J.S. Bach’s pieces for violin and violoncello solo) are examples of sophisticated technique, artistic sensibility and philological meticulousness. Everybody, whether expert programmer or complete novice, would always find valid and unselfish advice from him, which invariably ended with the motto “*hope this hilft*”. In time a small, but regular group of faithfuls gathered around this site, from all over the world, who never had met Werner in person, but nevertheless came to see a friend in him, mainly through frequent email exchanges.

Werner died suddenly on February 8, 2001, while bicycling home from work, as was his regular habit. That very day he had performed maintenance work for the site and replied to messages on its discussion list, of which he had been the main administrator. The community of users of the archive decided not to disband, but to continue Werner’s work. Starting with the contents of the GMD Archive, a new site was created, and dedicated to Werner’s memory. Its official name is the *Werner Icking Music Archive*.

After some difficult times, it is now hosted by **PALDAM IT** ; its URL is

<http://icking-music-archive.org>.

Its chief administrator is Christian Mondrup, who more than any other provided reasoned guidance and continuity during the difficult transition.

The archive houses four main categories of information:

Archive of sheet music : Contains hundreds of royalty-free scores in PDF, many with their source code in MusiX_{TEX}, **PMX**, or **M-Tx**. It is maintained by *Christian Mondrup*, who is also the main administrator of the Werner Icking Music Archive

E-mail: reccmo@daimi.au.dk .

MusiX_{TEX} and related software : Contains MusiX_{TEX} and related software, as well as user manuals and various add-ons. It is maintained by *Don Simons*

E-mail: dsimons@roadrunner.com .

T_EX-music list : This discussion list is invaluable to novices, advanced users and developers. Themes focus on technical problems of typesetting using MusiX_{TEX} and related software, but often drift into other musical subjects of historic or esthetic interest. It is not moderated, but you need to subscribe in order to participate. For information on how to

subscribe, visit <http://icking-music-archive.org/mailman/listinfo/tex-music> .
Its administrator is *Maurizio Codogno* (E-mail qve-al59@myamail.com).

Links to other sources of information regarding music: There are links to search facilities about composers/works/editions, a multi-lingual glossary of musical terms, and some lists of composers' works. This is maintained by *Jean-Pierre Coulon*:

E-mail: coulon@obs-nice.fr

As of 2012, (MusiX_{TEX} / **PMX**) is coordinated with **CTAN**, thus ensuring that **CTAN** is up to date with (MusiX_{TEX}).

This coordination with **CTAN** is maintained by *Bob Tennent*.

E-mail: rdt@cs.queensu.ca

Chapter B

A PMX Tutorial

Conventions for this tutorial

“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.”[†]

Typographical conventions:

- The `typewriter` typeface always indicates verbatim text *exactly as you would input it to the computer*. This includes file names, MusiX_{TEX} tokens, and **PMX** commands, e.g., `barsant.pmx`, `\internote`, `c44`.
- **bold** typeface is used for two purposes: first, for program names (e.g. **pmxab**), and second, when applied to a single letter within a normal word, to emphasize the mnemonics of a **PMX** command (e.g. **lr** signifies a “left-right repeat”).
- When used in commands, *italics* represent input variables for which the user would substitute the appropriate actual value. To make this notation quite clear – and distinct from the general use of italics for emphasis –, the variable will be surrounded by square brackets in such cases (e.g. `L[n]P[m]M`), but *the brackets are not to be included with the verbatim text*.

Musical terms:

The language of music is Italian, and so the radical solution of terminological problems would be to use the Italian names throughout. This would, however, defeat the whole purpose of this tutorial, so some compromises need to be made, and some terms, which may be used differently by different people, need to be clearly defined. So in this tutorial

[†]Quoted literally from the original introduction to **PMX** by its author, Don Simons.

- a *staff* (plural *staves*) means the set of 5 horizontal lines in which music is usually notated, aptly called a *pentagramma* in Italian¹.
- a *system* means a coherent set of staves to be played simultaneously.
- a *voice* means a line of music that is a musical entity of its own, such as a melody. In polyphonic music, there is often more than one voice to a staff, e.g. a 4-voice Bach choral is often written with the 2 top voices (soprano, alto) in one staff, the bottom ones (tenor, basso) in the other².

technical terms:

- a *system* means a coherent set of staves to be played simultaneously.
- a *block* is the series of *PMX* commands WITHIN ONE SYSTEM, formally ended by a / (slash), which then must be the LAST CHARACTER on this input line³.

B 1 Running *PMX*

As stated in the introduction (cf. Section A), *PMX* is a *preprocessor* to *MusiX_{TEX}*. That means that with *PMX* the sequence of steps from the input to the printed paper is one step longer:

0. The user writes *the symbols of the *PMX* language* to a normal text file, using any text editor. The name of this file, (say, `my_opus.pmx`) *must have the suffix pmx*,
1. the file `my_opus.pmx` is run through the *PMX* processor, `pmxab`. This produces the output file `my_opus.tex`,

Note that in *PMX* (like in Unix, but unlike Windows) spaces within a file name are *not allowed*! If in the above example, the input file name were `my opus.pmx` instead of `my_opus.pmx`, an unpredictable error, with confusing error messages, would occur.

2. the file `my_opus.tex` is then processed with *TEX* to produce a `.dvi` output file⁴. This `.dvi` file is machine readable and (usually) can be previewed on the screen,

¹Note that one-line staves, used for percussion instruments (drums, triangles etc.), are presently not implemented in *PMX*.

²*PMX* allows 1 or 2 voices in one staff.

³This often precludes even *TEX* comments (started with a %) on the same line, after the slash!

PMX does not check for compliance with this rule, but nevertheless it is good practice to adhere to it. Otherwise you are liable to encounter weird errors which may be difficult to trace.

⁴This step is a bit more complicated than it sounds because it actually is a 3 pass system: it consists of (i) running *TEX*, (ii) running `musixflx`, and (iii) running *TEX* again (details on the rationale behind this are described in the *MusiX_{TEX}* manual by Daniel Taupin).

3. another program, e.g. `dvips`, produces a PostScript file from the `.dvi` file, which again can be previewed on the screen, or sent directly to a suitable printer.

Usually this whole process is automated to some degree by the use of a so-called ‘script’ or ‘batch’ file, or with the help of some more elaborate system of interconnecting software (most of these tuned to the needs of the general T_EX community).

There are too many variants of this kind of supporting software – public domain or commercial – to discuss here, and personal preferences or dispositions are too varied. If you are at a loss on what to use, send a message to the T_EX-music list (cf. p.7), and you can trust that you get all the individual help you need.

To illustrate the above process, however, an example for a typical ‘batch’ file, for a standard **Windows** system, is given in table (B.1). This assumes that you have installed

1. a “MiKTeX system,
2. a MusiX_TE_X system,
3. a `dvips` system,

all installed in the standard way, and that you have produced the **PMX** input file `my_opus.pmx`, using your favorite ASCII text editor.

You would then run

```
runpmx my_opus
```

on a command line, and would see a Ghostview window opening, from which you could print your opus (or do with it whatever you like).

You perhaps will want to write your own batch program⁵, to suit your personal requirements. Some further remarks on how `pmxab` operates may prove helpful for that.

First, whenever `pmxab` terminates due to a syntax error, the exit code is set 1 (0 when there are no errors). There are various ways of detecting this with batch commands, and 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 `[filename].pml`.

⁵The `runpmx.bat` given in Table B.1 is identical with the `runpmx.bat` used in the installation guide (Section A 2).

When adapting this to your needs or writing your own batch file altogether, you should make sure that you use `etex.exe` in the 3 calls of T_EX. MusiX_TE_X-T115 and **PMX** `texttt2.6xx` both require eT_EX (extended T_EX)!

```

:~::~: runpmx.bat ~::~:
Call C:\Programme\MikTeX_2.9\localmiktex\miktex\bin\pmxab %1.pmx
if errorlevel 1 goto pmxerr
if exist %1.pml del %1.pml
if exist %1.mx2 del %1.mx2
if exist %1.mx1 del %1.mx1
if exist pmxaerr.dat del pmxaerr.dat
::
Call C:\Programme\MikTeX_2.9\miktex\bin\etex.exe %1.tex
if errorlevel 1 goto texerr
::
Call C:\Programme\MikTeX_2.9\localmiktex\miktex\bin\musixflx.exe %1.mx1
::
Call C:\Programme\MikTeX_2.9\miktex\bin\etex.exe %1.tex
if errorlevel 1 goto texerr
if exist %1.log del %1.log
::
Call C:\Programme\MikTeX_2.9\miktex\bin\dvips.exe %1.dvi
if errorlevel 1 goto dvierr
if exist %1.log del %1.log
::
Call C:\Programme\GhostScript\Ghostgum\gsview\gsview32.exe %1.ps
::
goto :end
:~::~:
:: error exits :
::
:pmxerr
echo.
echo          PMX has found errors when processing %1.pmx !
echo.
pause
goto :end
::
:texerr
echo.
echo          TeX has found errors when processing %1.tex !
echo          check %1.log for details.
echo.
pause
goto :end
::
:dvierr
echo.
echo          *** dvips had errors ! ***
echo.
pause

::
:end
:~::~: end of runpmx.bat ~::~:

```

Table B.1: An example of a batch file for running **PMX**

B 1.1 Concatenating several files

Sometimes one wants to split the work on a score into several smaller parts, not only when otherwise you would exceed some **PMX** numerical limit (cf. Section D 1), but simply for practical reasons. Logically, one should distinguish two separate cases:

1. Within *one and the same PMX program*, there might be parts that recur identically in several regions of the full code, and one would not want to retype these over and over again (both for efficiency and for clarity reasons). In such cases, you would want to have a command similar to the `\input` of \TeX , \LaTeX and many other programming languages.

In **PMX**, such an ‘`\input`’ command has the form `AR[file name]`, where *file name* is the name of the (ASCII) file you want to include, and `AR[file name]` will insert this file at exactly that point in your **PMX** program where the `AR[file name]` is.

A possible use for this procedure, i.e. using the `AR` command, could be to save typing when you want to include your favorite option defaults that in fact you use in almost every piece of music you write with **PMX**.

But a simple “cut & paste” is arguably a quite practical and more straightforward way of achieving this goal; moreover, if you want to be fancy about it, **PMX** has a much more sophisticated tool to offer for that purpose⁶, viz. a macro (cf. Sec. B 7).

2. Sometimes may have reason to make several completely independent **PMX** files, producing their output separately. In the end, you would want to concatenate these pieces again. This can be easily done:

Suppose you have coded the three movements of a piano sonata in three separate **PMX** files: `son1.pmx`, `son2.pmx` and `son3.pmx`. To create one single file from these, proceed as follows:

- (a) Process the three files `.pmx` separately, obtaining the three files `son1.tex`, `son2.tex` and `son3.tex`,
- (b) Create a \TeX file `son.tex` consisting of the following lines:

```

\input musixtex
\startmuflex
\let\startmuflex\empty
\let\endmuflexsav\endmuflex
\let\endmuflex\endinput
\input son1
\input son2
\input son3
\endmuflexsav

```

⁶In fact, there is yet another fancy tool, i.e. including a ‘global’ file with the specific name `pmx.mod`, useful for special purposes. For details on this, refer to the **PMX** manual.

\bye

- (c) Process the file `son.tex`, in the usual three passes ($\text{T}_{\text{E}}\text{X} \Rightarrow \text{musixflx} \Rightarrow \text{T}_{\text{E}}\text{X}$). The resulting file `son.dvi` then contains the full score of the entire sonata.

Alternatively, you can concatenate the files using \LaTeX , or more specifically, the `mtxlatex` package. For details on this see Section C 4.

String Quartet op. 76, No.2

F.J. Haydn

Allegro

Figure B.1: F.J. Haydn, *quartet Op.76, no.2*, bars 1–4

B 2 Preliminary Concepts

The **PMX** code of a musical score consists of two parts, the *preamble* and the *body*. Lines with a `%` in column 1 are comment lines (as in $\text{T}_{\text{E}}\text{X}$); they are disregarded by the program⁷.

The *preamble* contains the general specifications for a score, some musical (e.g., number of instruments, meter, key signature), and some typographical (e.g., number of pages, number of staves per page).

The coding for the music itself is given in the *body*, which in turn usually has a *header*. As an example, consider the first few bars of Haydn's *quartet Op.76, no.2*. The music shown in Fig. B.1 is obtained from the **PMX** source code given in Table 15.

⁷Note, however, the additional conventions of the `scor2prt` program (see Chapter C 1, pp. 85).

```

F.J.\ Haydn, Quartet op.76, no.2, bars 1--4
PREAMBLE:
nstaves ninstr mtrnuml mtrdenl mtrnump mtrdenp
      4      4      4      4      0      6
npickup nkeys
      0      -1
npages nsystems musicsize fracindent
      1      1      16 .08
Violoncello
Viola
Violin II
Violin I
batt
./
% BODY:
% HEADER:
Tc
F.J. Haydn
Tt
String Quartet op. 76, No.2
h
Allegro
Abep
w170m
% begin of music input -
% bar 1
d82 Df o. d+ o. d o. d o. r d o. d o. d o. /
f83 Df o. f o. f o. f o. r f o. f o. f o. /
r8 a83 Df o. a o. a o. r a o. a o. a o. /
a24 Df d- /

% bar 2
r8 d o. d o. d o. r e o. e o. e o. /
r8 g o. g o. g o. r g o. g o. g o. /
r8 b o. b o. b o. r cs o. c o. c o. /
e2 a- /

% bars 3-4
f8 s e f cs s d4 a | b2 s a4 s o. r Rb /
a2 t a4 t .cs- | d8 s e f d s e4 o. r /
d8 s cs d e s e s d c s a1+ s g s | f s g f e s d8 o. d o. cs4 r /
d8 s cs d e s g s f e s a | d4- zd+ cn1 s b a g s s a4 o. r /
% end of PMX source file

```

Table B.2: **PMX** source for excerpt of F.J. Haydn quartet

B 3 Preamble

The *preamble* consists of one or several lines of numbers, followed by lines with typographical data relevant for the whole score. The preamble ends with a line giving the name of the directory to which **pmxab** is to write its output `.tex` file.

B 3.1 Numerical input

The first line(s) of the preamble must contain 12 numerical parameters, separated by one or more spaces ('white space'). In **PMX**, as in **T_EX**, a line feed is equivalent to a white space, so the following 3 forms of the beginning numerical input are all equivalent:

```
4 4 4 4 0 6 0 -1
1 1 16 .08
```

— or —

```
4 4 4 4 0 6 0 -1 1 1 16 .08
```

— or —

```
4
4
4
4
0
6
0
-1
1
1
16
.08
```

For the purpose of this tutorial, the 12 s are named⁸.

```
nstaves, ninstr,
mtrnuml, mtrdenl,
mtrnump, mtrdenp,
npickup,
nkeys,
npages, nsystems,
musicsize, fracindent
```

Their significance is as follows:

⁸Their names are written in typewriter typeface here because in a way they *are* input and, as some of the examples show, used as such, albeit only in comment lines. But these names themselves never occur in **PMX** commands. (See, however, the footnote on p.99).

- `nstaves`, an integer ≤ 24 , is the total number of staves per system. Each staff may contain either one or two voices (lines of music). The number of voices in a staff may change as the piece progresses, but the total number of voices at any one time cannot exceed 24. So if, e.g., there are 24 staves, there can only be one voice per staff.

`ninstr`, an integer $\leq \text{nstaves}$, is the number of *instruments*. Each instrument has a unique name (cf. p. 21), and any instrument with more than one staff will have its staves joined with a curly bracket. Often there is only one staff per instrument, and `ninstr` = `nstaves` in that case. There are two ways to assign more than one staff to one or more instruments:

- if only the first, i.e. the *lowest*, instrument has more than one staff, such as in a score for piano and a one-staff solo instrument, simply make `ninstr` < `nstaves`, and any difference will show up in instrument 1, the bottom one in each system,
- When you need a more general assignment of staves to instruments, put a minus sign in front of `ninstr`, and follow `ninstr` with the number of staves in each instrument in succession, in sequence from the bottom one up (the same order as the instrument names, cf. p.21), separated by spaces. These numbers *must* add up to `nstaves`: in a way, they are simply a partition of the numerical parameter `ninstr` .

Example : The preamble of the first example (Fig. B.2) simply has `nstaves` = 3 and `noinst` = 2 : while the Stravinsky excerpt of Fig. B.3 was obtained⁹ with `nstaves` = 9

Allegro

Figure B.2: C.C. Noack, *sonata diabolica*, first movement, bars 12–17

and `noinst` replaced by

–7 1 1 1 1 2 2 1 .

This tells **PMX** that there are 7 instruments in all, with 1 staff each for the four strings, 2 each for the piano and the harp, and 1 for the trombone. Note that these numbers – 8 in all in this case – count as *one parameter*, i.e. the 12 numerical parameters of the preamble are made up of 19 numbers in this case:

⁹In the **PMX** text with which Fig. B.3 was produced, an additional M_usiX_TE_X command was used to gather the strings in a group in the usual way. This “inline use” of T_EX commands will be explained in detail in Section B 8 .

```

9           : (nstaves)
-7 1 1 1 1 2 2 1 : (ninst)
4 8         : (mtrnuml, mtrdenl)
4 8         : (mtrnump, mtrdenp)
2 0        : (npickup, nkeys)
0 2      16 0.14 : (npages, nsystems, musicsize, fracindent)

```

- The following 4 numbers serve to define the meter of the piece. The first pair, `mtrnuml` and `mtrdenl`, are the *logical* values which **PMX** uses to calculate the length of a bar: `mtrnuml` is the logical numerator of the meter, i.e. the number of beats per measure, `mtrdenl` the denominator.

The second pair, `mtrnump` and `mtrdenp`, determines the appearance of the meter in the printed output, but has no effect on the internal timing analysis of **PMX**:

- if `mtrnump` > 0, then it and `mtrdenp` are printed literally as the numerator and denominator of the time signature. For example, 4 4 4 4 prints a standard 4 quarters meter,
- if `mtrnump` < 0, then the numerator actually used by **PMX** will still be the positive value of `mtrnump`, but the entire time signature will be printed with a vertical slash through it,
- if `mtrnump`=0, then `mtrdenp` determines the printed meter as given in Table B.3 and shown in Fig. B.4.

There are special rules for n/16 and n/1 time signatures (where the latter "1" normally means a semibreve or whole note). To get n/1 time, use `mtrdenl` = 0 (zero) and `mtrdenp` = 1¹⁰. For n/16 time, it is `mtrdenl` = 1 and `mtrdenp` = 16. So the choice [3 0 3 1] for the 4 meter parameters, for example, will give a 3/1 time, both in the **PMX** internal computations and in the printed output, while a 5/16 time is indicated by [5 1 5 16].

0	no meter is printed ('blind' meter)
1, 2, 3 or 4	a single digit, between the 2nd and 4th lines of each staff
5	cut time (alla breve)
6	common time
7	numeral 3 with a vertical slash

Table B.3: Meter options for `mtrdenp` = 0

¹⁰To remember this rule, recall that the printed denominator is taken literally, while the logical denominator represents the same time value that is denoted when entering ordinary notes (see section B 4.1 below): there 0 stands for a whole note. The special rule for n/16 time is due to programming convenience.

The image displays a musical score for the first two bars of the first movement of Stravinsky's *Agon*. The score is arranged in a system with seven staves, each labeled with an instrument: Trombone I, Harp, Piano, Violins I and II, Violas, Violoncelli, and Basses. The key signature is one sharp (F#) and the time signature is 4/8. The Trombone I part begins with a forte (*f*) dynamic and a triplet of eighth notes in the second bar. The Harp and Piano parts play a rhythmic pattern of eighth notes, also starting with a forte (*f*) dynamic. The Violins I and II, Violas, Violoncelli, and Basses parts play a similar rhythmic pattern, marked with a forte (*f*) dynamic and *pizz.* (pizzicato). The score is presented in a clean, black-and-white format with standard musical notation.

Figure B.3: I. Stravinsky, *agon*, first movement, bars 1–2

Figure B.4: Result of meter options for `mtrdenp = 0`

- The 7th parameter, `npickup`, is the number of beats in a pickup bar if one is present. If there is no pickup bar, set `npickup = 0`. `npickup` need not be an integer. Cf. Fig. B.5 for examples of pickup bars.

A pickup bar is the *only* bar that can have a different number of beats than the current value of `mtrnum1`. It must be followed with the first regular bar *in the same block*, i.e. the pickup note(s) *must not be followed* by a `|`, `/` or `//`.

We shall later describe a **PMX** command to change the meter, and explain how to use it for pickups to later sections or movements (cf. section B 5.3).

Figure B.5: Examples of pickups in 4/4 (the values of `npickup` are given under the staff).

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

The last four numerical parameters concern the layout:

- If `npages > 0`, it is the number of pages, and `nsystems` is then 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 a given number of pages, one or more staves may spill over onto an extra page, but you will not see this until you preview the `.dvi` file. Possible remedies are to increase `npages`, decrease `nsystems`, or use the `Ae` command, to be described later (cf. section B 6.1).
- If `npages = 0`, `nsystems` is interpreted as the average number of bars 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 the height of a staff, in points. The only values allowed are: 16, 20, 24 or 29.

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

Note that although logically `fracindent` could have any value between 0 and 1, you should never use a value seriously over .5. It is not only esthetically rather unpleasing, but may cause some unexpected havoc for MusiX_{TEX}.

<code>nstaves</code>	:	4	(4 staves)
<code>ninstr</code>	:	4	(4 instruments)
<code>mtrnuml</code>	:	4	(4 beats ...
<code>mtrdenl</code>	:	4	... of quarters)
<code>mtrnump</code>	:	0	(option for ...
<code>nmtrdenp</code>	:	6	... common time)
<code>npickup</code>	:	0	(no pickup bar)
<code>nkeys</code>	:	-1	(key: d minor)
<code>npages</code>	:	1	(1 page)
<code>nsystems</code>	:	1	(1 system)
<code>musicsize</code>	:	16	(16 pt staff)
<code>fracindent</code>	:	.085	(indentation of 1st system)

Table B.4: Example of preamble parameters for the F.J. Haydn quartet in Fig. B.1, p. 14

In the next part of the preamble the names of the *ninstr* instruments are given, as you want them to appear within the indentation in the first system, one per line, *starting with the bottom* instrument. If you don't want instrument names to appear, you must still leave `ninstr` blank lines here.

Next, on a new line, comes a single string of *nstaves* letters or numbers for the clefs, again starting with the bottom staff. The choices are¹¹: `b`, `r`, `n`, `a`, `m`, `s`, `t`, `f`, which stand for **b**ass, **b**aritone, **t**enor, **a**lto, **m**ezzosoprano, **s**oprano, **t**reble, or **f**rench violin clef, respectively. Instead of these mnemonic abbreviations, you can use the digits 0–7, according to the notation 0 (treble), 1 (soprano), 2 (mezzosoprano), 3 (alto), 4 (tenor), 5 (bariton), 6 (bass) and 7 (french violin)¹². These two clef codes are summarized in Fig. B.6.



Figure B.6: Notation of the clefs

¹¹There are several more exotic clefs, available only in straight MusiX_{TEX} (cf. Section B 8 on “Inline \TeX ”, and the MusiX_{TEX} 1.15 manual, Section 2.21).

¹²This rather unmnemonic notation has been retained in **PMX** for compatibility with MusiX_{TEX}.

The preamble ends with a line that contains the path name of the directory to which you want the files (the `.tex` file, in particular) to be written when **pmxab** processes your source text. Usually you will want that to be the current directory: in UNIX this is denoted by `./`, in DOS (most versions, including Windows) it is `.\`. In any case, the path – and thus the whole preamble – must terminate with `/` or `\`.

The rest of the **PMX** input file is called the *body*. It usually begins with a number of global options, each on a line by itself, called the *header* (which can be empty).

Following the header, the input of the actual music begins. The basic unit is called a *block*, each one consisting of from 1 to 15 complete bars. The input data for all bars in a block are entered for each staff in turn, starting with the first (i.e. the *lowest*) staff. The sequence of staves (and thus of the instruments) corresponds to that in the preamble: the first is the bottom one of the system as it appears in the final score, and the last is the top one. The input for each staff ends with a `/` (slash). The data for one staff need not be contained on a single input line, but may spread over as many as needed, with any number of blank lines and comment lines between them¹³, but after the end of one staff (i.e. after the slash), you *must* start the coding for the next staff on a new input line. A *block* ends with the slash of the last staff of the last instrument (the top staff in the score).

Though you can put up to 15 bars in one block, many users of **PMX** have the habit of systematically writing one bar per block only. Sometimes, however, for reasons of context, **PMX** *requires* a set of bars to be in one and the same block (cf. Section B 4.12). If you do put more than one bar in a block, it is advisable (although not required) to separate the bars with a `|` (‘vertical stroke’). The main function of this is to provide visual separation in the input file, and to help isolate input errors: if you put a `|` anywhere except at a bar end, the **PMX** processor will stop and show you where it detected the timing error. Otherwise (with a few minor exceptions¹⁴) `|` has no effect.

It is also good practice to separate the blocks with comment lines that state which bars are represented, as has been done in the input source for Fig. B.1 (p.14).

If there is a pickup bar (`npickup > 0`), it must be in a block with the first full bar¹⁵.

When there are 2 voices in a staff (e.g. for an organ or other polyphonic instrument, or in a choir or symphonic score), these are entered consecutively in the coding for that staff, again starting with the lower voice, and this voice is ended with a `//` (‘double slash’). Here again, after the double slash, you *must* start the second voice on a separate input line. The coding for this staff is then again concluded by a single slash.

The two bars from *Pellas et Melisande* by Debussy in Fig. B.7 are an example. The bottom staff, that for the trumpet, has only one voice, while the top staff (for the flutes) has 2, thus giving a total of 3 voices.

¹³Remember that in **PMX**, as in **TeX**, a line feed is equivalent to a space, and a sequence of many spaces is equivalent to one space.

¹⁴For such an exception cf. the note on p.57.

¹⁵If, for some reason, you do need to put the pickup in a separate block, set the initial logical meter to fit the pickup bar, then after the pickup bar do a blind meter change, as described in section B 5.3.

Lento

```

%-----%
% C. Debussy, Pelléas et Melisande, p.149
%-----%
%
2      2
12    8      12    8
0     6
1     1      16    .07
%
Trumpet
Flutes
tt
./
It60itrfl
h-4
Lento
Abep1
\\nobarnumbers\
%
% %%%%%%%%%%% Bar 1 %%%%%%%%%%%
cd4 D"con sord."+17 sf dd fd dd sf /
r8+0 fr sfu+0+1 e sf+0 t ed4 t r8+0
br+ sfu+0+1 a sf+0 r+0 gr sfu+0+1 f sf+0 //
rb8 g4+ t gd t rb8 c4 rb8 a4 /

% %%%%%%%%%%% Bar 2 %%%%%%%%%%%
cd4 sf dd fd ad sf Rb /
r8+0 fr sfu+0+1 e sf+0 t ed4 t r8+0 br+ sfu+0+1 a sf+0 t ad4 t //
rb8 g4 t gd t rb8 c4 t cd t /

% %%%%%%%%%%% end of file %%%%%%%%%%%

```

Figure B.7: C. Debussy, *Pelléas et Melisande* (excerpt)

The number of voices in a staff (1 or 2) is determined solely by whether the first sequence of symbols for that staff ends with / or //. Therefore the number of voices in any given staff can vary from block to block, but not within a block. In other words, if you need to change the number of voices in a staff, you must start a new block.

Finally, a general characteristic of **PMX** coding should be noted: all data comprises sequences of *symbols*, each one containing one or more adjacent characters. These symbols – including the voice-terminating symbols / and // – are *always* separated from each other by at least one space or line feed. Whenever there are several characters strung together without spaces, they are considered *one single PMX* symbol. This concept has already been illustrated in prior examples. The construction of the symbols will be explained in much more detail in the remainder of this chapter.

B 4 Commands for the Individual Staves

B 4.1 Notes

The most important input item is of course a single note. Its two main features are its *pitch* and its *duration*.

The *pitch* is primarily – i.e. up to octavation – indicated by the note *name* in *lower case* letters: **c**, **d**, **e**, **f**, **g**, **a**, **b** (do, re, mi, fa, sol, la, si). If the note is to have an accidental, the note name is followed *without a space* by **s**, **f** or **n** for a sharp, flat or **natural**; **ss**/**ff** for a double sharp/double flat.

The *basic duration*, i.e. the duration exclusive of a possible dot, is indicated by the first unsigned digit following the note name, again with no space: **9**, **0**, **2**, **4**, **8**, **1**, **3**, **6** respectively for double-whole (breve), whole (semibreve), half, quarter, eighth (quaver), sixteenth (semiquaver), thirty-second, and sixty-fourth notes. For a consecutive sequence of notes of equal duration, this needs to be given explicitly only for the first note of the sequence. Subsequent, consecutive notes with no explicit duration will *inherit* their basic durations from the most recent note with an explicit duration.

A *dotted/doubly dotted note* is indicated by the letter **d**/**dd** just about anywhere¹⁶ in the note symbol, after the note name, of course.

It is important to note that, in contrast to the basic duration, a dot is never inherited by a subsequent note.

The *pitch* of a note is, of course, not determined by the note name alone; the actual octave needs to be specified as well. For this, there are three possibilities:

explicit octaves:

A second unsigned digit¹⁷ indicates the octave to which the note belongs. For reference,

¹⁶The only restriction on the position of **d** within the note symbol occurs if the dot is to be shifted (cf. p.27).

¹⁷In contrast to the letters for accidentals and dots and other letters to be explained in Section B 4.4, the 2 digits for duration and pitch, if present, must always be given in this order. Specifically, if by omitting the duration digit you use the inherited duration, you can no longer specify the octave explicitly by an unsigned digit.

octave 4 runs from middle C to the B above. The lowest note on an 88-key piano is the A in octave 0, while its highest note is the C in octave 8 (cf. also Fig. B.8).



Figure B.8: **PMX** notation for pitch (second digit of note symbol)

inherited pitch:

For a note without any explicit octave or relative pitch indication, the octave is assigned by **PMX** such that the note is placed in the octave that makes it nearest to the most recent note *in the same voice*, i.e. the pitch is ‘inherited’. Thus for jumps of less than a fourth up or down, you only need to enter the note name to fully specify the pitch. This feature often lets you go for long stretches in a voice before needing to enter the octave. For jumps of a fifth or more, you need to specify the octave either explicitly as described above, or relatively as described below.

relative pitch:

A + or - (if not part of one of the signed numerical suboptions to be discussed later) indicates that a note is to be an octave higher or lower than it would otherwise be. Two consecutive +’s will raise the pitch two octaves, and so forth¹⁸.

As an example of all this, compare Fig. B.9 with the **PMX** notations given under the staff¹⁹.



Figure B.9: Relative Octave Notations

The first note of each line of music in a block must contain at a minimum the note name and a basic duration value. It is good practice and can simplify editing if in addition an explicit octave is set there. However if it is not, **PMX** will make some assumptions: at the start of the first input block the pitch will be set as if there were a prior note of middle C. In later blocks **PMX** will use the obvious inheritance rules from the end of the prior block²⁰.

¹⁸Two consecutive +’s can be abbreviated by one ++ .

¹⁹The same result would be obtained with the full, absolute notation c84 d84 e84 c85 b84 a84 g84 g83 | c26 g23 | c04; but this is obviously much more verbose.

Explicit octave numbers can be combined with one or more + or -, although this is not recommended.

²⁰However, if the number of voices in a staff has changed from the prior block, it is safest to reset the octave

B 4.2 Dotted Notes

Inheritance of dotted notes is a little tricky. As was noted above, the dot itself is never inherited; you always have to use a `d` in the note symbol, even if the actual duration and octave are the same as the prior note. But the *basic* digit of duration need not be reentered if it hasn't changed²¹. 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 digit of duration was for a half note all along.

B 4.3 Stems

PMX usually determines the correct stem length and direction (*up* or *down*) of a note automatically. This can, however, be changed by the user, as needed:

- the stem direction is forced to go *up* with the letter `u` ('**up**') anywhere after the note name, or *down* with `l` ('**lower**'),
- the stem length of non-beamed notes can be shortened by adding the option `S`, followed by a decimal number between .5 and 4.0 , representing the shortening in units of `\internote`²². The shortening can be made "sticky", i.e. applied to all following notes by adding a colon (`:`) after the number. Stickiness is terminated by `S:` .

B 4.4 Other Note Parameters

Here are some more options that can be appended to a note symbol (as usual without spaces):

Inhibited beaming (`a`)

Joining the note stems with beams is usually done automatically by PMX²³. Sometimes, however, you may want to inhibit this explicitly for a single note; this is achieved by the letter `a` (for *alone*).

Horizontal shift (`e,r`)

Sometimes you want to shift the horizontal position of a note, for example to avoid overlap with a note in another voice of the same staff. The letters `e` and `r` do that: `e` shifts the note to the left by its own width (a 'notehead width'), while `r` shifts it to the right. An example of this can be seen in Fig. B.7.

Shift of accidentals (`< / >`)

Accidentals can be shifted too. One way is to enter `+` or `-` immediately after the accidental character, then an *integer* for the vertical shift in units of `\internote`, then another `+`

at the start of a new block. Duration is never inherited across block ends, and thus must be reset at the start of each input block.

²¹There is an exception to this: if you decide to indicate the octave with an explicit digit ('absolute octave'), you must then enter the duration explicitly as well.

²²1 unit of `\internote` is roughly half the distance between staff lines.

²³For details of this cf. Section B 4.10.

or `-`, followed by the horizontal shift, given as a decimal fraction of a notehead width. If you use this method, you *must enter both numbers*. Alternatively – if you need only an horizontal shift `-`, you can simply 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.

Shift of the dot in dotted notes

The dot in dotted notes can be shifted analogously to the first method for accidentals, using signed numbers, except that the first (vertical) shift may be a decimal fraction.

Shorthand for dotted notes

There are two special shorthand notations for dotted rhythms. For normal dotted rhythms (3 : 1 ratio), if you include a period (`.`) in the note symbol, **PMX** will

- assign a dot to the note just entered,
- terminate that note,
- prepare to receive the next note name *without any space*, and
- 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 parameters may. Ornaments and slurs (cf. Sections B 4.9 and B 4.11) following this symbol will apply to the second member. If you need to follow the main note with some modifying command, you can still use the shorthand (`.`) after that command and a space. The main advantage of this shorthand comes if you want to follow one dotted pair with another of the same rhythm; then you needn't enter any explicit duration value for *either* member of the second pair. This is because after using the shorthand, the basic inherited duration value is set to that of the *first* note in the pair; remember that the basic duration does not include the dot!

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

If you use this shorthand for dotted pairs, either embedded in a single symbol or to start a separate symbol for the second member, then you *cannot shift the dot position*. If you do need to shift the dot, you should not use the shorthand notation.

For reference, all the letter-type note parameters are listed in Table B.5. Examples demonstrating the use of most of them are given in Fig. B.10 .

Accidentals:	
s	— sharp
f	— flat
n	— natural
ss	— double sharp
ff	— double flat
sc, ssc, fc, ffc, nc	— cautionary accidental (accidental in parenthesis)
si, fi, ni	— MiDI accidental
	[does do not appear in the printed score — cf. Section C 2.2]
Dotted Notes:	
d	— single dot
dd	— double dot
.	— shorthand for 3 : 1 rythm
,	— shorthand for 2 : 1 rythm
Stems:	
u	— force the stem up
l	— force the stem down (lower)
Shifts of Position:	
e	— shift the notehead left by the notehead width
r	— shift the notehead right by the notehead width
<	— left shift of accidental
>	— right shift of accidental
+, -	— general shift of accidental
Beam Inhibit:	
a	— alone (cf. Section B 4.10)
Xtuplets:	
x	— (xtuplet) (cf. Section B 4.6)

Table B.5: Use of Note Parameters as shown in Fig. B.10 .

B 4.5 Rests

Rests are denoted by the letter **r** in place of a note name. Rests have duration digits just like notes, but obviously no second digit for giving an octave. As inheritance goes, an **r** behaves just like a note: it can inherit its duration from the prior note or rest, and notes as well as rests can inherit their duration from prior rests.

Any rest or sequence of rests that occupies a full bar will by default be horizontally centered in the bar.

There are a few special notations for rests:

- **rp** ('rest **p**ause') denotes a full-bar rest (whole rest), regardless of what the meter signature for the bar may be,
- **rpo** ('rest **p**ause **o**ff-center'). The option **o** suppresses the centering of a pause,

(Notations for Dotted Notes)

(Position Shifts of Accidentals)

Figure B.10: Use of Note Parameters

- `b` ('blank') denotes a blank rest, i.e. one that *does not appear in print*. Logically, it has a duration that is determined, as usual, either by an explicit duration specifier or by inheritance. Blank rests are most commonly used when there are two voices in a staff, and one drops out for some part of the current input bar²⁴,
- `rm` ('rest multi-bar') followed – without space! – by an integer n generates the multi-bar rest symbol with the number n above it, signifying a rest for n full bars²⁵.

Note that that there can be *only one instance* of a multibar rest in a given input block!

The default vertical position of a rest depends on whether there are one or two voices in the staff. For one voice it is just the MusiX_{TEX} default: approximately centered on the middle line. On the other hand, in the lower voice in a two-voice staff, the rest is lowered by `4\internote`, while in the upper line it is raised by `2\internote`. The default can be manually overridden by appending `+` or `-` and an integer for the offset (in `\internotes`) from the *middle* line of the staff. Note that this means that, if in the case of 2 voices in a staff you want to have the rest centered on the middle line, you need to code it as `r+0` — see the last bar in Fig. B.11 .

Some samples of rests are given in Fig. B.11.

²⁴Another interesting application is when you have two homophonic voices in one staff and, for simplicity's sake you want to write only one rest for both voices (cf. the last 2 bars in Fig. B.11) .

²⁵This is used in particular when the parts for individual instruments are generated from a full score, as is done automatically by `scor2prt`; cf. Section C 1).

The figure shows two musical staves illustrating various rest notations. The first staff contains the following rests: r0, r2, r4, r8, r1, r3, r6, rb, r4+3, r8-6, r, r, .r, rdd, and r3. The second staff contains: rp, rpo, rm15, r4, r4, and r4+0, r4b. The number 15 is written above the rm15 rest.

Figure B.11: Rests

B 4.6 Xtuplets

PMX insists very strictly on checking that the sum of durations of all notes or rests in a bar add up to the total required by the bar's meter. Thus, it cannot deal easily with modern extensively polyrhythmic-scores, percussion in particular ²⁶. The only polyrhythmic feature built into **PMX** are the traditional xtuplets: doublets, triplets etc. , together with their usual notation.

Xtuplets (duplets, triplets, etc.) can have from 2 to 24 notes or rests ²⁷. By default all notes in an xtuplet have the same duration; but some can be dotted or have twice the basic duration (cf. p. 31). The notation is as follows:

1. The symbol for the first note of an xtuplet begins exactly like a note symbol, with the name of the first note in the xtuplet, or an `r` if the xtuplet starts with a rest `-`, followed, as the case may be, by an accidental, a `d` or `dd`, a `+`, `-` or explicit octave, and an optional duration digit. However, this duration, whether given explicitly or inherited from a previous note or augmented by a dot, is not the duration of the first note, but represents the *total* duration of the *whole xtuplet*.
2. If the xtuplet is to be *unbeamed*, add an 'a' ('alone') right after the first note.
3. Next `-` with no space, as usual `-` comes `x` (for 'xtuplet'), followed by a one- or two-digit integer, for the number of notes in the xtuplet . If the first note is to be dotted, add the usual `d`, if it is to have twice the basic duration, add a `D` or `F` (cf. page 31 for more detail on this). The only options allowed after this begin with the letter `n` and control the printed appearance of the xtuplet:
 - If `n` is omitted, the xtuplet is printed in the standard way, i.e. with the xtuplet 3 printed over (or under, as the stemming may dictate) for, say, a triplet. Note also that by default **PMX** prints a bracket only if the xtuplet notes are unbeamed; otherwise just the xtuplet number is printed.

²⁶But such problems occur occasionally even in classical music. Thus, a real challenge to any **PMX** expert is to code the (in)famous third movement of Mozart's Oboe Quartet KV 370, where suddenly the oboe changes to an alla breve (4/4), while the strings continue on in a 6/8 meter!

²⁷The default maximum of 24 can be changed. Cf. **D 1** .

- If **n** is followed by an *unsigned* integer, this integer is taken as the number to be printed instead of the natural (default) one.
- If **n** is followed by the letter **f** (flip), the xtuplet number is flipped vertically from its default position.
- The position of the xtuplet number can be adjusted in the usual way with one or two *signed decimal numbers* following **n**: the first is a vertical shift in units of `\internote`, the (optional) second a horizontal shift, in notehead widths.
- If **n** is given, but followed by a space, thus ending the first-note symbol, *no number* at all will be printed.
- For a *non-beamed xtuplet only* a suboption **s** can be added to the **n** option by which the slope of the bracket can be adjusted ; i.e. you can use **ns**[*n*], where *n* is an *signed* integer (except zero): a positive integer will lift the end of the bracket upwards, while a negative one will move it downwards.

Note that this suboption will operate in a more elegant way for Sichernman-type xtuplets ²⁸ only: when the Sichernmann option **AT** is *not in effect*, the bracket number (the 3 in a triplet, e.g.) will *not* follow the slope adjustment and might have to be tweaked separately.

4. The second through last notes of the xtuplet are then each given by a separate note symbol, containing the meaningful subset of the parameters permitted for notes or rests:
 - (a) the note name as the first character. This is the only required character. It can be an **r** (for a rest), except that the *last note of an xtuplet cannot be a rest* ²⁹,
 - (b) an accidental,
 - (c) an octave change (+ or -). The octave may also given explicitly; this is the only digit allowed,
 - (d) a **d** (**dot**). The next note after the dotted one is automatically shortened to half the normal value,
 - (e) the character **D** in the note symbol for any note in an xtuplet doubles the duration of this note. As this accounts for two notes of the xtuplet, it will decrease the expected number of notes in the xtuplet by one. If used for the first note of an xtuplet, **D** goes *before* an optional **n** parameter,
 - (f) the character **F** is the same as **D**, except that in the printed output the doubled note will appear dotted (a notation Bach sometimes used). As with **D**, an **F** for the first note goes *before* an optional **n** .
 - (g) grace notes are allowed in xtuplets.

No explicit duration is allowed in symbols for the second through last members of the xtuplet, the duration having been determined by the first note.

²⁸For Sichernmann xtuplets cf. the next page.

²⁹Cf. however, Sec. E 2.2

Beaming of xtuplets is done automatically. If it is to be inhibited, add the `alone` option to the first-note symbol (cf. Section B 4.4).

Fig. B.12 displays some examples of xtuplets, with the **PMX** coding given below the staff:

Figure B.12 shows two staves of musical notation. The first staff contains a 5-note xtuplet (e4x5), a 3-note xtuplet (c4x3n-1), a 14-note xtuplet (c2x14n+2-1), a 3-note xtuplet (f4x3), a 3-note xtuplet (g-x3d), and a 3-note xtuplet (cx3). The second staff contains a 5-note xtuplet (f4x3Dnf-1), a 3-note xtuplet (bx5D), a 3-note xtuplet (d4x3Fn-2), a 2-note xtuplet (cd4+x2), a 4-note xtuplet (fd4-x4n), and a 4-note xtuplet (fd4-x4n4).

Figure B.12: Xtuplets

As was remarked at the end of Section B 3, the individual symbols of **PMX** generally consist of one or several characters strung together without spaces. In that sense, each note of an xtuplet is considered to be *a separate PMX* symbol; so these are separated by spaces as usual.

A special, non-standard bracket can be printed for *non-beamed xtuplets only* with an addition to **PMX** designed by Col. G.L. Sicherman. This prints the xtuplet number not above or below the bracket, but in line with it, as shown in Fig. B.13. To invoke this, simply add the option `AT` as a separate **PMX** symbol anywhere in the body of the source file: this will switch from standard to Sicherman brackets for *all xtuplets in the score*.

Figure B.13 shows two staves of musical notation, identical to Figure B.12 but using Sicherman brackets. The first staff contains a 5-note xtuplet (e4x5), a 3-note xtuplet (c4ax3n-1), a 14-note xtuplet (c2x14n+2-1), a 3-note xtuplet (f4x3), a 3-note xtuplet (g-x3d), and a 3-note xtuplet (cx3). The second staff contains a 5-note xtuplet (f4x3Dnf-1), a 3-note xtuplet (bx5D), a 3-note xtuplet (d4x3Fn-2), a 2-note xtuplet (cd4+x2), a 4-note xtuplet (fd4-x4n), and a 4-note xtuplet (fd4-x4n4).

Figure B.13: Xtuplets with Sicherman brackets

If you want to use this for specific xtuplets only or otherwise customize your use of the option, you will have to use inline $\text{T}_{\text{E}}\text{X}$ commands, following instructions given in `tuplet.tex` as comments.

To use the Sichertmann option, the file `tuplet.tex` (available from the software section of the WIMA) must be installed in your T_EX system!

B 4.7 Chords

Chordal notes are notes that share a common stem and have the same duration. They are coded in **PMX** as follows:

- One of the notes in the chord is chosen to be the ‘first note’. The first note is written as usual and as described in Section B 4.1 ; all other chordal notes are denoted by separate symbols starting with **z** , followed by a note name. The note name can optionally be followed by an accidental and an octave indicator (+ or -), but no absolute octave is allowed. And no duration value is allowed either, as this is logically determined by the first note.

Legally the first note can be any one of the chord notes; but wise use of this freedom of choice is definitely called for, since

- the first note sets the default stem direction,
- in a beam it can affect the height and slope of the beam,
- if the chord is beamed with other notes, the default height and slope of the beam will be determined by the ‘first note’ of the chord.

If a beam joining chords looks bad, you can often fix it by changing which note acts as the main one [of course there are other ways to fine-tune beam parameters, as will be described in detail in Section B 4.10]. For further details, consult Section B 4.7.1 .

If the main note is dotted, the dot is obviously ‘inherited’ for all other notes as well, so writing another **d** would be redundant. But if you want to shift the dot of a chordal note, you *must add* a **d**, followed by the shift parameters, as described on p.27.

- Normally **PMX** will automatically set notes in a chord so that collisions are avoided (in a second, e.g.). If you don’t like the result, you can shift the note by one notehead width to the right with **r**, or to the left with **e** .
- The position of accidentals will also be adjusted automatically by **PMX** so as to avoid collisions. But, as with single notes, you can shift accidentals of chord notes manually³⁰, using the parameters described on p.26.

If you manually shift *any* accidental in a chord, then automatic shifting will be disabled for *all* the accidentals in a chord, *unless* you preface the shift parameter with **A** (e.g. `zcsA<.5`), in which case the manual shift will be added to that applied automatically by **PMX**.

³⁰Note that if, in addition, you use a + or - to denote a relative octave (cf. p.25), this sign must come *after* the shift parameter for the accidental.

- Another option that affects accidental positioning in chords is `Ao` , entered in the main note symbol of a chord. It will force the accidentals in that chord to be posted in the order they come in the source (starting with the main note), each one as far to the right as it will go without crashing into a notehead, stem, or another accidental.

The stem length and direction of a chord are controlled by the first note, but may be manually overridden with `u` (up) or `l` (lower) in the first note symbol.

As described above, every chord note has its own note symbol. This means, as always: no space within a note symbol, but at least one space between the note symbols of a chord.

A beginner's trap

When slurring two chords, you may get confused by a simple programming error. Take, e.g., the sequence

`c45 zf d zg-` .

The first notes of the two chords are `c` and `d` , the chord notes (denoted by `z` in **PMX**) are `f` and `g` . However, **PMX** following your instruction, sets the lower notes as *c and g* , the higher ones as *f and d* .

So if you want to slur the chord say, in the higher notes (as shown in the picture),



you must obviously write

`c45 zf s d s zg-` , not `c45 zf s d zg- s` .

Moral: if you want to slur chords, look at the *names of the notes* you want to slur, not the `z` symbols!

B 4.7.1 Arpeggio

An **arpeggio** is, of course, simply a special form of a chord, denoted by a vertical wavy line in front of the chord proper. To write that in **PMX**, place the symbol `?` after the symbols, with a space as usual, for both the first and last note of the chord.

When doing that, you should remain aware of the fact that the wavy line is generally understood as being played from the bottom up: lowest note first, highest last.

So far, so easy. There are, however, a few special cases to note:

1. In an arpeggio, the wavy line often turns out to be crowded to the chord itself; this happens in particular when some of the chord notes have accidentals. In such cases one would want to move the arpeggio line away from the chord itself a bit, i.e. to the left.

This is easily achieved by adding a `-` and an (unsigned) decimal number, without spaces, to one of the `?` designating the arpeggio — it doesn't matter which one.

2. Sometimes the composer may want to have the arpeggio played in a more virtuose way (on string instruments, in particular): the first 2 – or even 3! – of the chord notes simultaneously, as a double (or triple) stop, then the other chord notes as a true arpeggio.

This can be done easily in **PMX**: simply place the first ? not after the starting chord note, but later, after one of the `z...` notes.

In this case it is even more important to remember the standard meaning of the wavy line in musical language that was just pointed out! So in general, it is best to stick with the order of the chord notes “bottom to top” unless you really know what you are doing.

3. In piano music, it is quite common that an arpeggio is meant to go across the chords of both hands. This also is easily implemented in **PMX**³¹: simply place the first ? at the start of the left-hand chord and the second one *at the end of the right-hand chord* — as common sense would suggest!

To see all this in examples, look at how the chords in Fig. B.14 were coded. When comparing this figure with its code it should become obvious that care is called for when producing chords with accidentals, and arpeggios in particular: some of the chords in bars 1 and 2 look ugly, to say the least, and the arpeggio line in bar 3 as well as the first one in bar 4 seem to be downright violations of the standards of good typesetting practice.

B 4.8 Grace notes

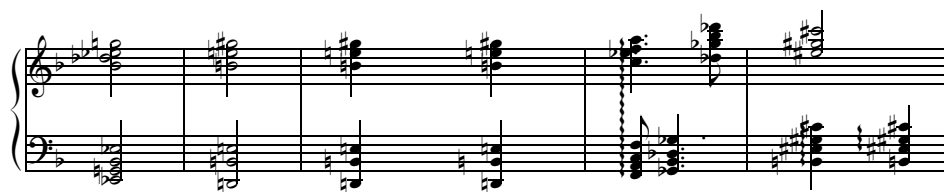
A grace note symbol starts with a **G**. It is entered in its natural order, i.e. before the main note symbol for a normal grace, behind the main note symbol for an after-grace . After **G** comes any combination of the following options:

- A single digit representing the number of notes³² in the grace. The default is 1 .
- **m** (for multiplicity) and a digit, representing the number of flags or beams. The default is 1; 0 is allowed.
- **s** (for slur): this joins all notes of the grace to the main note. No **s** is needed in the main note symbol.
- **x** for a slash (only for single graces),
- **l** or **u** to force the direction of the stem(s) as desired.

Next comes the only required character, the first grace-note name. No time value must be entered, but, if needed, an octave (relative or explicit) or an accidental can be given as in a

³¹To my knowledge, this feature of **PMX** (available from Version 2.503 on) has not been explicitly documented in the official **PMX** documentation. I am very grateful to André van Ryckeghem for drawing our attention to this feature in one of his contributions to the WIMA “Tips & Tricks” WEB page.

³²The *maximum* number of notes in a grace is not fixed, but depends on circumstances (cf. Section D 1); but up to 16 is usually safe.



```

% bars 1-3:
e22f zgn zb zef |
dn2- zbn+ zen   |
dn4- zbn<1.5+ zen dn- zbnA<1.5+ zen X4 /
b24 zdf zef zgn |
bn2- ? zen zgs   |
bn4- zen<1.5 zgs ? bn- zen<1.5 zgs X4 /

% bar 4:
f8- ? za  zc  zf  gd4f- zb  zdf  zgfd+0.5+2 /
cd4- zef  zf  za  ? d8f- zgf  zb  zdf /

% bar 5:
c4sAo ? zgs zes zbn ? X2 bn4 zes ? zgs zcs ?-2.9 Rb /
es2-u zgs zcs /

```

Figure B.14: Chords

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.

‘After’-graces (graces coming *after* a main note) are entered similarly to normal graces, with the same parameters as above; but in addition to the other parameters either an **A** (for **A**fter) or **W** (for **W**ay-after) is entered, before the grace note(s).

After-grace symbols associate the grace note(s) with the *prior* main note and are therefore entered *after* the main note symbol.

By default, **PMX** will place graces or after-graces *immediately* before or after the main note, way-afters as far to the 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.

For a grace (*not for an ‘after’-grace*), you can, if necessary, add some space between the grace and the main note by inserting an **X** followed by a unsigned decimal in the grace symbol³³.

Examples for grace notes are displayed in Fig. B.15.

³³It doesn’t matter where you insert this shift symbol, as long as it makes syntactical sense; to state this more simply: the $X[n]$ may *not be inserted before a number* and *not after the leading note* of the grace.

This added space for a grace is indeed essentially a hard space, so that its name, **X**, is quite appropriate. And, as with almost all such cases, it is wise to keep in mind that ‘hard spaces’ are what their name implies: a hard – or brutal, if you will – last resort if nothing else seems to help. Used unwisely, adding a hard space to a grace is liable to produce rather ugly results, as is exemplified in Fig. B.15.



```
w150m
Abepl
\\nobarnumbers\
%
% Bar 1
G3sm2g++ a b c4 f- G2slAe d c GsX1xb+ c /
% Bar 2
Ga- g4 Gfs- g c G13sm3X3d e f g a b c d e f g a b c /
% Bar 3
c2- G3slWb a b c2 of Rb /
%
```

Figure B.15: Grace Notes

B 4.9 Ornaments

Symbols for ornaments are entered *after* their associated note symbol, separated, as usual, by a space³⁴.

The ornaments available in **PMX** are listed in Table B.6 and are illustrated in Fig. B.16.

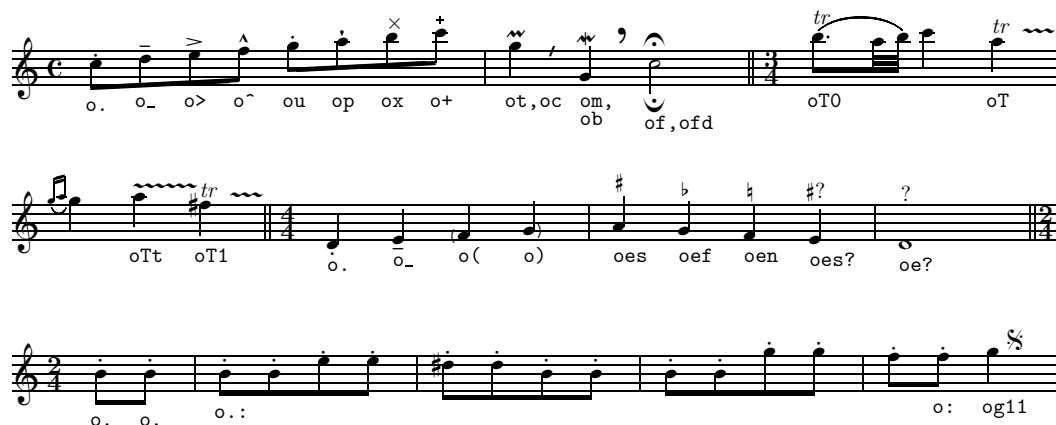


Figure B.16: Ornaments

All except the staccato, tenuto, down fermata and parentheses will appear above the staff; staccato³⁵ and tenuto appear just above or below the note head, depending on the stem

³⁴The ornament, although associated with a specific note, is considered a separate **PMX** symbol.

³⁵In fact, the only difference between staccato and pizzicato is the vertical positioning of the dot.

staccato	o.
tenuto	o_
accent	o>
sforzando	o^
pizzicato	ou
spiccato	op
×	ox
+	o+
caesura †	oc
breath †	ob
upper fermata	of
lower fermata	ofd
shake	ot
mordent	om
mordent [right parenthesis before notehead]	o)
appoggiatura [left parenthesis before notehead]	o(
trill †	oT x
implicit trill †	oTt x
segno †	og x

editorial marks:

editorial accidentals	oes oef oen
dubious accidental	oes? oef? oen?
dubious note	oe?

ornament repetition † :

† See explanation in main text

Table B.6: Ornaments

direction, the parentheses at the level of the note head, of course.

The trill and segno symbol are special in that they may have additional optional characters. Either trill symbol may include a decimal number to specify the length of the wavy line indicating the duration of trilling, in `\noteskips`. The default is 1. Thus `oT0` will be a *tr* without any wavy line, and `oTt2` is a wavy line of 2 `\noteskips` without any *tr* symbol starting the wavy line.

A segno may only be entered 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 staff of the system.

Once the ornament type has been specified, most ornaments can be raised or lowered from their default position by appending a signed integer to the symbol, giving the vertical offset in `\internotes`. Caesura and breath may in addition have a signed number, giving the horizontal shift from default in notehead widths. These two ornaments also differ from the others in their default horizontal position, which is 0.5 `noteskip` past the note.

An ornament can be automatically repeated on a series of consecutive notes, provided the notes are all in the same input block. To activate this feature, terminate the first ornament symbol with `:`. Then every note in that voice will have the same ornament until a note is followed by the repeat terminator `o:`.

B 4.10 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 notes to be beamed with `[` and `]` (with spaces, as usual). Conversely, if **PMX**'s selection is to beam notes, and you want to inhibit this for a single note, add the option `a` to the note symbol (cf. Section B 4.4).

One may also wish to edit certain features of a beam even when **PMX**'s grouping decision would otherwise be acceptable; this is done by adding some parameters to `[` and `]`:

The `[` can be followed, without space, by one or several of the following options:

- `u`, `l` or `f` will override **PMX**'s selection of the direction of the beam:
 - `u` makes the beam go *above* the noteheads (**u**pper beam),
 - `l` makes the beam go *below* the noteheads (**l**ower beam), while
 - `f` will flip the beam from whatever **PMX** would do automatically;
- `h` forces the beam to be **h**orizontal;
- `m` followed by a single digit (1, 2, 3 or 4) forces the **m**ultiplicity of the beam;

- `j` joins a beam grouping to a prior one started in another staff (cf. p. 41).

The beam symbol constructed so far can be followed, without space, by one, two or three consecutive integers, each preceded with + or - :

1. first is an adjustment of the starting height; it is given in `\internotes` and may range from -30 to 30,
2. the second is a slope adjustment (again in the range from -30 to 30),
3. The third is an additional adjustment to the starting height, given in units of the beam thickness. It may range from 1 to 3 only. It always acts to *increase* the stem length. This would only be used in rare cases, e.g. to align consecutive horizontal beams which have internal multiplicity changes. An example is shown in Fig. B.17, bar *f*.

Note that for technical reasons all three integers must always be given *with a sign* (even the last one, although it always is positive). Note also that, since the meaning of these numbers is interpreted by **PMX** in their sequence, if you need only the second or only the third number you *must not omit* the preceding number(s), but rather give explicit 0's for them.

```

% Bar a:
f1s c f a c fs a c- b g+ b g [l+12-8 b-- g1++ b g ] Rd /
% Bar b:
[l f1s- c f a ] [u c fs a c- ] [f b g+ b g ] [ b-- g1++ b g ] Rd /
% Bar c:
[m4 f1s- c f a ] [m1 c fs a c- ] b g+ b g [h b-- g1++ b g ] Rd /
% Bar d:
[ f1s- c f a ][ c fs a c- ] [ b g+ b g ]-[ b-- g1++ b g ] Rd /
% Bar e:
[ fs-4x3nf c a+ c1 fs a c- ] b g+ b g b-- g1++ b g Rd /
% Bar f:
m3434
cd84 c3 c6 c [+0+0+3 cd8 c3 c6 c ] [-1+0+3 cd8 c3 c6 c ] Rd /

```

Figure B.17: Beams

Here are some further possibilities regarding beaming:

- By default, xtuplets (cf. Section B 4.6) 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.
- Sometimes one may wish to define beamed groupings with subgroups joined by a single beam. The symbol] [, standing alone between two note symbols inside a forced beam, causes the multiplicity to decrease to unity and immediately increase to its natural value for the next note. An example of this, as well as the beaming together with an xtuplet, can be seen in bar 2 of the trombone voice of Fig. B.3 : the two doubly-beamed groups connected by a single beam are generated by [c15 c c] [c c cx3 c c] .
- Related to this is a *single-slope beam group*, which is the same as described in the previous item except that there is no connecting beam between the beamed subgroups. For this option, the beamed subgroups are separated by]-[standing alone between two notes inside the forced 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 involved³⁶,
 2. for proper appearance in crowded scores you may wish to insert hardspace or shifts as described in Section B 6.4.
- 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 other voice on a neighboring staff with [j , ending that part with the normal] . For staff-jumping beams, it's OK to have just a single note inside one or both of the partial beams. Each voice must, however, still have the right number of beats, so you will probably need to fill in the durations with blank rests after the first members of the beam and before the second.

With staff-jumping beams, some adjustment of the beam height and slope will usually be required. Also, sometimes the up/down-ness of the ending section must be overridden; you will know that this is necessary if on your first try the ending is shifted horizontally from the desired position. These editing commands may sometimes produce unexpected results, and some iteration may be required to get exactly what you want.

An example of a staff-jumping beam is seen in Fig. B.18 .

³⁶This is due to a fundamental technical limitation of MusiX_{TEX}.

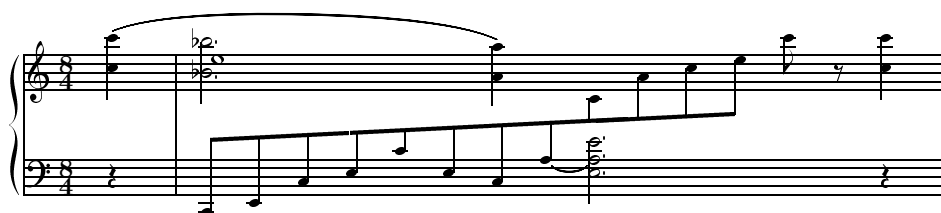


Figure B.18: A staff-jumping beam

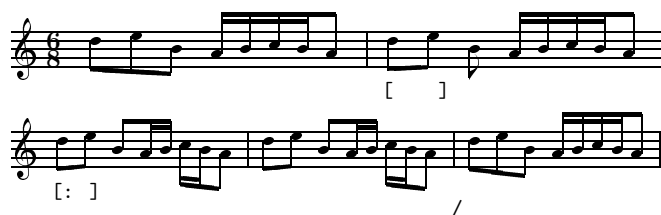
- If the option `:` (colon) is appended to the starting symbol `[` for a forced beam (‘automatic forced beaming’), then henceforth beams of the same length will be force-beamed similarly in that line of music *until the end of the input block*. Within an input block, the automatic forced beaming can also be stopped explicitly by starting a new regular forced beam.

But remember that **PMX** is actually quite good at beaming notes in the manner a human musician would normally expect; so it will be necessary to use this facility only in the special cases when you want (or need) beaming different from what **PMX** provides on its own. In such cases, consider this beaming automatism as a kind of shorthand; as such, it will be really worth your while only when you want to repeat the same ‘non-orthodox’ beaming an appreciable number of times.

Regarding automatic forced beaming, you should also be aware of the following:

As with regular beams, you can have notes of different durations beamed together in an automatic forced beam. The rule in that case is: in the beams following the initial one, the next sequence(s) of notes with the *same total duration time* as that of the notes beamed together initially will then be beamed together automatically. As usual, notes given without duration will ‘inherit’ their durations from the most recent note with an explicit duration.

All this is best seen in the following example:



which is coded as follows:

```

d85 e b a1 b c1 b a8
[ d85 e ] b a1 b c b a8
[: d85 e ] b a1 b c b a8
d85 e b a1 b c1 b a8 /
d85 e b a1 b c1 b a8 Rd

```

Note how the automatic beaming ends at the *end of the first input block*.

A last remark:

PMX was never intentionally designed to produce beams across bar lines. However, in an (undocumented) exception, **PMX** *in some cases* does accept beams that cross bar lines: as it turns out, the machinery designed for staff-jumping beams may also be used to force a beam across a bar line, whether the beam jumps to a different staff or stays within the same staff.

The syntax for that feature is applied in the obvious way, as described on p. 41, and then applied similarly in the single-staff example of Fig. B.19.



```
cd24 c8 [+0+1 c ]j | [jf c d e f ]
g4 g8 [+0-1 g8 ]j / [j f84 e d c ] c2 /
```

Figure B.19: An example of bar-crossing beams within a single staff

B 4.11 Slurs and Ties

Pick a package

By default **PMX** will use MusiX_{TEX}'s built-in, font-based slurs and hairpins (crescendi and diminuendi, see Section B 4.12, p. 52 ff). To use these, you don't have to install any more software. But they do have drawbacks: the fonts used by MusiX_{TEX} for font-based slurs come in fixed sizes and shapes and are therefore often not flexible enough in complicated situations. In contrast, slurs and hairpins produced with the help of the PostScript language overcome many of the shortcomings of their font-based counterparts.

There are two independent “third-party” packages that provide MusiX_{TEX} 1.15 and **PMX** users with PostScript-based slurs, ties, and hairpins:

Type K – This package was written by **Stanislav Kneifl**. It is directly supported by **PMX** and will be the focus of any future **PMX** enhancements. Detailed instructions for its use are given in the corresponding paragraph below.

Type M – This package, by **Hiroaki Morimoto**, is somewhat more flexible when used directly in MusiX_{TEX} 1.15, but it requires the **Metapost** package to be installed in the _{TEX} installation. These are not directly supported by **PMX**, but are advertised to be fully compatible with default font-based slurs of MusiX_{TEX} 1.15. From **PMX**'s standpoint they are not different from font-based slurs, and the same options and features that apply to font-based slurs should apply as well to these.

Both PostScript slur packages provide excellent slurs, ties and crescendi. In fact, for simple slurs, you can hardly discern the differences between the three possibilities, as can be seen in Fig. B.20 — although type M slurs are generally somewhat flatter, indeed. On the other

hand, for steep and long slurs, the difference is appreciable, as can be seen in the example of Fig. B.21.

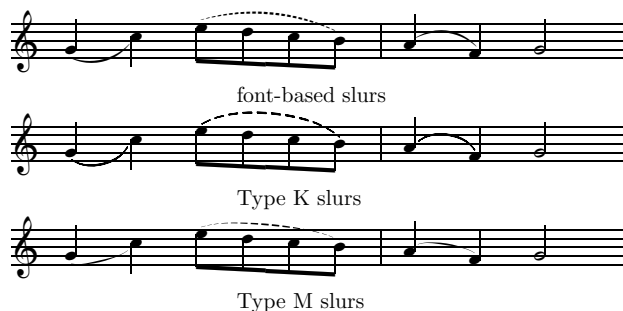


Figure B.20: Three realizations of simple slurs

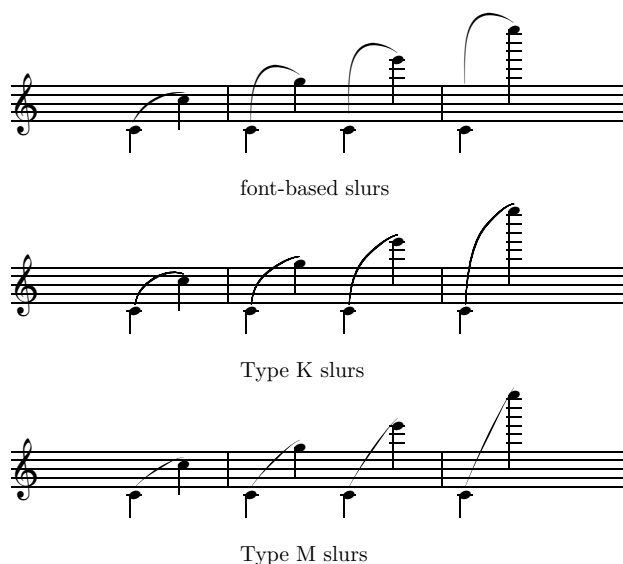


Figure B.21: Three realizations of exotic slurs

Here are some additional factors to consider when deciding which type of slurs and hairpins to use with **PMX**:

- True ties, which are shaped differently from slurs, are only available with Type K³⁷. In all cases, the starting and ending positions are different for ties and slurs³⁸.

³⁷In MusiX \TeX 1.15, font-based true ties are actually available; but they were never incorporated in **PMX**, now being superseded by type K slurs.

³⁸The ends of an ordinary slur are centered horizontally above or below the notehead, while tie ends are shifted inboard and closer to the midheight of the notehead.

- Font-based hairpins cannot wrap over a line break, but the PostScript ones can.
- Font-based hairpins cannot be longer than 68 mm.
- With either `postscript` package, the resulting markings do not show up on the screen if you view the score with one of the commonly used DVI viewers such as **XDVI** or **YAP**. To see them, you have to first produce a PostScript file from the `dvi` file (using, for example, `dvips`), which you can then view on the screen and print on your printer using **Ghostview** or some other such program. But if you use Type 1 PostScript fonts, as was recommended in Section B 4.11, you will have all this software at hand anyway!

Detailed instructions for installing the Type K or Type M packages are given in Section B 4.11 .

B 4.11.1 General slur usage

This subsection explains commands that apply to all types of slurs, followed by separate subsections describing operational features specific to either font-based or Type K slurs, and finally a subsection on the use of Type M slurs.

Here are the **PMX** symbols that normally define slurs or ties, except for slurs to or from graces³⁹:

- An opening parenthesis, ‘(’, begins a slur, and a closing parenthesis, ‘)’, ends it; similarly, an opening brace, ‘{’, begins a tie, and a closing brace, ‘}’, ends it. The opening parenthesis/brace is placed *before* the first note, and the closing parenthesis/brace *after* the last note of the slur or tie. As always, these symbols (including options, see below) are separated from notes and other symbols by spaces.⁴⁰
- Alternatively to parentheses, you can use the letter **s** (for slur) both to begin *and* end a slur; but these come *after* both the beginning *and* ending note. This is thus a *toggle*, turning a slur off if it’s on, and starting a new one otherwise.

For ties (instead of slurs), use the letter **t** (for tie), also a toggle.

Note that slurs or ties may end on a rest⁴¹, but they may *not start* on one. The default ending height of the tie (or slur) in that case will be that of the note on which it started (in other words, it will be horizontal); but you can vary that explicitly in the usual way.

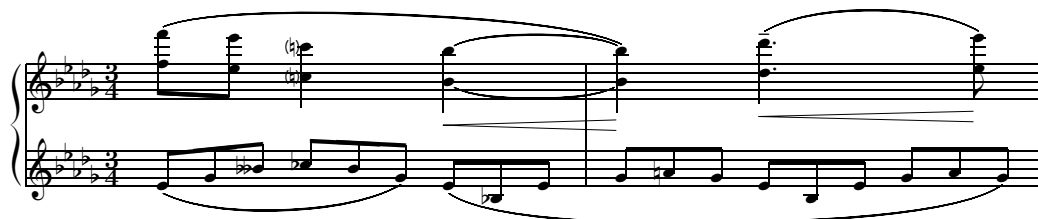
Often you need to open a slur or tie while another one is already open *in the same voice*, e.g., with tied chords, or if a second slur connects a subgroup of notes within a longer slur.

³⁹Recall that the slurring of grace notes is defined entirely within the the grace symbol, as described in Section B 4.8.

⁴⁰Braces – and similarly the tie toggles described subsequently – , although legal with all types of slur, are really useful only with Type-K slurs, since only with these they will produce a true tie.

⁴¹If you happen to think that this would be musically meaningless, you are wrong. Just watch a really good pianist (or conductor, for that matter) ending a piece, say in *pp*. It ain’t over until the fat lady has really finished singing: until she is really *completely done* with taking her hands off the keyboard! In other words: she visibly (and audibly!!) plays the slur over the final rest.

In such cases, **PMX** needs to be told which slur is which. For this purpose, the slur opening character [`s`, `t`, `(`] can optionally be followed by a single-character ID code [`0-9`, `A-Z`] to identify that particular slur. To close that slur, insert the ID code right after the corresponding *closing* character [`s`, `t`, `)`]. If an ID code is used, it *must* be the second character in the symbol. You cannot use any ID that is already in use on a slur or tie for a new slur or tie. See Fig. B.22 for an example⁴².



```

Ap
% Bar 25:
(A e4x3n g bff cf4x3n b g )A (B e4x3n bf e /
f8+ (C zf+ e- ze+ c4nc- zcnc+ (Dt1 b- D< (Et zb+ /
% Bar 26:
g4x3n an g e4x3n b e g4x3n a g )Bh Rb /
b4- )Dt D< zb+ )Et )C dd4- D< (F+1 zd+ o_ e8- D< ze+ )F /
%
```

Figure B.22: **E. Bloch**, *Waves (Poems of the Sea I)*, bars 25,26

Normally, you can leave further details of slur appearance to **PMX**; the result will usually satisfy. If not, you can change some features manually by additional options which go right after the initial character and any ID code, as usual *without* a space,:

- The default vertical position above or below the note heads can be overridden with
 - `u` (**u**pper),
 - `l` (**l**ower) or, equivalently, `d` (**d**own).
- The starting or ending point can be shifted from its default by entering one or two explicitly signed numbers: the first, which must be an integer, gives the vertical shift, in `\internotes`; the second, which may be decimal, the horizontal offset, in notehead widths.
- A dotted slur is obtained by adding the option `b` (for **b**roken) in the symbol that starts the slur (cf. Fig. B.23).

⁴²The example uses Type K slurs.



Figure B.23: A dotted slur

Slurs for staff-jumping beams

Often – in piano music in particular – when you have a staff-jumping beam (cf. p.41), you may want to join the two parts by a slur as well. **PMX** will, however, get very confused if you try to put the beginning slur symbol in one staff and the corresponding ending symbol in a different staff; both slur symbols *must always be in the same staff*. The solution is to put the ending slur symbol after the invisible rest; but do so carefully: even though the invisible rests are indeed invisible, **PMX** accounts for the horizontal space it would need for visible rests, and that determines the length of the slur. So if you want to avoid awkwardly adjusting the length of the slur by hand, you should subdivide the invisible rest appropriately so that one part of it will be at the note of the other staff where you want the slur to end. The vertical position of the slur end will have to be done manually. Fig. B.24 shows an example.

```

r4b [j b1 a g f ] r4b [j c14 b a g ] | Rb /
( [ f1 e d D"what you want"-22 c ]j r1b r1b r1b r1b )-15
( [ g14 f e D"what you don't want"-22 d ]j r4b )-15 | /

```

Figure B.24: Placing the slur ending with a staff-jumping beam

B 4.11.2 Invoking and using Type K slurs

To activate the Type K package, all you need to do is to add a line in the header of the **PMX** source after the preamble, containing the symbol **Ap**⁴³.

For type K slurs, some optional parameters can be used in the slur symbol to change the shape of the slur:

- an **f** (for flatter) will flatten the slur a bit, while
- **h**, **H** or **HH** (for higher) will increase the slur's curvature by increasing degrees, thereby raising (or lowering) its middle.

⁴³For further details on the use of **A** see Section B 6.1 .

These parameters can be used in either the starting or the ending slur symbol. They *do not work with ties*, i.e. with symbol **t**, and **PMX** will complain if you try that. Examples are given in Fig. B.25.



Figure B.25: Shape variations in type K slurs

The Type K package contains \TeX macros that activate or deactivate an automatic vertical adjustment of slurs or ties to avoid tangencies with staff lines. In **PMX** these adjustments are switched off by default, because they may alter the endpoint positions from what you would normally expect. However, if you wish to use them, they may be switched on or off globally at the start of any input block, and locally with options to individual slur or tie commands.

To activate these adjustments globally (or deactivate, if previously activated), at the beginning of any input block enter **Ap+s** (or **Ap-s**) for slurs, and **Ap+t** (or **Ap-t**) for ties. To activate them for one-time use only, simply include **p+t** or **p+s** as an option in the symbol for the affected slur or tie. No deactivation is then needed.


You can also sequentially increase (**Ap+c**) or decrease (**Ap-c**) the default curvature of slurs or ties (again at the beginning of any input block), so that the result is cumulative. But after the cumulative change has reached “HH” or “ff”, it will remain there, and nothing more will happen.

An example is given in Fig. B.26. The distinctions are fairly subtle; but some would see the adjusted markings as more pleasing since on close inspection they clearly avoid tangencies with staff lines.

Line-break tie options

There are two options that only affect ties that span line breaks:

1. by default a full tie is drawn at the beginning of the second line in such cases. There is a global option, **Ap+h**, which tells **PMX** to use *half ties* – which are flattened at their left-hand ends – on the second line, but *only* in cases where the second tie segment is shorter than 15 pt. This option may be cancelled with **Ap-h**,
2. the second option – which should not be used together with the first one! – affects *both* the first and second segments of linebreak slurs and ties. It is the global option **Ap1**. It alters two defaults to give what some might consider an improved appearance: first, it causes the first segment to be drawn as a normal tie, whereas by default it has a different shape than a normal tie; second, it moves the starting point of the second segment a bit to the left.



Ap-s
Ap+s
Ap-t
Ap+t

```

Ap
a44 s bu s | b s c s /
Ap+s
a4 s bu s | b s c s /
%
a4 t a t | b t b t /
Ap+t
a4 t a t | b t b t /

```

Figure B.26: Vertical tweaks of slurs and ties

The `Ap1` option has been enhanced since **PMX** version 2.411; the main new feature is that it now works for slurs as well as for ties. Other enhancements are more subtle; read the description of these in the announcement text for version 2.411, given here:

“`Ap1` in the preamble now activates special treatment of line-break slurs and ties (before, `Ap1` only affected ties).

“Specifically, it enables tweaking the ending position of the first segment (seg 1), the starting position of the second (seg 2), and the curvature of either, or both. [You must have `musixps.tex` available to `TEX!`] With `Ap1` every slur/tie at a line break is automatically broken into two separate ones (no additional **PMX** slur start or ending commands are required). Vertical/horizontal tweaks to the start of the seg 1 and the end of seg 2 are handled as before. Vertical/horizontal tweaks for the end of the seg 1 and start of seg 2 are entered as options in the otherwise normal command that starts the slur/tie. The option for the end of seg 1 starts with “s” (for sever or split), then the usual one or two signed numbers, then a second “s” and one or two more signed numbers for the start of seg 2.

“The usual curvature options `h`, `H`, `HH`, `f`, if included in the starting command for a linebreak slur, will apply to seg 1, and if in the closing command, to seg 2. Of course, if the slur/tie does not come at a linebreak, the special position tweaks (those after the “s” option) will all be ignored, and the curvature tweaks on the closing note take precedence (as before).”

Without either of these options, the second segment sometimes becomes too short to be clearly visible.

The global options mentioned (and in fact any of the other global options to be described later) can be combined into a single symbol, e.g., `Ap+s1`.

B 4.11.3 Use of Type M Slurs and Ties

In contrast to the type K slurs, type M slurs are not explicitly supported by **PMX**; as a consequence, you have to do some things manually. But it's not difficult:

- The first step is to insert into the header of your **PMX** source file the following line

$$\backslash\input musixpss\relax\ .$$
- To process a **PMX** file with type M slurs, three extra steps are needed to produce a printed musical score. Here is the full sequence of processing steps, for a file named `my_opus`:

	command		resulting new file(s)
0.	text editor		<code>my_opus.pmx</code>
1.	<code>pmx</code>	<code>my_opus</code>	<code>my_opus.tex</code>
2a.	<code>tex</code>	<code>my_opus</code>	<code>my_opus.mx1</code> , <code>my_opus.slu</code>
2b.	<code>musixflx</code>	<code>my_opus</code>	<code>my_opus.mx2</code>
2c.	<code>tex</code>	<code>my_opus</code>	<code>my_opus.dvi</code>
→ 3a.	<code>musixpss</code>	<code>my_opus</code>	
→ 3b.	<code>mpost</code>	<code>my_opus</code>	<code>my_opus.mp</code> , <code>my_opus.1</code> , ...
→ 3c.	<code>tex</code>	<code>my_opus</code>	<code>my_opus.dvi</code>
4a.	<code>dvips</code>	<code>my_opus</code>	<code>my_opus.ps</code>
4b.	<code>gsview</code>	<code>my_opus</code>	<i>screen view and printed output</i> .

B 4.11.4 Special considerations for font-based slurs

You should study this section if you choose not to use Type K PostScript slurs.

The slur command `t` (cf. p.45) has been retained for backward compatibility with earlier **PMX** versions. As already mentioned, with font-based slurs it does *not* provide a true tie at all, and it doesn't alter the positions of the endpoints either; in fact, it differs from `s` or `(` only in the following ways:

- ID codes cannot be used with font-based `t`-slurs.
- If a `t`-slur 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.

The shape of font-based slurs can be changed with options to the `s` or `)` command as follows:

- at the slur termination only, you must first enter two signed numbers that define a position shift as described earlier. Enter a zero for either or both if you don't want the shift,
- next, enter a signed, nonzero integer which specifies a vertical adjustment to the mid-height of the slur in `\internotes`,
- if you want to alter the starting or ending slope, enter a `:` (colon), then

1. a signed integer for the alteration to the starting slope, then
2. a signed integer for the alteration to the ending slope.

These numbers are passed directly as arguments of the MusiX_{TEX} macros `\midslur` (if only one is given) or `\curve` (if there are three).

If you want to *simulate a tie* with font-based slurs – PMX does not provide true ties with font-based slurs! – include the option `t` in both the starting and ending command for the slur⁴⁴. This will draw a slur with altered endpoint positions.

B 4.12 Dynamic Marks

PMX has a very simple tool for inserting virtually any dynamic mark in a score. Its Dynamics symbol is a `D`, followed by one of 3 types of parameters, optionally followed by position shifts:

1. **A standard dynamic mark** : this can be any one of the following: `pppp`, `ppp`, `pp`, `p`, `mp`, `mf`, `f`, `ff`, `fff`, `ffff`, `sfz`, `fp`.

⁴⁴For reasons of backward compatibility, it is legal to specify a tie with the symbol `st` or `t` even when PostScript ties are activated. In that case, these symbols are fully equivalent to simply using `t`.

2. **A crescendo/diminuendo** : < or >. These crescendo/decrescendo marks – aptly named “hairpins” – are *toggles*, i.e. the first D< starts the crescendo, and the next D< ends it; similarly for D>.

As of version 2.616 **PMX** allows hairpin dynamics to span input blocks!

3. **Arbitrary text** : "...", where ... stands for any text string, e.g. D"molto espressivo". The text will be set in italics unless another T_EX-style font specification is included ⁴⁵.

All dynamics symbols go *after* the note to which they refer (so e.g. `g Dpp` will write a pianissimo *g*). Hairpin marks must be contained *completely within the same input block?*, ⁴⁶

The default position of any dynamic mark or text entered with the D symbol is just below the notehead, stem end, or bottom staff line, whichever of these is lowest. There are numerous context-sensitive automatic adjustments to the positions of all the dynamics symbols. If you don't like the result you can add position shifts to the dynamics symbol: a signed integer for a vertical shift (in `\internotes`), then – optionally – another signed number for a horizontal shift (in notehead widths).

There can be a combination of dynamic marks at a single note ⁴⁷. Since these are distinct **PMX** symbols, they must be separated by spaces and must come in the right order, e.g.

[some notes] D< [more notes] D< Dffff D> [more notes] D>

If you aren't using PostScript slurs, there are some restrictions on hairpins which are due to MusiX_TE_X's use of font-based hairpins: they cannot be longer than 68mm, they cannot wrap over a system break, and they must be horizontal. Finally, only certain specific lengths are available, so some horizontal position tweaking may be needed, especially when standard dynamic marks and hairpins are combined. These restrictions don't apply when using one of the PostScript slur packages; PostScript hairpins will then be used without any further intervention ⁴⁸. Fig. B.27 (a fragment from the *Pathétique* by Tchaikowsky) shows some examples ⁴⁹.

B 4.13 Clef Changes

As was noted in Section B 3, the clef for each instrument is given in the preamble (cf. p. 21). A clef change in mid-stream, however – frequently occurring in piano music –, is signaled by a C followed by a single lower-case letter, as illustrated in Fig. B.28.

⁴⁵A practical example is `D" \ppff fz` (cf. [Tips and Tricks](#)).

⁴⁶If need be (e.g. for crescendi/decrescendi across bar lines), you can adjust the position and length of a crescendo/decrescendo by adding position shifts as described.

⁴⁷It doesn't make sense, of course, to have 2 different standard dynamic marks on the same note, and **PMX** will complain if you try that.

⁴⁸For details on invoking PostScript slurs, ties and hairpins cf. Section B 4.11 ff.

⁴⁹Fig. B.27 was obtained using the PostScript slur package type K.

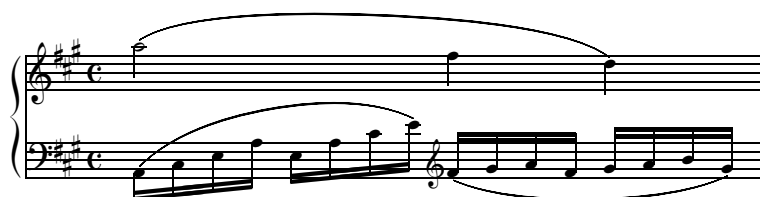


```

% Bar 1
r2 D"Adagio"+16 r4 e8-- Dpp-1 D< s f D< D>-1 s /
% Bar 2
g4 D>-1 s f2 s D<-1 f8 s D<-1 Dp-1 g s D>-2 /
% Bar 3
a4 s D>-2 D<-2 g2 s D<-2 g8 s Dmp-1 D<-1 a s /
% Bars 3-4
b4 D<-1 D>-1 asd2 D>-1 D<-1 /
% Bars 5-6
b0 st D<-1 Dsfz-1 D>-1 | b8 st D>-1 Dp-1 r r4 r2 Rb /
%

```

Figure B.27: Dynamic marks



```

2 1 4 4 0 6 0 3
1 1 20 0

bt
./
Ap
w180m
%
a12 sh c e a e a c e sh
Ct f s g a f g a b g s Rd /
a25 sh f4 d sh /
%

```

Figure B.28: A clef change

When there are two voices in a staff, the clef change command must be given in the *first* (lower) voice; if it is entered in the second voice, PMX will *silently disregard* the clef change completely. The clef change holds for all voices in a staff, of course.

A logical effect is worth mentioning: when, before the clef change, there is a note the duration of which extends beyond the point of the clef change, its clef does not change. Fig. B.29

shows an example of such a situation.

2 Violoncelli



```

c43u Ct a44u cu au //
f02 /
Cb rb4 Ct e4u au eu //
c23l r2-6 /
%
```

Figure B.29: A clef change in a staff with 2 voices

B 4.14 Octaviation

When in a voice the pitch of a number of notes extends far beyond the staff, resulting in an excessive number of ledger lines, readability is usually improved by using an ‘octaviation’ notation: an horizontal line above the staff, starting with an “8” or “8va”, indicates that the notes under this line are to be played an octave higher than printed, and similarly “8 bassa” or “8va bassa” indicate an octave lower.

In the present version, **PMX** does not provide a notation for this, so you will have to take recourse to inline \TeX coding. For an example see Section E.

B 4.15 Figured bass (basso continuo)

PMX lets you introduce the basso continuo notation (‘figured bass’) very simply, as illustrated in the example of Fig. B.30, which is the basso continuo of an excerpt of the famous aria “*Pur ti miro*” that concludes the Monteverdi opera *L’Incoronazione di Poppea*. Here are the rules:

- Figure symbols are entered – as plain numbers⁵⁰ – *after* their associated note symbols, with a space, as usual. As of **PMX** version 2.517, they are allowed in two staves, the first (lowest) staff and any other one. Enter the characters as they would appear from *top to bottom*, e.g. 64 [6 over 4], or 642 [6 over 4 over 2].
- Accidentals that modify a number *must* be entered *before* the number. Note that the characters that signify accidentals are different here than for notes: flats are written as ‘-’ (minus), sharps as #, and naturals as n. For example, *sharp third* is #3 (or simply #, in the standard shorthand), *six (over) flat five* is 6-5, and *sharp six (over) 4* is #64.

⁵⁰As of **PMX** version 2.520, the numbers usually needed in figured notation, namely 2, 4, 5, 6 and 9, are available in a different graphic design with perhaps a more baroque-like look.

If you want to use these, add an ‘s’ after the **PMX** string of every figure which you want to appear in that form.

Note that for this to work properly, you must have the special fonts for these numbers installed. You will find **figbas.zip**, which provides these fonts, in the **WIMA** software page. **figbas.zip** contains 2 files that can be of help with the installation: **README** and **test.tex**.

- For each staff with figures, **PMX** positions all the figures within each system below the staff with their tops at the same level. If you want to change this vertical position default, you have two possibilities:

to lower a figure symbol: prefix the figure symbol – *no space!* – with one or several ‘_’ (underscore). Each underscore will lower the figure symbol by 4 lengths of `\internote` ,

to raise a figure symbol : *append* the figure symbol – *again no space* – with a ‘+’ (plus) followed by an integral number for the number of `\internote` lengths by which to raise the figure.

The two options can be combined to provide full control over the vertical position of the figure symbol.

- If you want a figure symbol to align horizontally in the second tier, insert a ‘_’ (underscore) as placeholder for the top tier, before the one you want to go on the second tier (cf the sharp in bar 16 of Fig. B.30).
- Sometimes you may need to enter a figure when there’s no bass note sounding. To do this, *precede* the figure symbol with three additional characters, `x[n][m]`. Here *n* is a single-digit repeat count, and *m* is a single-digit duration time value, i.e., 0, 2, 4, 8, 1 or 3 . This will offset the figure from the associated note by the specified duration 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 (cf. Fig. B.30, bars 2, 4, 6, 8, 10 and 14).
- There is also a *continuation* symbol, viz. a 0 (zero) followed immediately by an unsigned number. 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 (cf. Fig. B.30, bars 9, 11, 16 and 17). The height and length of the line are set by the current note’s level and `\noteskip` respectively⁵¹. These can be mixed in with other figures to produce vertical stacks. If another figure follows in the same symbol, use : as a separator. For example, a continuation line over a 3 would be coded as `01:3` .

If there are figured bass commands in a **PMX** file, but you want them all to be ignored, then enter the symbol **F** in the header. This feature is most useful in the form `%1F` (cf. Sec. C 1, which makes a separate bass part with no figures, e.g. for a violoncello part.

Figured bass symbols will not be altered in any way under transposition by **PMX** (cf. Sect. B 5.5). There is no universal set of interpretations of figured bass symbols, so no automatic transposition is possible.

PMX does not admit 2-digit basso continuo figures. This is the notation of some old editions; if you want to use it, you will have to resort to inline `TEX` coding (cf. Section B 8). There is an example for this in the Caccini aria in the Appendix (Sec. G 2.3).

⁵¹If `\noteskip` changes, or if an unfigured 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 automatically join them together at the lower height (thanks to Werner Icking for this idea).

For an explanation of the special **PMX** coding in Bar 16 of Fig. B.30 see Section E .

The image shows four systems of musical notation for figured bass. Each system consists of a staff with notes and a line of figures below. The first system (bars 1-4) has figures: 6, 5, 6 4, 3, 6, 6, 4, 3. The second system (bars 5-8) has figures: 6, 6 4, 3, 6, 6 4, 3. The third system (bars 9-12) has figures: 6, 7, 6, 4, #. The fourth system (bars 13-15) has figures: 4, #, 6 5, 4, #.

```

% bars 1-4
gd2 fd 6 | ed 5 x126 dd 4 x123 | gd fd 6 | ed 6 dd 4 x123 | /

% bars 5-8
gd2 fd 6 | ed 64 x123 dd | gd fd 6 | ed 64 x123 dd | /

% bars 9-12
gd2 02 fd | ed 6 dd 7 | gd fd 6 | gd ad 4 x12# | /

% bars 13-15
dd- gd | gd 03 fd | ed dd 4 x12# | /

% batt. 16-17
cd- dd 64 x145 x12_00.2 x12_# | gd fd 01 Rd /
%
```

Figure B.30: Figured bass in **C. Monteverdi**, *L'Incoronazione di Poppea*, aria “*Pur ti miro*”

B 5 Commands That Affect All Voices

Unless indicated otherwise, the commands dealt with in this section affect all staves in a score. They must be entered only *in the first (lowest) voice in the first (lowest) staff*.

Such commands will automatically be transferred from score to parts when separate parts are generated by **scor2prt** (cf. Section **C 1**).

b	thin (single bar) line
d	thin-thin double bar line
D	thin-thick Double bar line
z	invisible bar
l	left repeat
r	right repeat
lr	left-right repeat
dl	thin-thin double bar followed by left repeat

Table B.7: Parameters of the bar symbol R

B 5.1 Single bars, Double bars, Repeats etc.

In keeping with general typesetting practice, **PMX** will by default produce

- a single bar line at the beginning of each system after the first, unless there is only one staff per system,
- a single bar line at the end of each bar except the last one in a movement or the entire score,
- the common ending bar line (thin-thick double bar line) as the last bar line of a movement or the entire score.

The user can, however, override these defaults – except the first one – by a bar symbol, viz. an **R** followed by one of the parameters given in Table B.7. If you are going to make parts from your score (cf. Section C 1), you *must* place bar symbols either before the first note in an input block or after the last one; otherwise **scor2prt** may behave erratically. Using two bar symbols in succession doesn't make sense and will produce unpredictable results.⁵²

Here are further comments on some particular bar symbols:

- **Rb** forces a single bar before a movement break, where the default would be a double bar. This can be useful, for example if you change the number of instruments, which **PMX** will allow only at a movement break, but you don't want it to *look* like a movement break.
- **Rz** will cause an invisible bar line at the end of the current system. It can be used together with blind meter changes (cf. Section B 5.3) when you want to split a bar across a system break.
- if a left-right repeat (**Rlr**) comes at a system break, **PMX** will automatically split it in two.

⁵²If you are using | marks at the end of bars, a repeat must always go *before* the |, or the next line must follow in the same input block. In other words: while **Rr | /** will not result in an error, it will produce a thin bar line only.

There is a simple consequence: avoid using | altogether, at least in connection with **R** symbols.

- `Rd1` is the same as a left repeat (`R1`) except at a system break: there the first system is ended with two thin lines, and the left repeat sign goes to the beginning of the next system.
- Finally, a general remark is in order regarding the `R` symbols that designate repeats, double bars etc. (`R1`, `Rr`, `Rlr`, `Rd`, `RD`, `Rd1`, `Rb`, `Rz`). To avoid conflicts with `scor2part`, you should *always* place any `R` command before the first note in an input block. This is always possible except at the very end of the whole piece. Otherwise, there is always a following input block where the command can (and should) be placed.

Fig. B.31 gives some examples.

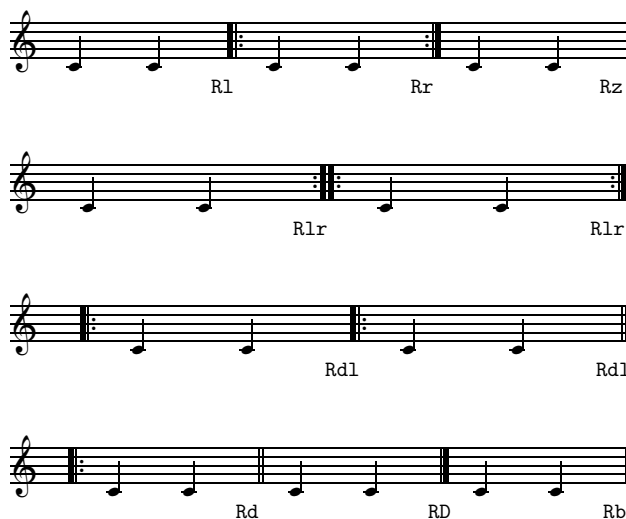


Figure B.31: Single bars, double bars, repeats

On some special occasions, you may want dashed or dotted barlines instead of the regular ones. For this and similar purposes, there is a very good and easy-to-use package by Rainer Dunker: `musixdbr.tex`, included in the M_usiX_T_EX 1.15 distribution.

B 5.2 Volta

The two versions of the ending of a repeated section of music, called ‘volte’ in musical parlance⁵³, are denoted by a symbol starting with a `V` (for **V**olta) in **PMX**. **PMX** needs to know where each volta starts and ends, how the ends are shaped, and what text is to be included under each volta. A string parameter to the `V` is used to convey this information:

⁵³The term ‘volta’ is, of course, Italian, and ‘volte’ is the Italian plural of ‘volta’.

- to signal the *start* of the first volta, add a text string (usually just a 1) that doesn't start with one of the characters `b`, `x`, or any literal spaces⁵⁴. `PMX` will print append a period to the text string,
- to signal the *end* of a volta and the start of the second volta, use a `V` with a text string that *does* start with either the character `b` or `x`. If `b` is present, a volta symbol ending with a vertical stroke will be printed (`box`), while `x` will print the ending *without* a vertical stroke (no `box`)⁵⁵.

Often composers will write out only the first ending of a repeated section of music, and simply continue play, dropping the first ending after the repeat. With `PMX`, the notation is analogous: simply end the first volta, with either `Vb` or `Vx`, without adding a second volta.

On the other hand, if one volta starts where another one ends, the symbols ending the first and starting the second volta can be combined into one, e.g. `Vb2`.

Other rules governing volte are:

- Volte should be placed at the beginning of an input block, in the first (lowest) voice of the first (lowest) staff, before the first note if it's the start of a volta, or after the last note if it's the end of a volta.
- If you will be making parts from the score using `scor2prt`, then to ensure that the volte are properly transferred to the parts, you must only include one volta symbol `V` in each input block, and it must come at the beginning of the block.
- If a score ends while a volta is still open, `PMX` will close it with a box.

Examples of volta usage can be seen in Fig. B.32.

B 5.3 Meter Changes

The meter of a piece can only be changed at the beginning of an input block, and thus naturally only at the beginning of a new bar, just before the beginning of the first (lowest) voice of the first (lowest) staff. As with the initial meter specified in the preamble, any new meter applies to all instruments and voices⁵⁶.

A meter change symbol starts with the letter `m`. There are two different ways to complete the symbol:

- enter the 4 meter-defining numbers `mtrnuml`, `mtrdenl`, `mtrnump`, `mtrdenp` for the new meter⁵⁷, separated by slashes `/`.

⁵⁴Actually you can include a space by using the `TEX` space symbol `~` instead of a blank.

⁵⁵Note that the characters `b` or `x` can appear anywhere in the text string and will *not appear in print*.

⁵⁶As was pointed out before (cf. p.30), this limits the use of `PMX` for modern polyrhythmic music. But you can always try to play with tricks...

⁵⁷For the meaning of these numbers cf. p.18.

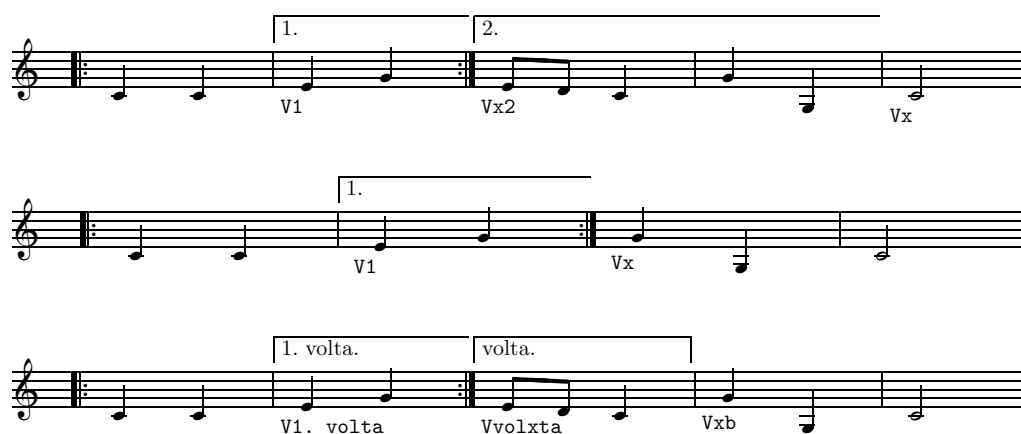


Figure B.32: Volte

- same as above, but enter the 4 numbers consecutively *without spaces*. In this case, it is necessary to distinguish between one single 2-digit number and two consecutive 1-digit numbers. For this – and for this case only – the following convention is adopted:
 - the number 1 is represented by the letter o,
 - consecutive digits 10,... 19 stand for exactly that: the 2-digit numbers 10 through 19.

Thus, 19 is the largest number that can be entered with this method.

Note that `mtrdenl=0` still represents a whole note (semibreve), as explained on p.18, and a ‘blind’ meter is given by `mtrnump=mtrdenp=0`, as explained in Table B.3.

The most common application of a ‘blind’ meter change occurs when a piece — often one starting with a pickup — has an incomplete final bar. In such cases 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 of /`.

Another useful application is for the pickup bar in the second or later movements, following a **PMX** movement break symbol. Since there is no special provision for pickups in this case like there is at the beginning of a piece,

1. the pickup bar should be in a block by itself, starting with a meter change symbol with a logical meter representing the pickup bar and a printed meter as appropriate, and
2. the next bar should start a new block and begin with a blind meter change symbol with a logical meter representing the true meter.

B 5.4 Key Changes

A key change can be signalled at any time⁵⁸. It must be entered in the first voice, but will affect all voices. Use the command `K+0`, followed by the new key signature: positive integer for sharps, negative for flats (cf. Section B 3.1) .

Some examples are given in Fig. B.33.

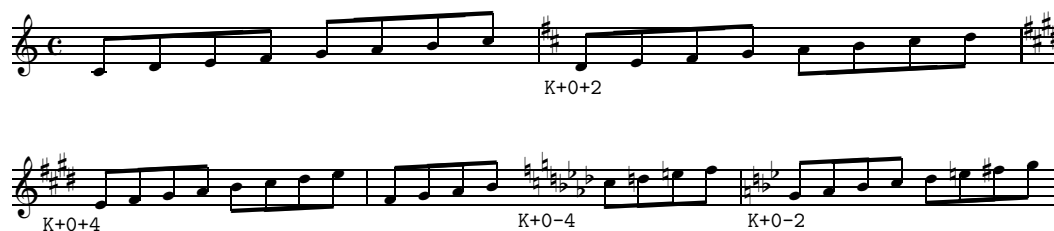


Figure B.33: Some key changes

B 5.5 Transpositions

B 5.5.1 Transposition of an entire score

To transpose an entire score to a different key from the one in which the data have been entered, use the symbol `K` (for **Key**), followed by two explicitly signed digits: (1) the distance to transpose by, in units of `\internote` (i.e. the vertical offset on the staff), and (2) the new key signature. This symbol must be entered at the beginning of the piece, in the header (cf. p.22).

There is one case that requires special handling, viz., when transposing by a half step, but the vertical position doesn't change, e.g. from D flat to D. In this case the shift parameter (the first parameter of `K`) is *always* `-0` regardless of whether the transposition is up or down a half step⁵⁹; the direction of the transposition will be determined by the new key signature, the second parameter. An example of transposition is given in Figs. B.34.

When transposing, one should always use relative accidentals, activated by the symbol `Ar` in the header (cf. Section B 6.1 for a detailed explanation of this notation).

When transposing an entire score of same-type instruments, it is of course easy to obtain both the original score and its actual sound, as shown in Fig. B.35.

B 5.5.2 Transposition of Individual staves


In orchestral scores, you usually have individual 'transposing instruments', i.e. instruments that are denoted (and, in fact, *played*) in a pitch different from what they sound: clarinets in E or B flat, Oboes in A (oboe d'amore) or F (oboe da caccia), and similar. For that purpose `PMX` now provides the option to transpose individual staves.

⁵⁸Only one key change is allowed in each input block.

⁵⁹This artifact is needed to differentiate a half-note transposition from a mere key change without transposition (cf. Section B 5.4) .


no transposition:

d8 e1 f g a bs cs d8 cn1 bn a g f e | d0 Rb /




transposition by one tone up:

K+1+1 d8 e1 f g a bs cs d8 cn1 bn a g f e | d0 Rb /



transposition by a half tone up:

K-0+6 d8 e1 f g a bs cs d8 cn1 bn a g f e | d0 Rb /



transposition by a half tone down:

K-1+4 d8 e1 f g a bs cs d8 cn1 bn a g f e | d0 Rb /




Figure B.34: Some transpositions of a d minor scale

The instructions given by Don Simons himself on introduction of **PMX 2.613** are so clear that it is best to quote them literally here:

“The syntax is an extension of the existing full-score transpose/key change command ”K”. Just follow ”K” with ”i”, then instrument number, then the same parameters as before for transposition amount and new key signature:

`Ki[instrument #] [+/-] [trans amt.] [+/-] [new key]`

For more than one instrument, you may immediately repeat everything after ”K”. The command must either be at the start of the score (right after setup), or if later, must be preceded by a normal (full score) key change command `K+0[+/-][new key]`. (Remember that later full-score transpositions are not allowed). ”

An important function is worth repeating/emphasizing:

- It is usually easiest to put the new K command where the old one was: right before the start of the actual score, i.e. in the ‘header’. But it is now also possible to enter it later, albeit with the (somewhat cumbersome) addition of a ‘dummy’ full-score key change command (as explained above).

This can indeed occur in practice, e.g. when the oboe player switches to English Horn in midstream!

B 5.6 Titles, and text above and below a system

A *title block* with up to three elements can be defined in the header (cf. p.22):

Oboe da caccia solo, on stage

Standard notation in the original score

Oboe da caccia solo, on stage

Actual sound (transposed with K-4-4)

Figure B.35: **R. Wagner**, *Tristan und Isolde*, beginning of third act

- **Ti**: an instrument name. It is set *above* the title, and left-justified,
- **Tc**: the composer's name. It is set *below* the title and right-justified,
- **Tt**: a title for the whole piece. It is centered on the page.

Each of these commands is to be followed by a text string, *on a separate line*. This text must be fully contained in one line of **PMX** coding; but the text for the title can be spread out over several lines in print by using the standard \TeX line break symbol $\backslash\backslash$.

Extra vertical space can be added between the whole title block and the top system by appending to **Tt** (without space) a one- or two-digit number representing this space, in units of $\backslash\text{internote}$ ⁶⁰. This works only if **Tt** is the *final* title block element entered. Thus, although the 3 parts of a title block can be given in any order, it is good practice to use the sequence **Ti** — **Tc** — **Tt**.

When you want to generate parts from a score with **scor2prt** (cf. Section C 1), you should *omit* **Ti** with an associated instrument name, because each separate instrument usually has a different name. Rather, **scor2prt** will automatically supply the appropriate instrument names, printing the name of each instrument at the upper left of the first page of each part; the names used here are the ones specified in the preamble of the **PMX** file for the score⁶¹. If you do have **Ti** together with an instrument name in the main score, **scor2prt** will print that name identically to all the parts — which is usually not what you want!

⁶⁰**PMX** does this together with the adjustment of other vertical spaces when fixing the general page layout. In some rare cases, the result may not be quite what you expect, in particular when you haven't used the **Ae** option to equalize inter-system spacing (cf. section B 6). In such cases you might want to use inline \TeX — take a look at the example in Section E 2.1 for how this can be done.

The symbols **h** (for **header**)⁶² and **l** (for **lower**) can be used to introduce text either above or below a system. These symbols must be placed in the first column of an input line (not necessarily the beginning of a block), and followed by a blank or – for **h** only – a signed integer. In the latter case, this integer is a vertical shift, in units of `\internote`.

The text string, which again must lie on a line of its own immediately following the symbol, will be printed above or below the *top* staff in the *first* bar of the block where it is entered. Figs. B.36 and B.1 show examples.

B 5.7 Page numbering and page headers

B 5.7.1 Page numbering

By default, **PMX** does *not* number the pages of a score, even when it has more than one page. You can, however, switch on page numbering by using the symbol **P** (for **P**age). Placed at the beginning of an input block anywhere within the **PMX** body of input, and followed optionally by a page number and/or by **l** (for **l**eft) or **r** (for **r**ight), this will

1. start page numbering at the current page⁶³ with the number given as optional argument. Further page numbers are then incremented automatically starting with this number. If **P** has no number argument, the default is page 1. Thus, **P** and **P1** are equivalent),
2. put the page numbers to the top left or right of the first of the pages to be numbered, after which the number positions will alternate. If no **l** or **r** is given, **PMX** puts the page numbers on the right on odd-numbered pages, on the left of even pages.

You can have as many **P** symbols in a score as you like; but there is no command to switch off page numbering again once it has been switched on.

B 5.7.2 page headers

There is one more option, **c** (for **c**entered header), available with the **P** symbol. This option defines a text string to be printed at the top of every page *after the first*. The notation is as follows:

- The normal usage is `Pc"[text string]"`. The text string enclosed in the quotation marks must be a single line of text. It may contain blanks, but *no* `\\`). If the string consists of blanks only, that is what will be printed: a blank page header.
- If the text string does not contain any blanks, you may omit the quotation marks, but you must then start the text immediately after the **c** *with no space*.

⁶¹`scor2prt` does this by inserting into the part `.pmx` file a `Ti` and the appropriate instrument name as taken from the preamble of the main score. Knowing this may be useful if you want to manipulate this process, e.g. to change the vertical shifting of a particular instrument name (cf. Section E 2.1 for details).

⁶²This usage of the **h** symbol is not to be confused with that for the page size (see Section B 6.2)! Although both can occur together in the header, they are in fact, different **PMX** symbols, distinguished by their differing syntax.

⁶³The ‘current page’ is the page where **PMX** puts the music currently processed at the point where it encounters the symbol **P**.

- If there is a blank after Pc, *no quotation marks, and no further text*, the text printed as the header will be the instrument name entered with the symbol Ti, as described on p.[B 5.6](#).

c must be the *last* option in the P symbol.

The P symbol and all its options will be ignored when making parts from a score using **scor2prt**, since page numbering will usually be different in the score from that in the parts. Page numbering – and centered headers – for parts can, however, be initiated independently with another mechanism. For the details on this, refer to Section [C 1](#).

B 5.8 Layout: line, page, and movement breaks

The parameters `npages` and `nsystems` of the preamble (cf. Section B 3) determine the layout of the score, either by fixing the total number of pages and systems, or by setting the average number of bars per system. With no further instructions, **PMX** will attempt to distribute all the music evenly over the total number of systems, and then spread the systems evenly over the specified number of pages, either specified or determined by default (with `npages=0`).

You can change the average ‘crowdedness’ by changing the preamble parameters. Once this is satisfactory, you can exercise finer, more local control by forcing, i.e. manually inserting, line or page breaks at exactly the places you want them to be. You may also want movement breaks, and **PMX** has a convenient way of doing that, in the process dealing easily with such ‘housecleaning’ chores as indenting the first system of the new movement, reprinting the time signature, resetting the bar number counter, and other details to be discussed below.

It is worthwhile, however, to quote Leslie Lamport at this point:

“Don’t worry about line and page breaks until you prepare the absolutely final version”

From: Leslie Lamport, *L^AT_EX, user’s Guide and Reference Manual* .

This is especially true for a **PMX**-generated score. Until the final edit, you should always use `npages=0`. Once you specify any forced line break, you can no longer use `npages=0`, and you will have to specify values for `npages` and `nsystems` . And because — as we’ll soon explain — a movement break is a sub-option of the `linebreak` command, you cannot set movement breaks either until the final edit.

But unless your score is just a few pages long, you probably will want to set some line, page, or movement breaks in the end. When you are ready, always first pick some reasonable values for `npages` and `nsystems`—perhaps the numbers that finally came out when you used `npages=0` during the main input process. Then one way to proceed is to start at the beginning, forcing breaks where desired. The most common places would be at movement breaks, and at places associated with *volte* or repeats. Recompile and view the result after each new line break is inserted. You may need to adjust the total number of systems or even pages depending on how crowded is the remainder of the score, after the last forced line break.

The other way to proceed is to begin by inserting any obvious page breaks, especially if you are laying out a booklet with facing pages and want page turns to come in musically convenient places.

Here are the rules for inserting forced line, page, and movement breaks:

- Line breaks can only come at the start of an input block. To force a line break at the start of the n^{th} system, enter `L[n]` . n must obviously be greater than 1 and less than or equal to the value of `nsystems` as specified in the preamble.
- Page breaks can only come where there is already a line break. To force a page break at the start of the m^{th} page and the n^{th} system, enter `L[n]Pm` . m must obviously be greater than 1 and less than or equal to the total number of pages.

- Movement breaks can only come where there is already a line break. To force a movement break at the start of the n^{th} system, enter `L[n]M`. If in addition you want a page break at that point, use `L[n]P[m]M`.

The movement break symbol `M` has several optional parameters. They can be used in any combination and should follow with no spaces.

- `M+[k]` adds k `\internote` units of vertical space between movements.
- `Mi[.x]` resets the first-line indentation of the new movement to `fracindent=.x`, where $.x$ is some decimal number (cf. Sec. B 3).

You must write the `fracindent` number exactly in the form $.x$, i.e. *without* the leading 0. If, e.g., you write 0.2 instead of $.2$, **PMX** will complain with an error message, asking for a decimal number!

- Beginning a new movement break, the bar number is reset to zero by default, whereas `Mc` continues bar numbering.
- `M[n]` changes the number of instruments to n . n must not exceed `ninstr` as set in the preamble. If n exceeds 9, it must be *preceded* with `:` (colon). This should be followed (without blanks) by a sequence of n instrument numbers in bottom-up order, again preceded with `:` if bigger than 9, then a sequence of clef symbols, one for each staff of each instrument starting from the bottom. An instrument's numeral is simply its position in the original sequence (in the preamble); these numerals can be permuted as desired, but the sequence of clef symbols should, of course, match the new sequence of instruments.
- `Mr+/Mr-` reprints/suppresses reprinting the instrument names at the beginning of the new movement. The default is to print them only if the number of instruments changes.

It should be emphasized here that the number of instruments in a new movement can never exceed the original number of instruments, although it is permissible to increase it after it has been decreased, as long as it doesn't exceed the original number; in other words, and to give an example: a sequence quartet — solo — duet, e.g., is possible, while the reverse sequence is not. There are two ways to circumvent this restriction:

1. start the score with a dummy page — to be discarded later — containing the maximum number of instruments. On the second page, start a new movement with the desired number and sequence of instruments for the first movement,
2. make separate **PMX** files and concatenate them afterwards (cf. Section B 1.1 or C 4 on how to do this).

Immediately after a movement break, any desired meter changes, key changes, or text can be entered as already described in Sections B 5.3, B 5.4 and B 5.6 (p.59 ff.).

The effect of movement breaks is illustrated in the example G 2.1 in the appendix.

B 5.9 Bar Numbering

By default **PMX** places a bar number above the first bar of the top staff in every system, and there is no provision in **PMX** to change this default. But it is easy to change this by using a straight MusiX_{TEX} command:

- if you want no printed bar numbers at all, add a line in the header containing `\\nobarnumbers\`
- if you want bar numbers printed over every bar, add a line containing `\\barnumbers\`

in the header or at the beginning of an input block.

This is an example of a **type 2** inline \TeX command, cf. Section B 8. As is explained there in detail, **PMX** will place a **type 2** inline \TeX command at the beginning of the `.tex` file produced, regardless of where in the `.pmx` file the command appears. This means that you cannot change this type of bar numbering in mid-stream.

If you do want to change the bar numbering somewhere in the middle of a score, use a **type 3** inline \TeX command: if, from the current point in the score, you want

- no printed bar numbers, add a line with `\\nobarnumbers\`
- bar numbers printed over every bar, add `\\barnumbers\`
- bar numbers above the first bar of every system (the **PMX** default), add `\\systemnumbers\`

at the beginning of the current input block.

Furthermore, you might want to change the number itself. This occurs most frequently when you have a pickup bar at the beginning of a repeat; some – including some commercial publishers – do not include this pickup bar in the number count, whereas **PMX** does so by default. To adjust this, use the **type 3** inline \TeX command `\\advance\barno-1\relax\`.

MusiX_{TEX} provides still more elaborate bar numbering schemes, such as a number over every n^{th} bar. For details refer to the MusiX_{TEX} 1.15 manual.

B 6 Some general options and technical adjustments

B 6.1 Global options

PMX has many layout parameters with default settings. Some of these can be changed by the user; for that purpose there are several symbols, all beginning with **A**. Many of these are usually included in the header, but some may be entered at the beginning of any input block. Table B.8 lists these options alphabetically; the following subsections, grouped by types of settings, describe them in detail. When using several of these commands, it is not necessary to enter them all on separate lines; rather, you can concatenate them, in any order, with just a single **A**, e.g. `AdI2.3p+h1br`.

Aa[<i>x</i>]	sets space before first note in a bar
Ab	makes accidentals big
Ac[<i>x</i>]	sets page sizes and offsets according to paper type
Ad	puts dots in lower voices <i>below</i> the line
Ae	equalizes inter-system spacing
AI[<i>x</i>]	changes default Interstaff spacing <i>for the whole score</i>
Ai[<i>x</i>]	changes default interstaff spacing <i>for the current page only</i>
AK	adjusts rest heights in 2-voice staves depending on context
AN[<i>i</i>] “ <i>name</i> ”	sets filenames generated by scor2prt (cf. Section C 1)
Ap[<i>further suboptions</i>] (cf. Section B 4.7)	enables PostScript type K slurs
Ar	switches accidentals to relative
AR[<i>file</i>]	inserts a text file in the PMX source file (cf. Section B 1.1)
AS[<i>ns</i>]	enables PMX Spacing algorithms to small-font staves
As	makes accidentals small
AT	activates special xtuplet brackets (cf. Section B 4.6)
Av	spreads systems vertically over all of an unfilled page when ‘on’

Table B.8: Symbols beginning with an A (global options)

B 6.1.1 Accidentals

By default, big accidentals are used unless regular spacing doesn’t provide enough room. Thus the default behavior may cause a mixture of big and small accidentals, and in fact is not recommended. Rather, the user is advised to make his choice:

- **Ab** makes all accidentals **big**; this is usually preferred,
- **As** makes all accidentals **small**.

As was mentioned earlier (cf. Section B 5.5), **PMX** by default uses the so-called ‘absolute’ notation for inputting accidentals, i.e. the notation explained in Table B.5. In this notation, there is an obvious one-to-one correspondence between accented characters in the **PMX** source file (**s**, **f**, **n**, **ss**, **ff**) and the printed characters. This has a disadvantage, but it only surfaces if you write the **PMX** file in one key and later transpose it using the **K** option described in Section B 5.5. For example, suppose the original key is F, with 1 **b** in the signature, and you enter **bn**. Now if you transpose to the key of C with **K-3+0**, that note will be printed explicitly as F natural, while it should be F sharp.

The remedy for such problems is to use another notation, aptly named ‘relative’ accidental notation, used by some musicians and some publishers in some countries. In this notation, a sharp, flat, or natural sign denotes an alteration by a half-tone up, down, or none *relative to the note which would otherwise be indicated based on the signature of the piece*. For example, in D major, with 2 **♯**s in the signature, an F natural would be denoted not as **fn** but rather as **ff**, while in d minor, with one **b** in the signature, a B natural would be input not as **bn** but

rather as **bs**. Similarly, **bs** accidentals refer to the notes prescribed by the signature; e.g., in G Major (1 \sharp in the key) a **gn** sounds as a *g*, while **fn** sounds as an F sharp.

For automatic transpositions to function properly, it is necessary that the **PMX** source for the score to be transposed use the *relative accidental notation*. This is effected by **Ar** :

- **Ar** directs **PMX** to interpret accidentals in *relative* notation⁶⁴.

B 6.1.2 General layout

The esthetic appearance of a score of music depends on the white space around it, and thus depends on the physical size of the paper.

In **PMX**, default values for this can be set with a global option **Ac1** or **Ac4**, with no further adjustments required in **dvips** . These commands set horizontal and vertical sizes and offsets that center the page:

- **Ac1** for letter paper
- **Ac4** for DIN A4 paper .

These 2 global options are available as of **PMX** version 2.618 , not before !!

B 6.1.3 Layout details

When the note head of a dotted note is *on* the line (e.g. for a *g* in a violin clef), it is customary to place the dot slightly *above* the line. In staves with two voices, there is an alternate custom, viz., to place the dot *above* the line in the *upper* voice and *below* the line in the *lower* voice. **PMX**'s default is the first possibility; dots for both voices *above* the line, but it can be overridden:

By default **PMX** inserts a horizontal gap of `1\elemskip` between a bar line and the first note in the bar. This can be changed globally:

- **Aa**[*x*] sets the space before the first note in every bar to to *x* units of `\elemskip` .

In **PMX** it's not yet possible to specify a smaller font for selected staves. But it can be done using an inline \TeX command (For details, cf. Section B 8 and Section E 2.4). If you do this, then you ought to use the **AS** option:

- **AS**[*ns*] . The additional sequence of *ns* is mandatory here. It consists of exactly as many characters - (minus) or 0 (zero) as there are staves in the score (**nstaves**, cf. Section B 3): 0 if the font size of this staff is normal, - if it is small⁶⁵. This tells **PMX** to modify some horizontal spacing decisions to account for the smaller font size.

⁶⁴Note that this makes logical sense only if used in the header, and accordingly there is no way to 'undo' this decision further down in a score.

⁶⁵The sequence of staves is from bottom to top, as described in the preamble. Cf. p.22.

B 6.1.4 Vertical spacing

a) Spacing of staves within a system:

If there is more than one staff in a system, **PMX** computes the vertical spacing between the staves automatically. However, the algorithm isn't fully robust, and the result may not be pleasing. To change the spacing between the staves within a system, you can apply a scale factor x to the default, either for the entire score, or for the current page only:

- **AI**[x] multiplies the default **Interstaff** spacing, `\interstaff`, by the decimal number x for the entire score. This option should be placed in the header.
- **Ai**[x] multiplies the default **interstaff** spacing by the decimal number x for the current page only. This option can be placed at the beginning of any input block (including the first), and overrides **AI**[x].

Note that if there is more than one system on a page, decreasing the interstaff spacing will increase the space between systems, and vice-versa.

b) Spacing of systems on a page:

MusiX_{TEX} normally draws a virtual box around each system and inserts equal vertical space between these boxes. When objects protrude above the top staff in a system – such as the note `c46` in the violin clef – or below the bottom one, this can lead to unequal spacing between the bottom staff line of one system and the top staff line of the next. You may prefer that this vertical spacing be constant for the whole page:

- **Ae** ensures that the spacing between the bottom staff line of one system and the top staff line of the next is constant for any one page⁶⁶.

c) Sparsely filled pages:

Sometimes it may happen that a page contains just a few staves altogether, and then the white space between staves on such a page becomes excessive. In such cases – specifically, if `\interstaff > 20\internote` – **PMX** will, instead of the equal spacing described above, group all systems near the top of the page. This can, however, be changed by the user:

- **Av** will suppress the grouping near the top, and ensure that systems will always be spread vertically regardless of how much white space is left between systems.

Note that **Av** acts as a toggle; the second time it is issued, the behavior will revert to the default.

d) Vertical position of rests in 2-voice staves:

- **AK** activates special rules for vertical positioning of rests in two-voice staves.

This general option is rather subtle; its explanation by Don Simons follows:

⁶⁶It is generally advised to use the **Ae** option by default, unless one has a specific reason not to. When using this option, you may Nevertheless, in certain cases you may want to force more vertical space between certain systems. There is a `TeX` macro, called `\spread[x]`, that can be inserted anywhere in the system before the desired wider gap. Its argument x is the desired extra space, in units of `\internote` (cf. Section B 8).

“Without this option, rests in two-voice staves have default positions based on a simple rule that is not context-sensitive: those in the lower voice (the one before //) are 4 \internote heights below their single-voice default positions, and those in the upper voice are 2 \internote heights above the single-voice default.

“The AK option invokes a set of context-sensitive rules to set the default position. The baseline rule is to align the rest in a horizontal line with the next following note in the same bar. If there is no following note in the bar, then the rest is aligned with the last note before the rest.

“If there are simultaneous rests in both voices, the old rule is applied.

“The AK option is a toggle: it may be used at the start of any input block to turn these special rules on (if previously off) or off (if previously on).

“When the AK option is on, it only affects places where there are two voices in a staff.

“Any user-defined tweaks on the height of a rest will supersede the option for that particular rest, i.e. the tweak will be applied relative to the single-voice default position.”

Sometimes you may want the rest to be aligned with the *previous* note (the one to the left of the rest), rather than the following note (the one to the right of the rest). This can be achieved simply by appending an L (for ‘Left’) to the rest symbol, i.e. by writing ‘rL’.

B 6.1.5 PostScript type K slurs, ties and hairpins

The use of PostScript slurs, ties and hairpins circumvents most of the shortcomings of the font-based counterparts. Of the two available packages, only the type K slurs are explicitly supported by **PMX**. There is an option, **Ap**, to enable the use of this package.

- **Ap** activate Type K PostScript slurs, ties and hairpins.

Ap may also be entered at the beginning of any input block to activate certain suboptions controlling shapes and positions of the slurs and ties that are to be used. The first few of these options affect vertical positioning. When Type K PostScript slurs are activated with **Ap**, slurs and ties will by default *not* have their vertical positions tweaked to avoid tangencies with staff lines.

To control this type of adjustment, use one of the following suboptions to **Ap** :

1. – **+s** activates automatic slur height adjustment,
– **+t** activates automatic tie height adjustment,
– **-s/-t** deactivate the corresponding height adjustment.
2. A second suboption, **+c/-c**, will increase or decrease the default curvature of the Type K slurs. The result of such commands is cumulative, and more than one suboption **+c** or **-c** may be used in a single command. If the cumulative change goes

above **HH** or below **f**, a warning will be issued, the default will be set to **HH** or **f**, and processing will continue.

Local curvature options in individual slur commands (cf. Sec. B 6.1.5) will take precedence over the global default, but will *not change the default*.

A special local option, **n** (‘normal’), is of the ‘undo’ type: it will cause that particular slur, no matter what the global default happens to be, to have the normal curvature, i.e. between **f** and **h**.

3. A third suboption, **l**, changes the appearance of slurs and ties that extend across line breaks:
 - Every slur/tie at a line break is automatically broken into two separate ones; no additional **PMX** slur start or ending commands are required. Vertical and horizontal tweaks for the end of the first segment and Start of the second segment are entered as options in the normal command that starts the slur/tie:
 - (a) the option for the end of segment 1 starts with **s** (for **sever** or **split**), then the usual one or two signed numbers, then a second **s** and one or two more signed numbers for the start of segment 2,
 - (b) the usual curvature options **h**, **H**, **HH**, **f**, if included in the starting command for a line-break slur, will apply to segment 1, and to segment 2 if in the closing command. If the slur/tie does not come at a line break, the special position tweaks (those after the **s** option) will all be ignored, and the curvature tweaks on the closing note take precedence.

The **l** option can also be invoked globally (i.e. for the entire score) by specifying **Ap1**, instead of only **Ap**, in the header (recommended).

The third suboption concerns only line-break ties (it does not affect slurs):

- **+h/-h** activates/deactivates the use of special **half**-ties for the second segment of line-break ties after the line break. These are horizontal at their left end, and are only used if the required segment is shorter than 15 pt.

Note that this option is somewhat incompatible with the **l** option, and they should not be used simultaneously.

B 6.2 Page Size

The default page size⁶⁷ is 740 pt × 524 pt (10.3” × 7.3”, or 261 mm × 185 mm). To change the height or width, use the special symbols **h**[*n*][*u*] or **w**[*n*][*u*] in the header. Here *n* is a decimal number for the new size, and *u* defines the units: **i** for inches, **m** for millimeters, and **p** for points. if no unit name *u* is given, **PMX** uses points as the default.

This command can be used together with **%** or **%!** (see Section C 1) to give the parts made by **scor2prt** different page sizes than the parent score.

⁶⁷This means the *extent of the print* on the page. The – empty – top, bottom, left and right margins depend on the paper/printer used; they can not be altered within **PMX**.

This usage of the **h** symbol is not to be confused with that for a line of text above a system (see Section B 5.6)! Although both can occur together in the header, they are in fact, different **PMX** symbols, distinguished by their differing syntax.

B 6.3 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 the symbol **B** in the header.

B 6.4 Horizontal Spacing

Minimum Spacing between Notes

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 this value to some other fraction, enter **W.[n]**, where *n* is a single digit between 1 and 9, giving the minimum spacing, in tenths of a notehead width. This adjustment is very rarely used.

Extra Horizontal Shifts and *hardspace*

PMX will usually provide satisfactory horizontal spacing. However, there may be some occasions where you will want to adjust it manually. A symbol starting with **X** controls one of two available types of horizontal adjustment:

1. a *shift* moves one or more characters in the current voice but does not affect any other spacing anywhere,
2. a *hardspace*, by contrast, is a fixed amount of space inserted at a particular time. The horizontal positions of everything in all staves in the system will be adjusted to accommodate the added space while maintaining vertical alignment.

Here are the syntactic possibilities:

- **XS[x]** (for **Shift**). This command shifts the the next character; *x* is a decimal number, giving the amount of shift, in units of a notehead width.
- **X:[x]** or **X[x]**: initiates a “group shift”, which operates like **XS[x]**, except that everything from the insertion point onward in the current voice is shifted until a termination symbol (“shift end”) is encountered.
- **X**: ends a “group shift”.
- **X[x]** (without any **S** or **:**) is a so-called *hard space*. It inserts the specified space, given in units of notehead widths, at the present point in *all staves* of the system. If *x* is negative, space will be removed.

Because horizontal spacing in parts will usually differ from that in the score, **scor2prt** (cf. Section C 1) will by default copy *only* the shift commands **XS**[*x*], **X**: [*x*] and **X**: into the parts, but *not* the hardspace command **X**[*x*] .

This behavior can be overridden using the methods described in section C 1. There is, however, an alternative method that helps to keep **PMX** score files neat and readable: add the suboption **B** or **P** (without space, as usual) to the **X** symbol:

- **B** (for **Both**): with this option, the spacing symbol is to be applied to **both** score and part,
- **P** (for **Part**): with this option, the spacing symbol is to be applied to the **parts** only.

B 7 Macros

A **PMX** macro is a single symbol that stands literally for any any string of characters that may occur in the input file⁶⁸. Macros may be useful to save tedious typing and disk space, and make **PMX** source text more readable if you need to repeat the same string many times in a score.

You may define up to 20 macros in any single **PMX** input file. There is no practical limit to the length of the character string that a macro represents, but of course it must respect the **PMX** rules on the length of input lines and what must go on separate lines.

Macros only exist within the **PMX** input file where they are defined; you can define a completely new set of macros in another **PMX** input file. Conversely, if you want to re-use the macros of file `myopus1.pmx` in `myopus2.pmx`, you need to copy them explicitly from one file to the other.

There are two distinct ways to define and record a new macro or redefine an existing one:

1. to **Record** a **Macro**: type **MR**[*n*], followed by a space, as usual. *n*, the “name” of the macro, is an integer between 1 and 20. Everything you type after this will be processed normally, at that point in the input file, as well as stored, until you end the macro by entering the symbol **M**,
2. to **Save** a **Macro**: type **MS**[*n*]. Everything you type after this will be recorded (saved) as you enter it, *without processing* the **PMX** code. When saving macros, it is usually best to put them in the header, for readability’s sake.

Whenever you would otherwise need to re-enter the same character string, you can simply Play back the macro by typing **MP**[*n*] .

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

⁶⁸Note to seasoned programmers: **PMX** macros are not really macros in the usual sense, but merely string abbreviations; they *do not allow* for variables.

When you use macros and want to make separate parts (cf. Section C 1), some care is necessary: **scor2prt** will transfer a MR macro only into the part where it originated, but will transfer MS macros into all parts.

So if you want to make macro 1, which you will use in several or all parts and then use it immediately in the part in which you write it, don't use simply MR1, but rather

```
MS1
MP1
```

Fig. B.37 contains an example of the use of macros.



```
% macro 2 saved:
MS2 e4 f g2 o_ | M
% Bar 1 (macro 1 recorded and processed):
R1 MR1 c45 d e c | M
% Bar 2 (macro 1 played):
MP1
% Bar 3 (macro 2 played):
MP2
% macro 4 saved:
MS4 c4 g c2 o_ M
% Bar 4 (macro 2 played):
MP2
% Bar 5 (macro 3 recorded and processed):
MR3 g85 a g f e4 c | M
% Bar 6 (macro 3 played):
MP3
% Bar 7 (macro 4 played):
MP4 |
% Bar 8 (macro 4 played):
MP4 Rr /
%
```

Figure B.37: “*Frère Jacques*” (usage of macros)

B 8 Inline T_EX commands

It may be worthwhile to reiterate at this point the very different intentions of the software ingredients which make up the **PMX** tool. It is, in fact, a hierarchy:

1. At the base of it all is **T_EX**. **T_EX** (written by Donald E. Knuth) is a general and extremely powerful tool for typesetting high-quality documents on a computer that are ready for printing⁶⁹. But **T_EX** is much more than that: it is a full-grown and powerful computer programming language with which – if you are a **T_EX**pert – you can do almost anything.

There is, however, a price to pay for this power: while you don't have to be a professional computer programmer to use **T_EX**, a pretty good familiarity with the basics of programming is unavoidable.

2. **MusiX_{T_EX}** (written essentially by Daniel Taupin) is a ‘macro’ for **T_EX**⁷⁰. That alleviates most of the **T_EX** programming tasks for the special purpose of typesetting music. Thus, while it helps to have a programming background, such a background is not absolutely essentially for someone who just wants to write a few simple music scores.
3. **PMX**, the ‘preprocessor’ for **MusiX_{T_EX}** written by Don Simons and described in detail in this tutorial, is yet another level of abstraction on the long way from the musical ideas that make up a score to getting ink on the paper at certain points. The whole purpose of **PMX** is to make typesetting of music with **MusiX_{T_EX}**/**T_EX** accessible to people with essentially *no* prowess in computer programming.

As a consequence of this hierarchy, **PMX** is vastly simpler to use than **MusiX_{T_EX}**, which in turn is simpler than straight **T_EX**. But the converse of this is also true: there are elements in **MusiX_{T_EX}** that are not directly accessible with the **PMX** language, just as you cannot unleash the full power of **T_EX** with **MusiX_{T_EX}** alone. So to allow the user full access to ‘lower-level’ **MusiX_{T_EX}** or **T_EX** – while maintaining its higher-level notational simplicity – **PMX** supports the inclusion of **T_EX** commands either directly in the **PMX** input file or via an external file.

B 8.1 Including **T_EX** Commands in the **.pmx** source file

The normal way for a **PMX** user to insert user-defined **T_EX** code is to enter it directly in the **.pmx** input file; this is what is referred to as “inline **T_EX**” in this tutorial. There are four ways to do that; these differ mainly in where the **T_EX** code will appear in the **PMX** output **.tex** file. The first 3 of these have `\`, `\\` or `\\\` (1, 2 or 3 backslashes) as a starting symbol, then comes a sequence of **T_EX** commands, then another `\` (backslash) as a terminating symbol, followed by the usual space⁷¹. All three of these must be contained fully in *one line of input*, limited by default to 128 characters (cf. Section D 1); but this one line can contain any number of concatenated **T_EX** commands.

The above description needs clarification on a subtle point: the starting symbols `\`, `\\` or `\\\` are *not* really **PMX** symbols in the sense used throughout this tutorial, in that there would be ‘white space’ after them, but the **T_EX** command(s) follow them *without a blank*. This has to do with the fact that all **T_EX** commands – called ‘control sequences’ – start out with

⁶⁹“Its emphasis is on art and technology, as in the underlying Greek word.” (quote from the introduction of *the T_EX*book, the authoritative book on **T_EX**, by Donald E. Knuth.

⁷⁰To be precise, **MusiX_{T_EX}** consists of a very extensive set of **T_EX** ‘macros’ and music typesetting fonts.

⁷¹This means that type 1, 2, and 3 **T_EX** strings may *not* contain the **T_EX** macro ‘\ ’ (backslash-space). If you really need that, replace it by `\relax` .

a `\`. And thus the last of the starting backslashes (the only one for type 1, the second one for type 2, the third one for type 3) counts as the T_EX control character of the following T_EX command. For example, if you wanted to define a new T_EX command `\Myspace` as a type 1 inline command, you would write

```
\def\Myspace\hskip{10mm}\
```

whereas if it is to be a type 3 inline command, it would read

```
\\\def\Myspace\hskip{10mm}\
```

in other words: the starting symbol itself also provides the first `\` of the T_EX command (control sequence).

This convention can also be seen clearly in the examples given below.

Here are the characteristics of the first three types:

Type 1 (`\ ... \`) : a Type 1 string usually serves to define or execute a command that is only needed locally for the current line of music, right after the place where it is entered in the `.pmx` file. **PMX** inserts it in the resulting `.tex` file right before the M_usiX_T_EX command for the next note or rest. Multiple type 1 strings associated 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 T_EX commands for a single note into a single type 1 string.

If an inline T_EX string contains a new T_EX definition, this definition is in general *not* local; depending on circumstances, it can remain valid throughout the score and for *all instruments* unless redefined somewhere! This is true for all types of inline T_EX, Type 3 in particular. So be careful!

Type 2 (`\\ ... \`) : a Type 2 string usually serves to define or execute a command that is needed throughout the entire score. It will appear near the top of the `.tex` file, right before the `\startmuflex`, regardless of where it appears in the `.pmx` file.

Type 3 (`\\\ ... \`) : Type 3 strings will appear in the `.tex` file right before the `\xbar` or `\alaligne` of the current input block, i.e. before its first bar line. Thus a Type 3 string is typically used when you want to redefine a previously defined T_EX command, to be valid from a certain point on, but not before.

Type 4 : The fourth way of inserting inline T_EX in a `.pmx` file is different from the first three in several ways:

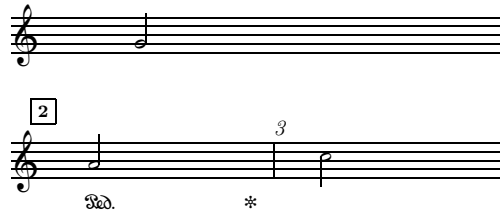
- it is started by the character sequence `---` (triple dashes), on a line by itself, as the very first line of the `.pmx` file,
- it is terminated by the next appearance of the same character sequence, `---` (triple dashes), on a separate line,

- it can contain any number of lines with \TeX code, although every single of these lines is again restricted to a total of 128 characters,
- the lines of \TeX code between the opening and terminating triple dashes will be copied *literally* to the very top of the `.tex` file.

So Type 4 can also be (ab)used to copy comment lines literally from the `.pmx` file to the `.tex` file. This is convenient for documentation purposes.

Another difference of practical importance is the way in which `scor2prt` handles these type of inline \TeX when making separate parts (cf. Section C 1): types 2-4 will be copied into all parts, while type 1 only goes into the part in which it appears in the original `.pmx` file.

To understand better the differences between these 4 types on inline \TeX , it may be useful to study the basic structure of the \TeX file `inlinesample.tex`, given in Table B.9, produced from the **PMX** input file `inlinesample.pmx`, given in Fig. B.38. In doing so, you may want to look at the box on p. 83.



```

% beginning of inlinesample.pmx
---
% This sample demonstrates the use of all 4 types of inline TeX
---
% PREAMBLE:
%
1      1
1      2      0      0
0      0
1      2      16     .0
%

t
./
% BODY:
% HEADER:
w80m
\\nobarnumbers\
% begin of music input:
g24 | /
\\systemnumbers\
\zcharnote{-2}{\PED}\ a D"\DEP"+3+15 | /
\\barnumbers\ c Rb /
% end of inlinesample.pmx

```

Figure B.38: A sample for the use of inline T_EX

```

%%%%%%%%%%
%
% inlinesample.tex
%
%%%%%%%%%%
6 % This sample demonstrates the use of all 4 types of inline TeX
7 \input musixtex
8 \input pmx
9 \input musixmad
  \smallmusicsize%
  \nopagenumbers
  \tracingstats=2\relax
  \hsize=227pt
  \vsize740pt
  \def\nbinstruments{1}
  \setstaves11
  \setclef10
  \setname1{ }
  \generalsignature{ 0}%
  \parindent 0pt
  \elemskip1pt\afterruleskip1.000pt\beforeruleskip0pt\relax
  \stafftopmarg0pt\staffbotmarg5\Interligne\interstaff{10}\relax
23 \nostartrule
24 \readmod{inlinesample}
25 \nobarnumbers%
  \startmuflex\startpiece\addspace\afterruleskip%
  % BODY:
  % begin of music input:
  \znotes\zcharnote{16}{\titles{2.0}{0}{0}{0}}\en%
  % Bar count 1
31 \pnotes{4.00}\hu g\en%
32 \systemnumbers%
  % Bar count 2
34 \alaligne
35 \pnotes{4.00}\zcharnote{-2}{\PED}\txtdyn{-2}{15.0}{\normtype\it \DEP}%
  \hu{'a}\en%
37 \barnumbers%
  % Bar count 3
39 \xbar
  \pnotes{4.00}\hl{'c}\en%
  \endpiece
  \vfill\eject\endmuflex
  \bye

```

Table B.9: `inlinesample.tex`, as produced from `inlinesample.pmx`

Here is a short survey of what happens in the file `inlinesample.tex` (cf. Table B.9), produced from `inlinesample.pmx` (cf. Fig. B.38):

- first, after the file name comment, which is inserted automatically by `pmxab`, the Type 4 string (the comment line on the top of the `.pmx` file) is inserted [line 6],
- then the files `musixtex.tex`, `pmx.tex` and `musicmad.tex` are always read in [lines [7,8,9]]. This means that the `.tex` that **PMX** produces has *access to all the T_EX macros* defined in these files, and therefore you can use all of these in your inline T_EX commands,
- next, up to and including `\nostartrule` [line 23], there are a lot of settings, the details of which need not concern us here,

The following T_EX macro [line 24], `\readmodsample`, tells T_EX to read in the file `inlinesample.mod` (if it exists). The file `inlinesample.mod` may contain any valid T_EX code of your choosing (cf. Section B 8.3 for details on this),

- next [line 25] is the Type 2 string `“\nobarnumbers\”`. Type 2 strings are *always* inserted at this point of the `.tex` file,
- `\startmuflex` [line 26] – never mind what exactly it does – is the actual beginning of the musical code,
- several lines down [line 31] there is an item `\hu g`: this is the half-note *g* of the first bar,

line 32 contains the Type 3 string `“\systemnumbers\”`. This changes the bar numbering scheme to the **PMX** default (cf. Section B 5.9). **PMX** inserted this Type 3 string right before the next `\alaligne` [line 34],

line 35 contains the Type 1 string `“\zcharnote-2\PED\”`, which prints the MusiX_T_EX pedal ^a,

line 37 contains another Type 3 string `“\barnumbers\”`, before the code for bar 3, inserted right before `\xbar` in line 39 (coding the bar line). This changes the bar numbering scheme once again.

^aThe ‘end of pedal’ symbol `\DEP` is coded here in an alternative (perhaps simpler) way by using a **PMX** dynamics textual symbol (cf. Section B 4.12).

B 8.2 Denoting pitch in inline T_EX

When using inline T_EX, you should be aware of a subtle aspect, which can be a hazard:

In MusiX_TE_X, you can always refer to a pitch with a number, rather than a letter, and that number denotes exactly the position on whatever staff you are using, counting from the bottom line as 0 .

When you use this for a pitch, it will not be transposable, nor will it respect (or indeed affect!) the memory **PMX** keeps on the pitch: remember that **PMX** doesn't interpret any inline T_EX .

You should keep this in mind when designating a pitch within an inline T_EX command.

We close this section with two practical examples and a final remark:

- As was stated in Section B 4.11.3, when using Type M slurs you need to insert the line

```
\\input musixpss\relax\
```

into the header of the .pmx file. This tells **PMX** to read in the file `musixpss.tex`, which contains the macros for Type M slurs.

This is the case of a type 2 inline T_EX string, and thus the input command is added to the .tex file right before `\startmuflex` (which is where it belongs).

- Sometimes one needs clefs other than those defined in **PMX** (cf. Section B 3.1). Some special clef symbols (among them no clef at all) are defined in MusiX_TE_X and can be invoked by the command `\setclefsymbol` (cf. the MusiX_TE_X 1.15 manual, Sec. 2.21.2). The most common of these are octave treble and octave bass clefs, in which a small 8 is attached to either the top or the bottom of the clef symbol. To get a higher octave treble key for instrument number 2, for example⁷², just add the following type 2 inline MusiX_TE_X command

```
\\setclefsymbol{2} \trebleoct\ .
```

The other 3 octave clef symbols are `\treblelowoct`, `\bassoct`, `\basslowoct`, for lower octave treble, higher octave bass and lower octave bass clefs, resp.

More examples can be seen in Section E and in some of the scores of the appendix.

B 8.3 Putting T_EX Commands in an external file

PMX provides one further option for entering an unlimited set of T_EX commands, read into the .tex file by `\readmod` just before the `\startmuflex` line. command, and before any Type 2 inline T_EX strings. Simply put the commands into a text file named `[filename].mod` in the current directory⁷³.

⁷²Matters are more complicated for instruments with more than 1 staff, such as piano or organ, because the number in the command refers to the number of the *instrument*, not the staff. Cf. the MusiX_TE_X 1.15 manual, Sec. 2.27.10 for details.

⁷³This feature is retained mainly for backward compatibility with previous versions of **PMX**; it has now been essentially replaced by the 4 options for inline T_EX strings described above.

Chapter C

Special Features

C 1 Making Parts from a Score

Separate parts can be made from a score with the tool **scor2prt** (included in the standard **PMX** distribution).

Suppose you want to produce the parts to page 1 of a J.Chr.Bach quartet (Fig. C.1), and the `.pmx` file of that score is called `JChBach.pmx`. All you need to do is to run

```
scor2prt JChBach
```

from the command line (if you omit the file name, you will be prompted for one). The program will then create the following `.pmx` files, one for each instrument:

```
JChBach1.pmx  
JChBach2.pmx  
JChBach3.pmx  
JChBach4.pmx
```

To obtain the parts, you then need to run each of these files through **PMX**, in the usual way.

As can be seen in the example, the parts files will be named `[yourscorename][n].pmx`, where `[n]` is the number of the instrument, by default; but the names of the files created by **scor2prt** can be changed from within the **PMX** file for the whole score, the “parent file”. For example, to cause the file for instrument 3 to be named `mviolins.pmx`, include the command `AN3"mviolins"` in the header of the parent file (cf. Table B.8).

In the remainder of this section we describe how to control the layout of the parts separately from that of the score by using commands that are placed in the parent file. This eliminates the need for editing the `.pmx` files for the parts separately. You can accomplish all the editing in the parent file, and then re-run **scor2prt** as required. Thus both the score and the parts can be corrected together, and the parts need not be re-edited each time they are re-generated from the score.

Quartett B Dur

Johann Christoph Bach (1735 - 1782)

Allegro

The image shows the first ten measures of a quartet in B major by Johann Christoph Bach. The score is written for four instruments: Oboe, Violine (Violin), Viola, and Violoncello (Cello). The tempo is marked 'Allegro'. The key signature has one flat (B major). The time signature is common time (C). The first system (measures 1-3) features the Oboe with a melodic line marked 'mf' and three fingerings (1, 2, 3). The Violine and Viola have sustained notes, also marked 'mf'. The Violoncello plays a rhythmic pattern of eighth notes, marked 'mf'. The second system (measures 4-6) shows the Oboe with dynamic markings 'p' and 'f'. The Violine and Viola play sixteenth-note patterns, marked 'p' and 'f' respectively. The Violoncello has dynamic markings 'p', 'f', and 'p'. The third system (measures 7-10) continues the Oboe's melodic line with dynamics 'f' and 'p'. The Violine and Viola play sixteenth-note patterns, marked 'f' and 'p'. The Violoncello has dynamic markings 'f' and 'p'.

Figure C.1: F.Chr. Bach, quartet B-Dur (p. 1)

C 1.1 Usage

Since comment lines are by definition disregarded by \TeX and by the **PMX** program itself (**pmxab**, to be exact), they can, with a slight extension of their syntax, be used to transfer information *meant for the parts only*¹. The following rules are used for this by the **scor2prt** program:

- If a line has % in columns 1 and a space in column 2, it is considered a regular comment and transferred as such to all parts.
- If a line has %% in columns 1-2, both it *and the following line* will be ignored when making parts. If the ignored line (the second line) has h, 1, or T in its first column, then *one additional line* will be ignored (cf. B 5.6 for the rationale behind this rule).
- If a line has %! in columns 1-2, these first 2 characters will be stripped, and the rest of the line will be put in the .pmx files for *all* the parts.
- If a line has %[h] in columns 1-2, where *h* is the hexadecimal digit representing the instrument number (1, 2, . . . , 9, a, b, c), the first two characters will be stripped and the rest of the line transferred to the part for instrument *h only*.

A good example for this usage is the page break(s) in a longer score, where the appropriate page numbers for the parts will be different from those in the full score. 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 hexadecimal digits a-c in this rule creates a potential incompatibility with previous versions of **PMX**. To minimize this, the character after “%” will *only* be interpreted as a part number if it represents a number less than or equal **noinst**, as given in the parent file; otherwise the entire line will be treated as an ordinary comment, and transferred to all parts.

- In addition to all entries in the header except I options (cf. Sec. C 2, p. 89), the following **PMX** symbols *with all their options* will automatically be copied to *all parts* (unless the previous line starts with %, of course):

R	(bar symbol)
V	(volta)
K	(key change or transposition)
A	(global options)

This is true for R,V,K, in spite of the fact that they can be entered in the *first* voice of the score only!

¹This is quite a useful method, employed in a similar way by other programs, such as the PostScript language.

C 1.2 The S symbol

- $S[n]$:

By default the total number of systems in each part will be the same as in the score. If you want to override this, you can do that with $S[n]$, where n is the desired number of systems. $S[n]$, if used, must appear in the header of the parent file. Legally, you could use $S[n]$ to reset the number of systems for the parent file itself, which you had just defined in the preamble. That, of course, does not make much sense, but in the form

$\%!S[n]$

or

$\%[k]S[n]$

it does, because then it does *not* affect the full score, *but all parts*, or part k only, resp. `barsant.pmx`, one of the examples given in Section G 1, includes a demonstration of this.

- $S[n]P[m]$:

`scor2prt` will also compute how many pages it thinks each part should have, and enter that into the preamble for that part. If you wish to override that, then insert into the parent file $\%3S14P2$ for example, which would 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.

- $S[n]m[k]$:

A musicsize of 20 is the default in all parts. This may be overridden with the option m in the symbol S : e.g., $\%2S15m16$. Again, you must specify the number of systems before setting the musicsize.

Keep in mind the distinction 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,
- as a **PMX** command on its own, it controls page numbering and centered headings.

C 1.3 Other usage rules

- Inline \TeX strings of Type 2-4 will be copied to all parts, while a type 1 inline \TeX string will only go into the part in which it occurs in the parent file.
- User-defined hardspaces (X without $:$) are by default not copied to parts. There are two ways to change this default:
 1. use the options of the X command (cf. Section B 6.4): B causes the hardspace to be used in **both** score and parts; P puts it into the **parts**, but not the score,

2. to insert x notehead widths of hardspace into part n , place the symbol $\%[n]X[x]$, on a line of its own, in the parent file.
- As was already noted (cf. Section B 5.7), 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, $\%!P$, with appropriate options added, anywhere within the **PMX** code representing the first page of the parts². 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.
 - 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.
 - One function of **scor2prt** is to condense consecutive bars of rest into a single group of special printed characters with a number above it. Thus **scor2prt** will automatically insert `rm` symbols (cf. Sec. B 4.5) 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 file, either with a digit or with `p`. In other words, the feature will not work if the first rest in the sequence inherits its duration from the previous note.

In standard \TeX the rule for comments is: a `%` character *and all text that follows it on the same line* is treated as a comment, i.e. it is totally ignored by the program. As a consequence of this rule, many users of \TeX have a habit of adding comments at the end of short lines of text, instead of writing a whole comment line, starting with a `%` in column 1. This is fine as far as **PMX** is concerned; but **scor2prt** redefines this rule a bit and in general *will misinterpret comments added in the same line after some legal **PMX** code.*

*So when you intend to make parts using **scor2prt**, it is good practice to restrict true comments to lines with `%` in column 1, followed by at least one blank.*

C 2 Making MIDI Files

Some users of **PMX** and **MusiX \TeX** may never have heard of MIDI and may therefore be a bit bewildered by this section. So here is a quick introduction:

MIDI stands for “**M**usical **I**nstruments **D**igital **I**nterface”. It is essentially a standard, consisting of a language and its hardware implementation, designed to produce digitally coded music with a device like a synthesizer or a multi-media

²From \TeX 's standpoint the command must occur between the beginning and end of the page on which the numbering is to begin.

computer. If your computer is equipped with the appropriate hardware and software (which is standard on PC's these days), you can listen to the music encoded in a MIDI file on your computer just as you can read the text encoded in a text file³.

This section describes how to produce such MIDI files when writing a piece of music with **PMX**. When using this facility, you should be aware both of the general restrictions of MIDI and the special ones that **PMX** imposes:

- First of all, you should not expect the MIDI files that **PMX** produces to be anything that resembles a performance of the piece by real, human musicians. While MIDI does make different shades of sound for the different instruments, the lack of any kind of expressivity (rhythmic accentuation, dynamics, vibrato, intonation etc.) does not really do justice to the musical character of acoustic instruments.

So the MIDI file can be hardly more than an acoustic check on whether the notes produced are indeed what you had intended. But for that purpose it can be of great help in coding a score with **PMX**, so it is generally advisable to produce the MIDI files.

- The MIDI module of **PMX** does not recognize graces, ornaments, repeats, volte, or segnos. The only ties that are recognized are those using `s`, `t` or `(` ; alone, with no explicit ID number.
- the MIDI file generator does not support changing the number of instruments in mid-stream. You will not see an error message, but the results will be unpredictable.

The MIDI file is produced by **PMX** concurrently with the MusiX_{TEX} output. To start this, enter the symbol `I`, together with any options (as described below), usually in the header of the input file; but `I` symbols can appear later in the file as well, but only at the *start of an input block*.

If the name of the **PMX** source file is, say, `Mymusic.pmx`, the MIDI symbol `I` will cause a file `Mymusic.mid` to be written in the directory given in the preamble. If `I` is used without any options, **PMX** will use default values for several of the parameters; these default values are indicated below.

Usually, however, you will want to specify some options. They follow the MIDI symbol `I` immediately, without a space. Sometimes the order of the options matters, so it is generally advisable to adhere to the order in which they are given here:

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

³If you are interested in details on MIDI in general, a good place to start is the home page of the "MIDI Manufacturer's Association": <http://www.midi.org/about-midi/aboutmidi3.shtml> .

2. `i[i1i2...in]` assigns MIDI instrument names $i1, i2, \dots, in$ to the staves of the respective **PMX** instruments. The default is harpsichord, of course.

If you use this option, you must specify *all* instruments. Each i is either an integer between 1 and 255 or a 2-letter mnemonic. The instrument mnemonics allowed with **PMX** are listed in Table C.1. Numbers and mnemonics may be mixed, but consecutive pairs of numbers must be separated by `:` (colon)⁴. Care is needed with multi-staff instruments: there must be one instrument name *per staff*; so for a sonata for violin and piano, e.g., the instrument names entry would be `ipipivl`, not `ipivl` !

3. `v[i1]:[i2]:[...]:in` assigns the relative MIDI volume to each instrument. Each i is an integer between 1 and 127; the colons are required. The volume parameter `v` must either be given with exactly as many parameters as there are instruments, or none at all. The default (no parameters given) is 127.
4. `b[m1]:[m2]:[...]:mn` assigns the MIDI stereo balances to each instrument. The numbers m may vary between 1 and 128; otherwise their usage is similar to that for the volume. The default value is 64, which represents the center; smaller numbers favor the left stereo channel, larger ones the right.
5. `p[x]` inserts a pause of x quarter notes – for all instruments! – at the beginning of the first bar of the input block in which the MIDI symbol appears. Decimals are allowed, but will be rounded to the nearest sixteenth note.
6. `g[i]` sets the MIDI gap to i MIDI clock ticks: 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 corresponds to 2/3 of a 64th note.

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 with some MIDI file players and editors.

The instruments given in Table C.1 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 (cf. Table C.2).

C 2.1 MIDI macros ⁵

IM initiates a MIDI macro operation. Although the syntax of a **PMX** MIDI macro is analogous to that of regular **PMX** macros (described in Section B 7), its function is different:

1. a MIDI macro cannot be saved (i.e. recorded without playing — there is no command `IMS[i]`),

⁴Otherwise the assignment could be ambiguous!

⁵**PMX** normally does not allow tempo changes in MIDI macros. As of version 2.511, this restriction no longer holds; but it is designated as “experimental”. So if you use it, be prepared for surprises! Naturally, Don Simons will certainly be interested to hear of your experiments.

pi	Acoustic Grand Piano	vl	Violin	re	Recorder
rh	Rhodes Piano	va	Viola	fl	Flute
ha	Harpsichord	vc	Cello	ob	Oboe
ct	Clavinet	cb	Contrabass	c1	Clarinet
or	Church Organ	ab	Acoustic Bass	ba	Bassoon
so	Soprano Sax	tr	Trumpet	ma	Marimba
al	Alto Sax	fr	French Horn	gu	Acoustic Nylon Guitar
te	Tenor Sax	tb	Trombone	vo	Synth Voice
bs	Baritone Sax	tu	Tuba		

Table C.1: Mnemonics for instruments acceptable in **PMX**

- the code sandwiched between **IMR**[*i*] (“start record MIDI macro”) and **IM** (“end MIDI macro”) will be recorded as with a regular macro, but
- IMP**[*i*] (“**P**layback of **M**IDI macro *i*”) will *not insert anything in the written score*, it will merely replay in the MIDI output the number of bars defined by the macro.

MIDI macros are needed for repeats or dacapos, as the **PMX** coding for these features of a score merely provides the appropriate notation on paper, nothing else. MIDI Macros must have ID numbers between 1 and 20. Only one macro may be active at a time, recording or playing, but not both. Never try nesting or overlapping macros!

C 2.2 MIDI only accidentals

In rare cases a special notation for accidentals is needed to guarantee that the MIDI output corresponds exactly to what is intended in the score. Cases in point are:

Repeated notes with accidentals: the generally accepted rules of musical orthography, at least for tonal music, say that when two identical notes with accidentals occur in the same bar (and in the same voice, of course), only the first one is written with an explicit accidental; for the following notes the accidentals are implied (‘inherited’, in **PMX** parlance).

This rule is less clear when there is a bar line between two successive notes. Some composers – in particular in the baroque era, where this notation is almost the default – will assume the first note in the new bar to have ‘inherited’ the accidental along with it, while others would insist on repeating the accidental, and still others would put a cautionary accidental over the first note of the new bar. **PMX** takes the first position, and MIDI files generated by **PMX** will reflect this attitude.

PMX users with a different attitude could thus find their intention misrepresented in the MIDI output.

Editorial/dubious accidentals: in editing an historical piece, a conscientious editor will provide information about dubious points, but she nevertheless, in creating a MIDI file, might want to assert her own position.

1-8	PIANO	9-16	CHROMAT. PERC.	17-24	ORGAN
1	Acoustic Grand Piano	9	Celesta	17	Drawbar Organ
2	Bright Acoustic Piano	10	Glockenspiel	18	Percussive Organ
3	Electric Grand Piano	11	Music Box	19	Rock Organ
4	Honky-tonk Piano	12	Vibraphone	20	Church Organ
5	Electric Piano 1	13	Marimba	21	Reed Organ
6	Electric Piano 2	14	Xylophone	22	Accordion
7	Harpsichord	15	Tubular Bells	23	Harmonica
8	Clavinet	16	Dulcimer	24	Tango Organ
25-32	GUITAR	33-40	BASS	41-48	STRINGS
25	Acoustic Guitar (nylon)	33	Acoustic Bass	41	Violin
26	Acoustic Guitar (steel)	34	Electric Bass (finger)	42	Viola
27	Electric Guitar (jazz)	35	Electric Bass (pick)	43	Cello
28	Electric Guitar (clean)	36	Fretless Bass	44	Contrabass
29	Electric Guitar (muted)	37	Slap Bass 1	45	Tremelo Strings
30	Overdriven Guitar	38	Slap Bass 2	46	Pizzicato Strings
31	Distortion Guitar	39	Synth Bass 1	47	Orchestral Strings
32	Guitar Harmonics	40	Synth Bass 2	48	Timpani
49-56	ENSEMBLE	57-64	BRASS	65-72	REED
49	String Ensemble 1	57	Trumpet	65	Soprano Sax
50	String Ensemble 2	58	Trombone	66	Alto Sax
51	Synth Strings 1	59	Tuba	67	Tenor Sax
52	Synth Strings 2	60	Muted Trumpet	68	Baritone Sax
53	Choir Aahs	61	French Horn	69	Oboe
54	Voice Oohs	62	Brass Section	70	English Horn
55	Synth Voice	63	Synth Brass 1	71	Bassoon
56	Orchestra Hit	64	Synth Brass 2	72	Clarinet
73-80	PIPE	81-88	SYNTH LEAD	89-96	SYNTH PAD
73	Piccolo	81	Lead 1 (square)	89	Pad 1 (new age)
74	Flute	82	Lead 2 (sawtooth)	90	Pad 2 (warm)
75	Recorder	82	Lead 3 (calliope)	91	Pad 3 (polysynth)
76	Pan Flute	83	Lead 4 (chiff)	92	Pad 4 (choir)
77	Blown Bottle	84	Lead 5 (charang)	93	Pad 5 (bowed)
78	Shakuhachi	85	Lead 6 (voice)	94	Pad 6 (metallic)
79	Whistle	86	Lead 7 (fifths)	95	Pad 7 (halo)
80	Ocarina	87	Lead 8 (bass+lead)	96	Pad 8 (sweep)
97-104	SYNTH EFFECTS	105-112	ETHNIC	113-124	PERCUSSIVE
97	FX 1 (rain)	105	Sitar	113	Tinkle Bell
98	FX 2 (soundtrack)	106	Banjo	114	Agogo
99	FX 3 (crystal)	107	Shamisen	115	Steel Drums
100	FX 4 (atmosphere)	108	Koto	116	Woodblock
101	FX 5 (brightness)	109	Kalimba	117	Taiko Drum
102	FX 6 (goblins)	110	Bagpipe	118	Melodic Drum
103	FX 7 (echoes)	111	Fiddle	119	Synth Drum
104	FX 8 (sci-fi)	112	Shanai	120	Reverse Cymbal
121-128	SOUND EFFECTS				
121	Guitar Fret Noise				
122	Breath Noise				
123	Seashore				
124	Bird Tweet				
125	Telephone Ring				
126	Helicopter				
127	Applause				
128	Gunshot				

Table C.2: The General MIDI Instrument Specification

For these and similar applications **PMX** provides the *MIDI only accidentals*. These are written and used like normal accidentals, except with an ‘i’ added. Such accidentals are effective *only in the MIDI output*; they are disregarded in the printed score.

This is particularly useful when you want to override the MIDI “baroque default” described above. To understand the “baroque default” clearly, consider the example given in Fig.C.2, where the **PMX** coding is given directly below the system:



Figure C.2: The “baroque default”, and overriding it with MIDI only accidentals

Now compare the printed score with the MIDI version – provided [here](#) for your convenience –, and you can hear the differences explicitly⁶.

Another example for the use of MIDI only accidentals is given in Fig.C.3:

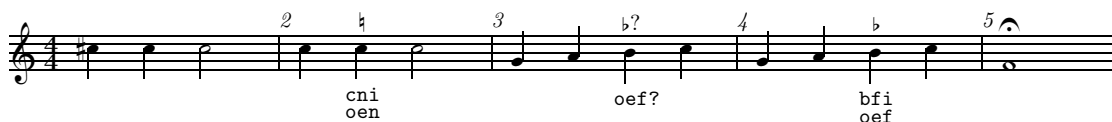


Figure C.3: Further use of MIDI only accidentals

- MIDI plays the first *C* in bar 2 as a *C* \sharp . The editorial \natural over the second *C*, however, suggests that *C* \natural was intended (at least from here on), so the MIDI accidental *in* is appended to that *C*,
- the final *F* (in bar 5) suggested to the editor that we are moving towards F major here, and thus the *B* in bar 3 should actually have been a *B* \flat . By bar 4 the assumption is confirmed, and so he changes the ‘dubious’ accidental to an editorial one, and the MIDI only accidental *fi* is appended to that *B*.

C 3 Lyrics

PMX has no special provisions for lyrics. One way to include them is by using the macro package `musiclyr.tex` developed by Rainer Dunker. It introduces lyrics into \TeX more easily than with `MusiX \TeX` ’s own facilities. The macros could be entered as inline \TeX directly into the `.pmx` file, but most would prefer the convenient interface to `musiclyr` via the program `M-Tx` developed by Dirk Laurie (cf. Section F).

⁶It is also instructive to note the MIDI implementation of slurs and ties generated by **PMX** here (Cf. the list of MIDI restrictions on p.90).

If you have foreign-language lyrics, you may run into the problem of accented letters. Straight T_EX does not provide these directly, as the modern font encoding schemes used by L^AT_EX do. However, Olivier Vogel has developed a method with which to use these encoding schemes with M-T_x/PMX/MusiX_{T_EX}. For details you should look at his contribution on this in the [Tricks and Tricks](#) section in the WIMA, or contact Olivier directly: oliviervogel@freesurf.ch

C 4 PMX and L^AT_EX

L^AT_EX (“A Document Preparation System”), although written for a very different purpose, has a lot in common with **PMX**: it is a (huge) set of T_EX macros that allows to solve complicated layout problems without the need to dive into the – sometimes arcane – depths of T_EX itself. So L^AT_EX simplifies the use of T_EX for the non-T_EXpert in much the same way that **PMX** does for MusiX_{T_EX}.

So it would certainly be wonderful to have the best of two worlds: merge L^AT_EX and **PMX**. The problems with that (or merging L^AT_EX and MusiX_{T_EX}, for that matter) are threefold:

1. **PMX** deals with many of the layout aspects of a musical score on its own (such as width of systems, line breaks, page breaks etc.), and with techniques that are not really compatible with those of L^AT_EX.
2. MusiX_{T_EX} / **PMX** and L^AT_EX make heavy use of T_EX’s resources, both in memory and registers, and together they may strain these resources beyond their usual limit.
3. both MusiX_{T_EX} / **PMX** and L^AT_EX use many special command definitions, often enough incompatible with one another.

While with modern implementations of resources are no longer a serious problem, the incompatibility problems are, and their resolution would be a major programming task. So there have, to this day, not been any serious efforts to provide a truly merged version of L^AT_EX with **PMX**.

There are, however, several methods to use L^AT_EX and **PMX** in (partial) “coexistence”:

1. The best way to include short scores, of less than one printed page, in a L^AT_EX document is to
 - (a) produce the musical score with **PMX** – MusiX_{T_EX} – **dvips** in the way described in this tutorial. The end product is a PostScript file,
 - (b) make an **.eps** (“encapsulated PostScript”) ⁷ file from the **.ps** with one of the standard tools, e.g. **ghostview**,
 - (c) include this **.eps** file in the L^AT_EX document with the L^AT_EX `\includegraphics` command, e.g.


```
\includegraphics[scale=0.58]{sample.eps}
```

⁷In general, this is possible for single-page PostScript files only.

This is the method with which the present tutorial was produced.

2. use the \LaTeX package **musixltx**, together with M μ siX \TeX ⁸. The M μ siX \TeX code is then sandwiched between the start and end of a specific environment:

```
\begin{music} ... \end{music} .
```

For more detail on this method see the M μ siX \TeX manual by Daniel Taupin.

But be aware of the fact that the `.tex` files produced by **PMX** contain a number of incompatibilities with \LaTeX . So if you want to use the **musixltx** package together with **PMX**, you may have to comment out a few lines in the `.tex` file — by hand, unfortunately. For that reason, the **musixltx** package usually works well with **PMX** only in rather simple cases. An example of how to proceed can be found in the WIMA:

<http://icking-music-archiv.org//software/musixtex/add-ons/pmx-l2e.tex>

3. Dirk Laurie, the author of **M-Tx**, has developed a \LaTeX package, **mtxlatex**. It is part of the **M-Tx 0.60b distribution**, available from the WIMA, and can be considered a successor to the **musixltx** package described above. An outstanding feature of this package is that it interfaces quite well with **PMX**; so this is usually the method of choice in cases where you want to include longer music scores in a long text such as a book. Detailed instructions on how to use this package are contained in the **M-Tx** manual.

When using **mtxlatex**, be sure to include the **mtxlatex** package as the *first* package in the \LaTeX source file.

⁸**musixltx** is part of the standard distribution of the M μ siX \TeX 1.15 package.

Chapter D

Limitations, error messages, and bugs

D 1 Limitations

For simplicity in writing the program, **PMX** has numerous variables with fixed dimensions. In most cases there are no checks against these limits, so occasionally there may be hangups due to exceeding a dimension.

The limits of the variables given in Tables [D.1](#) and [D.2](#) fall in two distinct classes, named “soft” and “hard” limits to distinguish them:

1. soft limits can be increased fairly easily: all it takes (sic!) is to change the dimensions in the FORTRAN source of `pmxab.exe` (the **PMX** program) accordingly, and recompile,
2. hard limits, on the other hand, could only be increased with more or less elaborate reprogramming ¹ (or at least recompiling the source code), so for all practical purposes they are unchangeable.

In both cases, however, it will usually be possible to work within existing limits by breaking the input into smaller blocks. Consider that solution before asking Don Simons for changes to the **PMX** source code, or attempting such changes yourself!

¹Often these hard limits are a consequence of features of MusiX_TE_X and thus cannot be changed without changes in MusiX_TE_X.

Note, however, that this character count *includes all empty spaces before the final character* (such as */*) at the end of an input line. If – for better readability, e.g. – you are in the habit of aligning these end-of line characters, being more stingy on the resulting blanc characters often helps to stay within the limit.

characters per input line	:	128
staves	:	24
voices per staff	:	2
voices per system	:	24
systems	:	125
bars	:	600
forced line breaks	:	40
forced page breaks	:	10
key changes	:	18
pages	:	20
notes per input block	:	600
bars per input block	:	15
slurs per input block	:	101
figures (figured bass) per input block	:	74
grace note groups per input block	:	37
notes in grace note groups per input block	:	74
inline \TeX strings per input block	:	52
volte per input block	:	6
trills per input block	:	24
chordal notes (non-spacing) per input block	:	62
beams per voice per bar	:	8
forced beams per voice per input block	:	40
clef changes per voice per input block	:	10
notes per beam	:	24
notes per xtuplet	:	24

Table D.1: Numerical limits of **PMX** variables (soft limits)

total number of characters in source file	:	65 536 (2^{16})
<code>\notes</code> groups (total)	:	2000
<code>\notes</code> 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	:	20
inserted anti-collision end-of-bar hardspaces per system	:	19
inserted anti-collision end-of-bar hardspaces (total)	:	83
inserted standard anti-collision spaces per system	:	400
inserted anti-collision spaces within xtuplets per system	:	100
inserted standard anti-collision spaces	:	1000
inserted anti-collision spaces within xtuplets	:	200
bytes of MIDI output data per line of music	:	24576

Table D.2: Numerical limits of **PMX** variables (hard limits)

D 2 PMX's error messages

When run on, say, `my_opus.pmx`, **PMX** will always generate two files in the working directory, `my_opus.pml` and `pmxaerr.dat`. `my_opus.pml` is a log file, and `pmxaerr.dat` contains a single integer: 0 if the run was successful, otherwise the line number in the source file `my_opus.pmx` of the fatal error (useful for batch processing). Also, on successful completion, `my_opus.tex` will be placed in the path specified in the preamble.

Usually, when there is an error, **PMX** will give you a pretty good explanation². There are some cases, however, in which the error message will be issued not by **PMX**, but by the FORTRAN compiler directly, and it may look very cryptic to you. An infamous example is:

```
forrtl: severe (24): end-of-file during read, unit 10
```

Such messages are almost always caused by an extra line in your source file that shouldn't be there, so experiment with deleting lines towards the end of your file!

D 3 Bugs

As was mentioned before, **PMX** is continuously maintained and improved by Don Simons, and there is a very active group of **PMX** users. Thus there is no serious bug known (as of Version 2.514) that hasn't been fixed.

If you think you have discovered a new bug, don't hesitate to send a message to the TeX-music users' list: tex-music@icking-music-archive.org³. The same is true if you have a problem with **PMX** that you simply cannot solve by yourself, even though you suspect that it's *not* a bug: don't hesitate to ask!

D 3.1 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 **PMX** inserting `\eject` at the end of every page, which automatically spaces the systems vertically without having to fiddle with `\staffbotmarg`. As far as is known, the warning is benign, and may be ignored.

²Note that sometimes an error message will use a name for a preamble numerical parameter that differs from that used in this tutorial — but it usually is easily recognized.

³Almost all reported “bugs” turn out to be misunderstandings. So if you state politely that you *think* you may have found a bug, and ask for help, you will avoid irate replies of the “RTFM” species.

Chapter E

Tricks of the Trade

E 1 Simple tricks

E 1.1 Special coding in L’Incoronazione di Poppea

Bar 17 in Fig. B.30 requires an explanation: the continuation figure appears under the 5, but these two figures are coded as `x145 x12_00.2`. Why not `x14500.2`, giving them both the same horizontal offset? That is because the continuation figure always starts `0.3 \noteskips` to the *left* of the associated note’s position. This works well in most normal situations. But here `\noteskip` is large, based on the dotted half, while the desired length is short, so in fact the left offset of the starting point is even larger than the desired length. Consequently the entry point of the continuation figure has to be offset by a half note to make it appear offset by a quarter note.

E 1.2 Text after final system

Suppose you want to add some text, or any other T_EX material, after the final system of a score. How can you include that in the **PMX** file?¹

The answer, given by Christian Mondrup, is instructive:

1. Write the text to a separate file, say `Myendnotes.tex`,
2. Insert

```
\let\Endpiecesav\Endpiece
\def\Endpiece{\Endpiecesav\input Myendnotes}
```

in the header of the **PMX** source file, i.e. as a Type 4 inline T_EX command (cf. Sec. B 8.1). You can see that the redefinition of T_EX commands, if done carefully, can be a rather powerful tool.

Another solution to this problem is to use the L^AT_EX/**PMX** interface `mtxlatex` (cf. C 4); this may be simpler in cases involving more complicated layout.

¹This question was posted to the T_EX-music list by Herrmann Hinsch.

E 1.3 Clef octavation

As was noted in Section B 4.14, **PMX** does not presently provide a notation for octavation; but native MusiX \TeX does; cf. the MusiX \TeX 1.15 manual, Sec. 2.11 for details.

Consider, for example, bar 155 of the 2nd movement of Beethovens piana sonata op. 111 :

```

\\interstaff{13}\
w170m
Abepl
%
[ e83d ze+ e8- ze+ sl c1- zc+ sl ] c8d-1 zc+ | Rb /
[l c15x3n g+ c b1-x3n g+ b c1-x3n g+ c ]
[l gs1x3n b gs+ g1-x3n b e a1-x3n e+ a ]
[l a-1x3n e+ a g1-x3n e+ g a1-x3n e+ a ] | /
%
```

In most editions, you will find the figures in the right hand in the second and third three beats written in octavation notation:

which is clearly easier to read. This octavated version was produced by the following **PMX** code:

```

\\interstaff{13}\
w170m
Abepl
\\def\octnumber{8$^{va}$}\
%
[ e83d ze+ e8- ze+ sfu c1- zc+ sfu ] c8d-1 zc+ | Rb /
[l c15x3n g+ c b1-x3n g+ b c1-x3n g+ c ]
\Ioctfinup1d\ [l g-s1x3n b gs+ g1-x3n b e a1-x3n e+ a ]
[l a-1x3n e+ a g1-x3n e+ g a1-x3n e+ a ] \toctfin1\ | /
%
```

The octavation is started with the Type 1 inline \TeX symbol `\Ioctfinup1d\` and ended with `\toctfin1\`; the transposition downward is, of course, generated in the standard **PMX**

way by writing `g-s1x3n` instead of `gs1x3n`. By default, MusiX_{TEX} will start the octavation symbol with a simple 8; this is changed to 8^{va} by the Type 2 inline _{TEX} symbol given in the preamble.

When using this octavation notation in a score, do not try to produce a MIDI file for that score: it will come out faulty

E 2 More tricks

The above examples are actually rather simple cases of the use of inline _{TEX}, not really “tricks”; they have been included here for you to ‘warm up’ to doing such things yourself. But there are many further-reaching tricks to overcome shortcomings of **PMX** — some truly ingenious. Here are a few of general interest:

E 2.1 Changing vertical positioning of instrument name

When making parts from a score with `scor2prt`, you will sometimes find the adjustment of the vertical position of the instrument name not to your liking. In particular, if you have a tempo indicated in a line of text above the first system (e.g. “allegro”), you may find that the instrument name of some (but not all) parts collides with the tempo indication. How can you fix that?

A practical solution of this problem has been given recently by Andre Van Ryckeghem. It is quite instructive to see in this example how the mechanism of handing over commands to `scor2prt` operates:

Suppose that you are writing a string quartet, and the relevant part of the preamble and header in the score are:

```
...
Violoncello
Viola
Violine II
Violin I
batt
./
h
Allegro
Abpl
...
```

After running `scor2prt`, you find that the `allegro` command is positioned the way you want it, but the cello name collides fully, and the Violino II slightly, with the “allegro”. Therefore you want to raise the cello by `3ex`, the Violino II by `1.5ex`. To do so, replace the above part of the preamble and header in the score by

```
...
%% Violoncello
```

```

%1 \raise+3ex \hbox{Violoncello}
Viola
%%
Violin II
%3 \raise+1.5ex \hbox{Violin II}
Violin I
batt
./
h
Allegro
Abpl
...

```

All the comment lines are disregarded in compiling the score; but in the `.pmx` file for the cello, you will find

```

Ti
\raise+3ex \hbox{Violoncello}
b
./
h
Allegro
Abpl
...

```

and in the part for the second Violin

```

Ti
\raise+1.5ex \hbox{Violin II}
t
./
h
Allegro
Abpl
...

```

which does what you want.

E 2.2 Xtuplets ending with a rest

As was pointed out in Section B 4.6, **PMX** does not allow a rest as the last note in an xtuplet.

Can one find a way to circumvent this restriction? A pedestrian, awkward way around might be this: remembering that **PMX** is a preprocessor for MusiXTeX, and in a second pass

\TeX processes the **PMX** output, one could simply enter some arbitrary note in place of the desired rest, run **PMX**, and then in the resulting \TeX file replace the arbitrary note *manually* by the appropriate MusiX \TeX rest symbol *before* running the file through \TeX for the the 2nd pass.

This semi-solution however, has two major drawbacks:

1. it takes a certain amount of acquaintance with basic MusiX \TeX , so this is not for everyone;
2. more dangerously, this hack goes away if and when you rerun **PMX** ! So if you are still in the middle of a development, you may find yourself redoing the hack over and over, which could quickly become a real nuisance.

Luckily, for xtuplets with stand-alone notes there now exists a real solution: an ingenious, generally usable inline \TeX method – developed by Don Simons – solves the problem completely for unbeamed xtuplets. Here is a simple example with two such triplets:



And this is the **PMX** source text that creates these staves:

```

2 2
2 4 0 0
0 0
0 3 20 0.12

tt
./
w100m
\\def\qpforqu{\let\qut\qu\def\qu##1{\qp\zcharnote{##1}{~}\let\qu\qut}}\
\\def\dsforqu{\let\cut\cu\def\cu##1{\ds\zcharnote{##1}{~}\let\cu\cut}}\

e4ax3 \dsforqu\ e \dsforqu\ e e4ax3 \qpforqu\ eD /
e4ax3          e          e e4ax3          eD /

```

As you can see, there are two inline commands of type 2, which are therefore effective throughout the file (from where they are first entered), then followed by two standard **PMX** lines; the lower one (the top staff) of these has the standard triplets, as demonstration, while the upper one (bottom staff) shows the corresponding triplets with ending rests.

The two inline \TeX commands produce the type of rest: `\dsforqu` stands for a quarter rest, and `\qpforqu` for a half rest. Note that these inline \TeX commands not only provide the correct graphics, but **PMX** considers them to have the corresponding duration values! So if you use wrong ones, **PMX** will complain in the usual way.

Explaining the inline \TeX commands in detail would go beyond the scope of this tutorial. But if you know some of the MusiX \TeX terminology, you will recognize that these commands conform to the MusiX \TeX names for quarter and half notes etc.²

The inline commands are then each followed by a dummy note. Usually it is a good (and safe) habit to use the same note name as the last real note in the xtuplet; but any **PMX** note name is valid; variations can produce surprising, but possibly useful graphical results. So if you feel up to it, you can play with such variations!

E 2.3 Shorthand notation for consecutive quavers

Often, in accompanying voices, in particular, there are several repeated quavers (eighth notes), e.g. 4 quavers to a half note. There is a commonly used shorthand notation for this that helps sight-reading enormously: a half-note, with a line through its stem, indicating the quavers.

Fig. E.1 shows the beginning of the cello part of a Joh. Chr. Bach quartet. In this example an inline MusiX \TeX macro (devised by Andre Van Ryckeghem) provides this shorthand.

²This knowledge may be useful if you want to produce more complicated xtuplets with ending rests than the simple examples given here. The new MusiX \TeX 1.15 manual may be helpful in such cases. If that doesn't help: consult a MusiX \TeX pert!

If you want such a shorthand for semiquavers (sixteenth notes), you need to consult the MusiX_{TEX} 1.15 manual: you will find that all you need to do is to replace `\ib10` and `\ib10` by `\ibb10` and `\ibb10`, resp.

E 2.4 Varying the stave sizes

In modern editions of works with basso continuo e.g., the implementation of the bass, as suggested by the editor, is often given in a staff using a smaller size. MusiX_{TEX} does not support such a possibility.

Fortunately, Mthimkhulu Molekwa has written two macros that provide alternative solutions of this problem that are often sufficient. The first, `musixbar.tex`, which is included in MusiX_{TEX}, allows you to easily define a set of staves (the two bottom ones, in the case of a basso continuo) that have common bar lines joining these staves *and only these*³.

The second macro, named `curly.tex`⁴, provides a brace as an accolade of a set of staves together⁵.

Suppose we want to set a sonata for violin and basso continuo with **PMX**. There will be three staves: at the bottom the basso continuo proper, then immediately above it, and grouped together with a brace accolade (in a smaller size), its implementation, and above these the violin staff.

To get this, do the following:

- indicate in the preamble the number of instruments, *three* in this case!
- add a Type 4) \TeX inline command at the beginning of the file, invoking the macros `musixbar.tex` and `curly.tex`, e.g.⁶

```

---
\let\:=\relax\input musixtex\:\sepbarrules\input pmx
\input musixbar\input curly
---
```

- insert the following \TeX inline commands in the header of the **PMX** input file:

```

\\indivbarrules\sepbarrule3\
\\setsize2\smallvalue\curlybrackets{{1}{2}}\
\let\interstaffsav\interstaff\def\interstaff#1{\interstaffsav{9}\
```

³Normally, MusiX_{TEX} will draw common bar lines for *all* staves or for *none*.

⁴`curly.tex`, if not included in your MusiX_{TEX} installation, is available in the “add-ons” software section of the WIMA.

⁵MusiX_{TEX} provides only the choir-type square accolades. For the practical usage of `curly.tex` see the Caccini example in the appendix, Fig. G.3.

⁶The MusiX_{TEX} command `\sepbarrules\` sets the vertical bar to be discontinuous, i.e. *not* extend across different staves.

The command `\interstaffsav{9}` determines the extension of the brace (and can be adjusted). The command `\\indivbarrules\sepbarrule3` says that the third staff, that for the violin, has its own discontinuous bar line, not joined with the other two.

The second line contains two commands: `\setsize2\smallvalue` sets the size of staff 2 to be smaller, and `\curlybrackets{{1}{2}}` says which staves are to be accoladed, i.e. joined by the brace.

The file `curly.tex` must, of course, be in a directory where **PMX** and **T_EX** can find it, viz. either in the current directory or in one in which the other MusiX_{T_EX} files are stored.

E 2.5 Stuff in front of the clefs of the first system

And finally, to show you what is possible, here is a special example of using inline **T_EX**: it was contributed to the MusiX_{T_EX} users' list by Olivier Vogel. Here is the literal text of his contribution, with the result shown in Fig. E.2:

Dear all,

I've got a score of "*Locus iste*" by Bruckner, which begins with the indication of the range of voices. I tried to reproduce the result, and share with you my solution, since it seems to me to be an interesting trick.

```

---
\input musixtex
\input musixlyr
\setlyrics{soprano}{%
Lo-cus i-ste a De-o fa-ctus est lo-cus i-ste a De-o %
fa-ctus est, a De-o, De-o fa-ctus est in-ae-sti-ma-bi-le}
\copylyrics{soprano}{alto}
\copylyrics{soprano}{tenor}
\copylyrics{soprano}{basse}
---
4 4 4 4 0 6 0 0 1 2 20 0.04

btvt
./
B
Abd
h180m
%1-2
\\staffbotmarg3\Interligne\
\\setclefsymbol{2}{\treblelowoct}\
\\groupbottom{1}{1}\grouptop{1}{4}\
\\sepbarrules\
\\startmflex\indent\hskip-\parindent\hbox{\vbox{\hsize=\parindent\
\\setclefsymbol{1}{\empty}\setclefsymbol{2}{\empty}\
\\setclefsymbol{3}{\empty}\setclefsymbol{4}{\empty}\
\\grouptop{1}{0}\nostartrule\generalmeter{\}\parindent=0pt\
\\startpiece\hardspace{2pt}\notes\zq{F}\nq{'C}&\zq{c}\nq{'e}&\
\\zq{N}\nq{'b}&\zq{b}\nq{'g}\en\zstoppiece}}\
\\assignlyrics{1}{basse}\assignlyrics{2}{tenor}\
\\assignlyrics{3}{alto}\assignlyrics{4}{soprano}\
...

```

(The regular **PMX** encoding of the music follows).

If you encounter a tricky **PMX** problem that you think is of general interest, so that it may have been solved by someone else already, a good place to search is the **PMX** ‘[Tips and Tricks](#)’ section of the WIMA.

And if you have found a nice trick yourself, don’t hesitate to share it via the TeX-music users’ list. Perhaps it can be included in ‘Tips and Tricks’ !

Quartett B Dur

Johann Christoph Bach (1735 - 1782)

Allegro

Oboe *mf*

Violine *mf*

Viola *mf*

Violoncello *mf*

4

5 *p*

6 *f* *p*

7 *f*

8 *f*

9 *p*

10 *p*

%------%
 %
 % JCBach.pmx
 %
 %------%
 %

Lo - cus i - ste a De - o fa - ctus est lo - cus i - ste
 Lo - cus i - ste a De - o fa - ctus est lo - cus i - ste
 Lo - cus i - ste a De - o fa - ctus est lo - cus i - ste
 Lo - cus i - ste a De - - o fa - ctus est lo - cus i - ste a

Figure E.2: A. Bruckner, *Locus iste*

Chapter F

An Extension of PMX: M-Tx

Although **PMX** is already vastly simpler to use than MusiX_{TEX}, anything can be improved. So Dirk Laurie set out to simplify **PMX** even further, and thus created **M-Tx**.

M-Tx is actually a preprocessor to **PMX**. Its input is a file with extension `.mtx`, for example `mymadrigal.mtx`, its output has the extension `.pmx`, `mymadrigal.pmx`. Its input language is similar, but not identical, to that of **PMX** and includes most of the functionality of **PMX** as a subset.

The major purpose of **M-Tx** is to facilitate the introduction of lyrics in a musical score of a song, cantata or opera. **M-Tx** does this in conjunction with the **musiclyr** package by Rainer Dunker; so if you want to use **M-Tx**, you ought to have **musiclyr** installed as well.

Laurie's Work on **M-Tx** had essentially been ended in November 1998 with the 'final' version 0.52. But, as many such projects, **M-Tx** continued to evolve; presently, the 'official' version is 0.60, dated March 2005, and the current patch is 0.60c, dated 22 November 2007 !

Since there is a full [manual on M-Tx](#) available in WIMA, these few remarks may suffice here.

Chapter G

Appendix: Examples

G 1 Dons Example Files

The ‘official’ distribution files for **PMX**, which are available from WIMA [(pmx2514.zip)], , contain 3 instructive examples (not reproduced here):

`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 Inline \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 (1690-1772). 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. It uses the global option `Ae` for equal space between systems. The space between systems was increased (using the option `AI1.1`) to give a more pleasing appearance. This is a good score to try making parts with `scor2prt`. The special command `%2S9` is used to increase the number of systems in the recorder part (as explained in Section C 1.2).

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

G 2 Full-score examples

In this appendix you will find the **PMX** code (by Luigi Cataldi) of a full piece, together with the actual score which was generated from it (Section G 2.1), as well as two extended **M-Tx** examples (also coded by Luigi Cataldi).

The **PMX** code for the Dufay Kyrie contains a few inline \TeX commands; this is done intentionally so you again can get a feel of the usage of inline \TeX ; to understand them fully, however, you will have to consult the MusiX \TeX 1.15 documentation. Similarly, you can get an idea from the Vivaldi and Caccini example what **M-Tx** is all about. For a full understanding, you again need to look at the **M-Tx** documentation.

G 2.1 Dufay, *Kyrie* (PMX code) :

```
%-----%
%
% Dufay, Kyrie
%
%-----%
%
---
\font\tit = cmcsc10 scaled \magstep 5
\font\dat = cmr12
\def\comp{\rightline{\medtype Guillaume Dufay}}
\def\data{\rightline{\medtype (1400?--1474)}}
---
2 1 3 4 3 4 0 0
1 6 20 0

bt
./
Abe
\\def\writebarno{\ifnum\barno>1\lrlap{\oldstyle\the\barno\barnoadd}\fi}%\
\\def\shiftbarno{0\Interligne}\
Tt
{\tit Kyrie}
Tc
\ vbox{\comp\data}
It92iororb60:68
% 1-6
h-4
Kyrie
dd23 | a23 bf4 | c24 d4 | ad23 | dd24 | a23 d44 //
a23 d4 | c2 d4 | e2 d4 | csd2 | f2 f4 | e2 f4 /
a24 a4 | e2 d4 | g2 f4 | e2 r4 | ad4 g8 a b | cd45 a84 b c85 /

% 7-9
c24 bf43 | ad2 | g2 d4 //
e24 d4 | e d8 c4 bf8 | d4 r+7 d /
a84 bf a g g f | ad2 | bf2 a4 /

% 10-12
e2 d4 | g4 bf a | gd2 Rd //
g8- f g e f4 | r8+8 [ d+ d cs c bn ] | dd2 /
g2 a4 | g2 fs4 | gd2 /
```

```

% 13-18
L3Mc+4
h-5
Christe
dd23 | fd2 | gd2 | d2 d4 | e2 d4 | g2 a4 //
dd24 | c2 f8 e | d4 c bf | ad2 | g2 f4 | r8+7 [ d+ c b c d ] /
ad24 s | ad2 s | r8 [ bf b a b g ] | fsd2 | g2 a4 | d- e f /

% 19-24
bf23 a4 | gd2 | d24 c4 | bf43 a g | d f e | dd2 Rd //
g4-r d+ c | d2 r4 | f2 e4 | d c b | a2 gs4 | ad2 /
g24 f4 | g r g | a bf a | g8 f4 e8 g d | f e c d cs b | dd2 /

% 25-28
L5Mc+4
h-5
Kyrie
d23 g4 | d r8 d e f | g4 f ef | d r+0 r+0 //
d24 ef4 | d2 c4 | bf2 c4 | dd2 /
a24 g4 | f2 g4 | bf a g | fsd2 /

% 29-32
g2 a4 | g2 f4 | g2 a4 | d2- d4 //
d2 cs4 | d2 a4 | bf2 c4 | ad2 /
r8 [ g g fs f e ] | g4 f8 g a bf | g4 f e | d r r /

% 33-36
g43 f g | a f e | d d+ c | d f e //
bf4 a g | fr a8 g4 f8 | a4 bf8r g4 a8 | d4-r d8+r c4 bf8 /
e84 d4 c8 bf4 | a8 c4 d8 c4 | d8 f4 g8 e4 | f8 d4 a8+ g4 /

% 37-39
d2 c4 | dd2 | gd2- //
d2 e4 | fd2 of-2 | dd2 of-2 /
a4 b c | ad2 of | gd2 of /
%
%-----%
%
% end of Dufay, Kyrie
%
%-----%

```

KYRIE

Guillaume Dufay
(1400?-1474)

Kyrie

7

Christe

19

Kyrie

32

Detailed description: This figure shows a piano accompaniment for a Kyrie by Guillaume Dufay. The score is in 3/4 time and consists of six systems of two staves each (treble and bass clef). The first system is labeled 'Kyrie' and measures 1-6. The second system is labeled '7' and measures 7-12. The third system is labeled 'Christe' and measures 13-18. The fourth system is labeled '19' and measures 19-24. The fifth system is labeled 'Kyrie' and measures 25-31. The sixth system is labeled '32' and measures 32-37. The music features a mix of quarter, eighth, and sixteenth notes, with some rests and accidentals. The bass line is generally more active than the treble line, providing a harmonic foundation.

Figure G.1: G. Dufay, *Kyrie* (generated by PMX)

G 2.2 Vivaldi, *Mundi Rector* (M-Tx code) :

```

%-----%
%
% Vivaldi, Mundi Rector
%
%-----%
%
Title: \vbox{\titA\titB}
Composer: \vbox{\comp\data}
Flats: 2
Meter: 3/8
Pages: 1
Systems: 2
Style: SATB4
Size: 16
Space: 6 6 6 12

%%\font\rxii = cmr12
%%\font\tixii = cmti12
%%\def\titA{\centerline{Mundi rector}}
%%\def\titB{\centerline{\rxii {\tixii Juditha Triumphans}, I, 27}}
%%\def\comp{\rightline{\medtype Antonio Vivaldi}}
%%\def\data{\rightline{\rxii (1678--1741)}}
%%\let\endpiecesav\endpiece\
%%\def\endpiece{\endpiecesav\input judt}\
%%\font\rix = cmr9
%%\rix

{sopA}={altoA}={tenorA}={basA}
Mun-di Rec-tor de Cae-lo mi-can-ti
Au-di pre-ces, au-di pre-ces, et su-sci-pe vo-ta
Quae de cor-de pro te di-mi-can-ti
Sunt pie-ta-tis in si-nu de-vo-ta.

{sopB}={altoB}={tenorB}={basB}
In Ju-di-ta tuae le-gi di-ca-ta
Flam-mas dul-cis, flam-mas dul-cis, tui_a-mo-ris-ac-cen-de
Fe-ri-ta-tis sic hos-tis do-ma-ta
In Be-thu-liae spem pa-cis in-ten-de.

{sopC}={altoC}={tenorC}={basC}
Re-di, re-di iam Vic-trix pu-gnan-do
In ci-li-cio, in ci-li-cio in pre-ce ri-vi-ve
De Ho-lo-fer-ne sic ho-die trium-phan-do
Pia_Ju-di-tha per sae-cu-la vi-ve.

```

```

%%Abp
%%B
%%\def\writebarno{\ifnum\barno>1\lrlap{\oldstyle\the\barno\barnoadd}\fi}%
%%\def\shiftbarno{0\Interligne}
%%\It58ibaclobobb60:70
% 13
%%h-4
%%{\ppff ~~~Allegro}
r8 d+ e | e d d | [ c1 b ] c8 a | b g4
L: {sopA,sopB,sopC}
r8 g g | g g g | fs f f | d d4
L: {altoA,altoB,altoC}
r8 b c | c b b | a a c | b b4
L: {tenorA,tenorB,tenorC}
@+13 r8 g g | g g g | a d- d | g g4
L: {basA,basB,basC}

%17-20
r8 b b | [ b1 a ] b4 | r8 c c | [ c1 b ] c8 e
r8 f f | [ g1 f ] g4 | r8 g g | [ a1 g ] a8 c
r8 d d | b b4 | r8 g g | c c e
r8 d d | e e4 | r8 en e | f f f

%21-24
d8 e c | ( ~ [ b1 a b c d e ] | [ f b- ] c4 oT0 ) ~ | bd4 :|:
b8 c a | ( ~ [ bd8 a1 b c ] | d b a4 oT0 ) ~ | bd4 :|:
f8 g f | ( fd4 | fd4 ) | fd4 :|:
b8 e- f | ( ~ ( bd4- | b8 ) f4+ ) ~ | bd4- :|:

%25-28
%%L2
r8 bn b | [ c1 bn ] c8 g | d+ d d | [ e1 d ] c4 |
r8 g g | g g g | g g g | g g4 |
r8 d d | e e e | d d d | c c4 |
r8 g+ g | c- c c | bn b b | c e4 |

%29-31
c8 b4 | b8 a c | b c a |
a8 g4 | g8 fs a | g a fs |
fs8 d4 | eN8 FS F | D EF D |
D8 D4 | CS8 D D | G C- D |

%32-34
( ~ [ G1 FS G A B C ] | [ D G- ] A4 OT0 ) ~ | GD4 OF :|
( ~ [ GD8 FS1 G A ] | [ B G ] FS4 OT0 ) ~ | GD4 OF :|
( DD4 | DD ) | DD OF :|
{ ~ ( GD4 | G8 ) D4 } ~ | GD- OF :|
\END{VERBATIM}

\NEWPAGE

```

```

\BEGIN{VERBATIM}
%%%%%%%%%% TEXT WRITTEN TO A SEPARATE FILE
%%%%%%%%%% JUDT.TEX
\FONT\BIG = CMSC10 SCALED \MAGSTEP 5
\LEFTSKIP = 3 CM
\FONT\LYR = CMTI10
\LYR

\DEF\LYROFFS{5MM}

\VSKIP 10 MM
\HALIGN{\HSKIP 12MM # \HFILL & # \HFILL & # \HFILL & # \HFILL \CR

MUNDI RECTOR DE CAELO MICANTI
& IN JUDITA TUAE LEGI DICATA
& REDI, REDI IAM VICTRIX PUGNANDO \CR

AUDI PRECES ET SUSCIPE VOTA
& FLAMMAS DULCIS TUI AMORIS ACCENDE
& IN CILICIO IN PRECE RIVIVE \CR

QUAE DE CORDE PRO TE DIMICANTI
& FERITATIS SIC HOSTIS DOMATA
& DE HOLOFERNE SIC HODIE TRIUMPHANDO \CR

SUNT PIETATIS IN SINU DEVOTA.
& IN BETHULIAE SPEM PACIS INTENDE.
& PIA JUDITHA PER SAECULA VIVE. \CR
\END
%
%-----%
%
% end of Vivaldi, Mundi Rector
%
%-----%

```

Mundi rector

Juditha Triumphans, I, 27

Antonio Vivaldi
(1678–1741)

Allegro

Mundi Rector de Caelo mi-canti Audi preces, audi preces, et susci-pe vo - - ta
In Ju-di-ta tuae le-gi di-cata Flammas dulcis, flammis dulcis, tui amorisac-cen - - de
Redi, re-di iam Victrix pugnando In ci-li-cio, in ci-li-cio in prece ri-vi - - ve

Mundi Rector de Caelo mi-canti Audi preces, audi preces, et susci-pe vo - - ta
In Ju-di-ta tuae le-gi di-cata Flammas dulcis, flammis dulcis, tui amorisac-cen - - de
Redi, re-di iam Victrix pugnando In ci-li-cio, in ci-li-cio in prece ri-vi - - ve

Mundi Rector de Caelo mi-canti Audi preces, audi preces, et susci-pe vo - - ta
In Ju-di-ta tuae le-gi di-cata Flammas dulcis, flammis dulcis, tui amorisac-cen - - de
Redi, re-di iam Victrix pugnando In ci-li-cio, in ci-li-cio in prece ri-vi - - ve

Mundi Rector de Caelo mi-canti Audi preces, audi preces, et susci-pe vo - - ta
In Ju-di-ta tuae le-gi di-cata Flammas dulcis, flammis dulcis, tui amorisac-cen - - de
Redi, re-di iam Victrix pugnando In ci-li-cio, in ci-li-cio in prece ri-vi - - ve

13
Quae de cor-de pro te di-mi-can-ti Sunt pie-ta-tis in si-nu de-vo - - ta.
Fe-ri-ta-tis sic hostis do-ma-ta In Be-thuliae spem pacis in-ten - - de.
De Ho-lo-fer-ne sic hodie triumphando Pia Judi-tha per saecu-la vi - - ve.

Quae de cor-de pro te di-mi-can-ti Sunt pie-ta-tis in si-nu de-vo - - ta.
Fe-ri-ta-tis sic hostis do-ma-ta In Be-thuliae spem pacis in-ten - - de.
De Ho-lo-fer-ne sic hodie triumphan-do Pia Judi-tha per saecu-la vi - - ve.

Quae de cor-de pro te di-mi-can-ti Sunt pie-ta-tis in si-nu de-vo - - ta.
Fe-ri-ta-tis sic hostis do-ma-ta In Be-thuliae spem pacis in-ten - - de.
De Ho-lo-fer-ne sic hodie triumphan-do Pia Judi-tha per saecu-la vi - - ve.

Mundi Rector de Caelo micanti *In Juditha tuae legi dicata* *Redi, redi iam Victrix pugnando*
Audi preces et suscipe vota *Flammis dulcis tui amoris accende* *In cilicio in prece rivive*
Quae de corde pro te dimicanti *Feritatis sic hostis domata* *De Holoferne sic hodie triumphando*
Sunt pietatis in sinu devota. *In Bethuliae spem pacis intende.* *Pia Juditha per saecula vive.*

Figure G.2: A. Vivaldi, *Mundi Rector* (generated by M-Tx/PMX)

G 2.3 Caccini, *Amor l'ali m'impenna* (M-Tx code)

```

%-----%
%
% Caccini, Amor l'ali m'impenna
%
%-----%
%
Title: {\tit {Amor l'ali m'impenna}}
Composer: \vbox{\compA\compB}
Meter: C
Style: Singer Cont Bass
Singer: Voices Sop; Vocal; Clefs G
Cont: Voices RH1,RH2; Clefs G
Bass: Voices B; Clefs F
Flats: 1
Systems: 12
Pages: 3
Space: 4 1
Size: 20
Indent: 0.10
Name: {\it{Soprano}} ~ {\it{Continuo}}

%%\input musixbar\input curly
%%\font\data = cmr10
%%\font\tit = cmb10 scaled \magstep 4
%%\font\rxii = cmr12
%%\font\mov = cmmib10 scaled \magstep 1
%%\font\num = cmr9
%%\def\compA{\rightline{\rxii Giulio Caccini}}
%%\def\compB{\rightline{\data (1550--1618)}}
%%\def\writebarno{\ifnum\barno>1\lrlap{\oldstyle\the\barno\barnoadd}\fi}%
%%\def\shiftbarno{0\Interligne}

{Aria}
A-mor l'a-li m'im-pen-na.
A-mor dol-ce, a-mor ca-ro,_a-mor fe-li-ce.
Tal che non spe-ro pi{\u} n{\e} pi{\u} mi li-ce.
Pas-so nem-bi_e pro-cel-le,
pas-so'l ciel e le stel-le,
del pia-cer que-st'\e'l re-gno.
Ah, mia for-tu-na non se l'ab-bia a sde-gno.
Que-sto, que-sto m'ac-co-ra:
ch'al-tri ca-deo,
ch'al-tri ca-deo dal pa-ra-di-so_an-co-ra.
Ah, mia for-tu-na non se l'ab-bia a sde-gno.
Que-sto, que-sto m'ac-co-ra:
ch'al-tri ca-deo,
ch'al-tri ca-deo dal pa-ra-di-so_an-co-ra.

```

```

%%w187m
%%h251m
%%Ab
%%B
%%It86ivchaobb58:70:64
%% \setsize2\smallvalue\curlybrackets{{1}{2}}\
%%\let\interstaffsav\interstaff\def\interstaff#1{}\interstaffsav{9}\
%%\indivbarrules\sepbarrule3\
% Bar 1
@+2 f0
L: {Aria}
f0
c0 za
f0

% Bar 2
f2 ( f2
f2 f4 e
a2 zd c zf-
d2 - a2 6

% Bar 3
f4 ) f8 g8 (~ [ a1 b1 c1 b1 ] [ a8 g8 ]
d4.e f.e
b2 zf f zc+
b2 a2 6

% Bar 4
[ fd8 g1 ] [ f8 g8 ] [ a1 ( g g8 ) ] [ a1 ( f f8 ) ] )~
d0e
cd2 zf- g4 zbn
d2 7 ( d4 7 d4 #6 )

% Bar 5
g2 r4 g8 g8
e2 g
c2 zg e+ zc
c2 c2

% Bar 6
b2 g4 d8 d8
b4 zd- a+ zc- g2+
g2e+ d zb
g2 - g2

% Bar 7
f4 [ f1 e1 f1 g1 ] a4 [ e1 d1 e1 f1 ]
fd2 za- g4 ze+
de2 c
d2+ - a4 c4

```

```

% Bar 8
g0
d0 zg-
ce2 bn
\zcharnote{-6}{\num 11}\ ( g2 \zcharnote{-6}{\num 10}\ g )

% Bar 9
g0
e0+
c0 zg
c0

% Bar 10
c4 g8 g8 b4 [ a1 b1 c1 b1 ]
g2 g
c2 ze d zb
c2 g2 -

% Bar 11
ad4 d8- f4 [ e1 f1 g1 f1 ]
f2.g
a2 zd d zb
d2+ - b4 g4 -

% Bar 12
e0
\zw e\ dr2 cs
a2d-2 g4
%% The two following lines must actually be written in one line!
\zcharnote{-6}{\num 11}\ (5t a2 \zcharnote{-6}{\num $\sharp $10}\
(6t a4 )5t a 7 )6t

% Bar 13
d2 r4 fs8 f8
d2 fs
fs2 zd+ d za
d2 # d2

% Bar 14
g4 d8 d8 (~ [ e8 f1 e1 ] [ f1 e1 d1 e1 ] )~
g4 fn e2
b2 zd c zg
g2- c2

% Bar 15
f2 r4 fs8 f8
f2 fs
a2 zc d za
f2 d2

```

```

% Bar 16
g4 d8 d8 [ e8 f8 fd8 e1 ]
g4 fn e2
b2 zd c zg
g2- c2

% Bar 17
f4 f8 g8 a2
f2 f
a2 zc c za
f2 f4 e8 d8

% Bar 18
cd8 c1 (~ [ b1 a1 g1 f1 ] g2 )~
e4 ( f f ) e
g2 zc c zg
%% The two following lines must actually be written in one line!
( c4 \zcharnote{-6}{\num 11}\ c4 )
\zcharnote{-6}{\num 11}\ c4 \zcharnote{-6}{\num 10}\ c4

% Bar 19
f0
f0
a0 zc
f0-

% Bar 20
ad2 g8 f8
f4 ( c c ) bn
a4 g ( f f )
f4+ e4 ( d4 d )

% Bar 21
e8 d8 e8 f8 g2
c2 r8+0 e d c
g2 ze rb
( c2 c8 ) c8 b8 a8

% Bar 22
g4 d4 d2
bnr4 ( c c ) b
g0 zd+
%% The two following lines must actually be written in one line!
\zcharnote{-6}{\num $\sharp $10}\ ( g4 \zcharnote{-6}{\num 11}\ g )
( \zcharnote{-6}{\num 11}\ g4 \zcharnote{-6}{\num $\sharp $10}\ g )

% Bar 23
c0
e0

```

c0 zg
c0

% Bar 24
gd4+ e8 a4 g8 f8
e2 e4 f
g2 zc c za
c2 a4 f4

% Bar 25
g2 f8 f8 f8 e8
e2 f
g2 zc c za
c2+ f2-

% Bar 26 d2 r8 b8+ b a f2 g b2 zd d zb b4 b8 a8 g2 -

% Bar 27 g8 e8 e8 f8 g4 [f1 g1 a1 b1] g4 f e (f c2 ze c zg c2
c4 \zcharnote{-4}{\num 11}\ c4

% Bar 28 g0 f2) cr4 b g2 zc e zg- %%%% The two following
lines must actually be written in one line! \zcharnote{-4}{\num
11}\ (5t c2 (6t \zcharnote{-4}{\num 10}\ c4)5t c 7)6t

% Bar 29 f0 f0+ a0 f0-

% Bar 30 ad2 g8 f8 f4 (c c) bn a4 g (f f) f4+ e4 (d 7 d #6
)

% Bar 31 e8 d8 e8 f8 g2 c2 r8+0 e d c g2 ze rb (c2 c8) c8 b8
a8

% Bar 32 gd8 (~ g1 [f1 e1 d1 c1])~ d2 bnr4 (c c) b g0 zd+
%%%% The two following lines must actually be written in one
line! \zcharnote{-6}{\num \$\sharp \$10}\ (g4 \zcharnote{-6}{\num
11}\ g) (\zcharnote{-6}{\num 11}\ g \zcharnote{-6}{\num
\$\sharp \$10}\ g)

% Bar 33 c0 e0 c0 zg c0

% Bar 34 gd4+ e8 a4 g8 f8 e2 e4 f g2 zc c za c2 a4 f4

% Bar 35 g2 f8 f8 f8 e8 e2 f g2 zc c za c2+ f2-

% Bar 36 d2 r8 b8+ b a f2 g b2 zd d zb b4 b8 a g2 -

% Bar 37 g2 r8 c8 c8 b8 g2 a4 b c2 ze c4 zf d zg c4 c8 b8 a4 6
g4

% Bar 38 a4 b4 (c2 a2 a f2 c f2 a4 6 b4

```
% Bar 39 [ c8 ) (~ b1 a1 ] [ g1 f1 e1 d1 ] [ e1 d1 c1 d1 ] [ e1
f1 g1 a1 ] g0 c0 ze c0
```

```
% Bar 40 [ g1 c1- d1 e1 ] [ f1 g1 a1 b1 ] [ c1 b1 a g1 ] [ a1 g1
a1 f1 ] e0 c0 zg c0
```

```
% Bar 41 g0 )~ f2 e \zw N\ ce2d+0+1 b4 %%% The two following
lines must actually be written in one line! \zcharnote{-4}{\num
11}\ (1t c2 (2t \zcharnote{-4}{\num 10}\ c4 )1t c 7 )2t
```

```
% Bar 42 f0 f0 a0 f0- % %-----% % % end Caccini,
Amor l'ali m'impenna % %-----%
```

Amor l'ali m'impenna

Giulio Caccini
(1550-1618)

Soprano

A - - mor l'a - - - li m'im - pen - - -

Continuo

4

- - - - - na. A - mor dol - - ce, a - mor

7

ca - ro, a - mor fe - - li - - ce. Tal che non spe - ro

11

più né più mi - - - ce. Pas - so nem - bi e pro - cel - - -

15

le, pas - so'l ciel e le stel - - - le, del pia - cer

18

que-st'è'l re - - - gno. Ah, mia for - tu - na non se l'ab - -

11 11 10

22

bia a sde - - gno. Que - sto, que - sto m'ac - co - ra: ch'al-tri ca -

#10 11 11 #10

26

deo, ch'al-tri ca - deo dal pa - ra - di - so an - co - - - ra.

11 11 10 7

30

Ah, mia for - tu - na non se l'ab - - bia a _____ sde - -

33

gno. Que - sto, que - sto m'ac - co - ra: ch'al-tri ca - deo, ch'al-tri ca -

37

deo dal pa - ra - di - so an - co - - - - -

40

- - - - - ra.

Detailed description of the musical score: The score is for a vocal piece with piano accompaniment. It is in G major and 3/4 time. The key signature has one sharp (F#). The time signature is 3/4. The score is divided into four systems, each starting with a measure number (30, 33, 37, 40). The vocal line is written in a soprano clef, and the piano accompaniment is in a grand staff (treble and bass clefs). The lyrics are in Italian. The piano part includes various ornaments and fingerings, such as 7 #6, #10 11 11 #10, 6, and 11 10 7. The score ends with a double bar line.

Figure G.3: G. Caccini, *Amor l'ali m'impenna* (generated by M-Tx/PMX)

Index

- L^AT_EX**, 14, 95
 - mtxlatex**, 96
 - musixltx**, 96
 - e_T**E**X, 96
 - interface with **PMX**, 14, 96, 100
 - with short musical scores, 95
- M-T_x**, 2–4, 94, 96, 112
 - full-score examples, 117, 121
- MiK_TE_X**, 11
- MusiX_TE_X**, 2–4, 78
 - `\DEP`, 81
 - `\PED`, 81
 - `\setclefsymbol`, 107
 - `\spread[x]`, 72
- PMX**, 2–4, 78
 - author, 6
 - commands for all voices, 56
 - commands for individual staves, 24
 - concatenating several files, 13, 68
 - full-score example, 114
 - macros, 76
 - no variables, 76
 - options, 26
 - general, 22, 69
 - global, 69, 70, 113
 - meter options, 18
 - notes, 26, 28
 - running **PMX**, 10, 11
 - symbols, 24
- T_EX**, 2, 78
 - line break symbol, 63
 - space symbol, 59
- `\spread[x]`, 72
- mtxlatex**, *see* **L^AT_EX**, 100
- musixflx**, 10
- musixltx**, 96
- musixlyr**, 94, 107, 112
- musixpss**, 50, 84
- pmxab**, 10, 11, 16, 22, 83
- scor2prt**, 55–57, 65, 89
 - page numbering, 89
 - adjusting instrument name, 64, 102
 - HEX digit usage incompatible with previous **PMX** versions, 87
 - macros, 76
 - multibar rest, 29
 - placement of bar symbols, 57
 - usage, 87, 88, 102
 - use of **Ti**, 63, 102
- .eps file, *see* encapsulated PostScript file
- curly.tex**, 106
- fracindent**, 21
- mtrdenl**, 18
- mtrdenp**, 18
- mtrnuml**, 18
- mtrnump**, 18
- musicsize**, 20, 71
- musixlyr.tex**, 94
- ninstr**, 17
- nkeys**, 20
- npages**, 20, 67
- npickup**, 20
- nstaves**, 17
- nsystems**, 20, 67
- pmxab.exe**, 97
- pmxaerr.dat**, 11, 99
- T_EX-music users' list**, 7, 11
- A symbol, 70
- A symbol, 69

- accented letters in lyrics, 94
- accidentals, 24, 28
 - absolute, 70
 - big, 70
 - cautionary, 28
 - dubious, 38
 - editorial, 38
 - MIDI, 28, 92
 - position shift, 26
 - relative, 61, 70
 - small, 70
- arpeggio, *see* chords
 - across staves, 35
 - moving arpeggio line to the left, 34
- authors, 6
- automatic beaming, 42
- B symbol, 75
- bar lines, 57
 - dotted, dashed, 58
 - in source, 22
 - single, double, repeat, 57
- Bar number count adjustment, 69
- bar numbering, 67, 69
- basso continuo, *see* figured bass
- beaming, 39
 - beamed groupings, 41
 - forced, 39
 - in xtuplet, 32, 41
 - inhibited, 26, 39
 - large jumps, 41
 - parameters, 39
 - rest within beamed notes, 41
 - single-slope beam, 41
 - staff-jumping, 41
 - up/down beams, 41
- beams across bar lines, 43
- block, *see* input block
- body of Input File, 22
- breath, 38
- breve, 24
- bugs, 99
- C symbol, 52
- caesura, 38
- Cataldi, Luigi, II, 113
- chords, 33
 - arpeggio, 34
 - main note, 33
 - position of accidentals, 34
 - stem length and direction, 34
- clef codes, *see* clefs, symbols
- clefs, 21
 - change, 52
 - change with two voices in a staff, 53
 - empty, 84
 - exotic, 21, 84
 - octave clefs, 84
 - symbols, 21, 84
- Codogno, Maurizio, 8
- comments, 14
 - usage with **scor2prt**, 87, 89, 102
- Coulon, Jean-Pierre, 8
- crescendo, *see* dynamic marks
- current directory, *see* directory
- D symbol, 51
- dacapo, *see* volte
- decrescendo, *see* dynamic marks
- directory, 22, 84
- dotted notes, *see* notes
 - shorthand, 27
- Dunker, Rainer, 6, 58, 94, 112
- dynamic marks, 51
 - hairpin, 52
 - length limit, 43
 - textual, 52, 83
- eighths, 24, 105
- encapsulated PostScript file, 95
- error messages, 99
- F symbol, 55
- fermata, 38
- figured bass, 54
 - 2-digit figures, 55
 - continuation, 55, 100
 - varying staves sizes, 106
- genlayout, 71

- grace notes, [35](#)
 - ‘after’-grace, [36](#)
 - added space, [36](#)
 - in xtuplet, [31](#)
 - not in MIDI, [90](#)
 - slurs, *see* slurs, in grace notes
- h symbol, [64](#), [74](#)
- hairpins, *see* dynamic marks
- hard space, *see* spacing
- header, [22](#), [69](#)
- Hinsch, Herrmann, [100](#)
- I symbol, [90](#)
- Icking, Werner, [7](#)
- indenting first system, [21](#), [67](#)
- inline TeX, [77](#)
 - bar numbering, [69](#)
 - handled by **scor2prt**, [80](#)
 - in .pmx source, [78](#)
 - in external file, [84](#)
 - pitch given as a number, [84](#)
 - placement in .tex file, [83](#)
 - special trick, [107](#)
 - Type 1/2/3/4, [79](#)
- input block, [22](#)
 - ending, [22](#)
- Installation, [4](#)
- instruments, [21](#)
 - names, [21](#), [102](#)
 - in MIDI, [91](#)
 - numbers increased/decreased, [68](#)
- K symbol, [61](#)
- key change, [61](#)
- key signature, [20](#), [70](#)
- Kneifl, Stanislav, [7](#), [43](#)
- Knuth, Donald E., [2](#), [78](#)
- l symbol, [64](#)
- Lamport, Leslie, [67](#)
- Laurie, Dirk, [6](#), [94](#), [96](#), [112](#)
- layout, [67](#), [71](#)
 - `\spread[x]`, [72](#)
 - general options, [69](#)
 - manual line/page breaks, [67](#)
- limitations, [97](#)
 - hard limits, [98](#)
 - soft limits, [98](#)
- line break, [67](#)
 - tie options, [48](#)
- lyrics, *see* **musixlyr**
- M symbol, [76](#)
- manual
 - Cataldi’s Italian **PMX** manual, [II](#)
 - Don Simons’ manual for **PMX**, Version 2.40, [II](#)
- meter, [18](#)
 - blind, [18](#)
 - blind change, [60](#)
 - change, [59](#)
 - printed, [18](#)
- MIDI, [89](#)
 - accidentals, [92](#)
 - file players and editors, [91](#)
 - General Instrument Specification, [93](#)
 - instrument mnemonic names, [92](#)
 - parameters, [90](#)
 - with **scor2prt**, [89](#)
- Molekwa, Mthimkhulu, [106](#)
- Mondrup, Christian, [7](#), [100](#)
- mordent, [38](#)
- Morimoto, Hiroaki, [7](#), [43](#)
- movement breaks, [60](#), [68](#)
- notes, [24](#)
 - chordal, [33](#)
 - dotted note, [26](#)
 - dubious, [38](#)
 - duration, [24](#)
 - not inherited across blocks, [26](#)
 - not inherited for dotted note, [24](#)
 - grace notes, [35](#)
 - horizontal shift, [26](#)
 - parameters, [28](#)
 - pitch, [24](#)
 - explicit octave, [24](#), [25](#)
 - inherited, [25](#)
 - relative, [25](#)

- octavation, 54, 101
 - of clefs, 84
- ornaments, 37, 38
 - not in MIDI, 90
 - repeated, 39
- P symbol, 64, 65, 88
- page
 - breaks, 67
 - headers, 64
 - numbering, 64
 - with **scor2prt**, 89
 - size, 74
- path name, *see* directory
- pause, *see* rest
- pickup bar, 20, 22, 60
- pitch
 - in inline $\text{T}_{\text{E}}\text{X}$, 84
- pizzicato, 38
- polyrhythmic scores, 30, 59
- PostScript, 52, 87, 95
 - fonts, 45
 - slurs, ties and hairpins, 43, 47, 50, 73
- preamble, 16
 - end of, 21
 - numerical parameters, 21
- quavers, *see* eighths
- Quick Reference Table, 1
- R symbol
 - placement at begin of block, 58
- R symbol, 57
- repeats, *see* bars, *see* volte
 - not in MIDI, 90
- rest, 28
 - alignment in 2-voice staves, 73
 - blank rest, 29
 - in beam, 41
 - in xtuplet, 31
 - multibar rest, 29, 89
- S symbol, 88, 113
- segno, 38
- semibreve, 24
- semiquavers, *see* sixteenths
- shake, 38
- short scores, 95
- shorthand
 - for dotted notes, 27
 - for quavers, *see* also tremolo 105, 105
 - for semiquavers, 106
- Sicherman, Col. G.L., 32
- Sicherman-type xtuplets
 - see* xtuplets, 31
- signature, *see* key signature
- Simons, Don, II, 6, 7, 78, 97, 99
- sixteenths, 24, 106
- slur
 - ending on rest, 45
- slurs, 43
 - across line break, 48, 74
 - additional options, 46, 47
 - font-based, 50
 - general usage, 45
 - in grace notes, 35
 - in MIDI, 90
 - PostScript, 43
 - Type K usage, 47
 - Type M usage, 50
 - with staff-jumping beam, 47
- spacing, 72
 - hard space, 75
 - hard space in grace, 36
 - hard space use with **scor2prt**, 88
 - horizontal, 75
 - minimum between notes, 75
 - vertical, 72
- staccato, 38
- staves, 9
 - size, 20
 - varying size, 71, 106
- stems, 26
 - direction forced, 26, 34
 - direction of bass notes, 75
 - length, 26
 - options, 26
- T symbol, 62

- Taupin, Daniel, [6](#), [10](#), [78](#), [96](#)
- Tennent, Bob, [8](#)
- text
- above/below system, [62](#)
 - after final system, [100](#)
- tie
- ending on rest, [45](#)
- ties, *see* psslurs
- titles, [62](#)
- with **scor2prt**, [63](#)
- transposition, [61](#), [71](#)
- not with figured bass, [55](#)
- trill, [38](#)
- tutorial, [1](#)
- conventions of this tutorial, [9](#)
- V symbol, [58](#)
- Van Ryckeghem, Andre, [102](#), [105](#)
- Vogel, Olivier, [95](#), [107](#)
- voice, [10](#)
- several voices in one staff, [17](#), [22](#), [26](#)
- volte, [58](#)
- not in MIDI, [90](#)
 - with **scor2prt**, [59](#)
- W symbol, [74](#), [75](#)
- Werner Icking Music Archive, *see* WIMA
- WIMA, [4](#), [7](#)
- working directory, *see* directory
- X symbol, [75](#), [88](#)
- with **scor2prt**, [76](#)
- xtuplet, [30](#)
- beamed with other notes, [41](#)
 - doubled note, [31](#)
 - doubled note in Bach notation, [31](#)
 - inhibited beaming, [32](#)
 - non-standard bracket, [32](#)
 - rest in xtuplet, [31](#)
 - tweaking the bracket slope, [31](#)
- xtuplets
- adjusting slope with Sicherman xtuplets, [31](#)
 - ending with a rest, [103](#)