xxx Pre-Processor©

Using TEX and the xxx macro package to write parts and scores of music

bèta–Test Version 0.46 – December 15, 1995

Jan Nieuwenhuizen and Han-Wen Nienhuys

If you are not familiar with TeX at all
I would recommend to find another software
package to do musical typesetting.
Setting up TeX and xxx
on your machine and mastering it
is an awesome job which gobbles up
a lot of your time and disk space.

But, once you master it...
HANS KUYKENS

Contents

Chapter 1

M$^{\text{P}}$P: the xxx Pre-Processor

Preface

Dear user,
what you are looking at is the manual to M$^{\text{P}}$P, a preprocessor which is still in bèta-test stage as is xxx, and as long they are bèta, almost everything in M$^{\text{P}}$Pand xxxis subject to change.

This manual is far from completed and, as such, of questionable use at this time, but do have a look at the examples.

1.0.1 About xxx

xxx is a TEX macro package for typesetting music, written by Daniel Taupin. Even though it is still in the β–test phase, it works fine and it has a number of advantages over its predecessor MusicTEX.

As stated in documentation of xxx, xxx is *only* a typesetter, and not intended to interpret some kind of musical notation. Which, in turn, is one of the strong points of xxx. For simple notes direct manual input works fine, but when accents, beam and slurs are added, the input gets more unreadable and error prone (especially with the cummulative accents). Coding a score of music would be a disaster.

1.1 Why a xxx Pre-Processor

The documentation of xxx mentions that setting of a piece of music using xxx 'might appear to be awfully complicated, just as the real keyboard or orchestral music.' This is certainly true. Furthermore the docs state that the only real option that is left for a composer/typesetter, is having 'aesthetic decisions automatically made by somebody (or something) else.'

We cannot completely agree on this one: setting music is not solely subject to aesthetic decisions, there are certain rules which a piece of music must satisfy, e.g. when setting a score of music, the spacing of notes in each staff of a score is coupled to notes in the other staffs.

There are also a lot of assumptions one can safely make that could simplify music input, but do not imply any major aesthetic decisions, for example the relative spacing of notes, the drawing of beams and slurs, the placing of accents and directions. The complexity of a xxx input file can be seriously reduced without any loss of information.

With this in mind a new music definition language was developed.

Compared to xxx M^{P}P is a quite simple program, but it has the potential to make xxx much more *accessible* if not *useable*. Its goal is not to cover all of xxx's possibilities, but to make the coding of simple pieces (or parts) of music a lot easier.

Chapter 2

The MPP musical definition language

2.1 Introduction

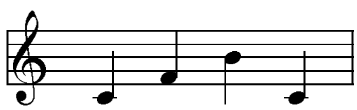
The MPP input language is mostly visually based and should be relatively easy to read (and write).

This means that:

- Notes are coded by their names.
- Most other musical definitions such as duration, bars, accents, beams and slurs are represented by single characters.
- Typesetting details are computed by MPP and can be overruled for special cases.
- Each staff of music is coded in its own `.mpp` file. Several `.mpp` input files can be compiled to create a multiple-staff part or score.
- To enhance the readability of the music, comments and whitespace may be inserted at any point between notes.

As examples are often found to be more fun than technical descriptions that nobody reads anyway, the examples come first, and you will be bored with the definitions afterwards.

Here is a simple piece of music:



which was coded as:

```
c f b c
```

2.2 The notes

Notes are the basic building blocks of music. In MPP You can produce notes by specifying their pitches and lengths, and optional "features". The pitch is specified by a note-name (a, b, etc.) and an octave. In the above example all notes lie in the same octave, and have the same duration, a quart. There should be no spaces or comments in a note.

The piece of music shown above is perhaps a bit too simple to be taken seriously, but it is a start. In the following sections the examples might gradually become more interesting.

2.2.0.1 The duration

A duration is specified by a number: the reciprocal value of its length. For a pointed notes you can also type a number of periods (.) following the length.

The duration immediately follows the notename.

Example:

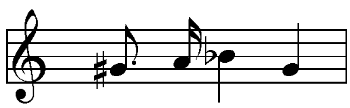
which was coded as:

```
g8. a16 b g4
```

As you can see. The third note has no duration specified, and MPP makes its length the default length, is a quart. If you omit the duration of a note, MPP uses the value of `\duration`. You can change this default duration at any time with the macro `\duration`: to change the default duration of notes to, say, sixteenth `\duration{16}`.

2.2.0.2 The pitch

The pitch of a note is determined by its name and the octave where MPP should produce the note.



which was coded as:

```
gis8. a16 bes gis4
```

As you can see, MPP correctly puts accidentals in, depending on the key of the piece. You can set the `\key` with `\key\G`. Supported keys are from C flat (7 flats) to C sharp (7 sharps), since double-sharps are as yet unimplemented.