ABC2MTEX

An easy way of transcribing folk and traditional music

Version 1.0 – December '93

Chris Walshaw
School of Maths,
University of Greenwich,
London, G.B.
email: `C.Walshaw@gre.ac.uk`

Contents

1 Introduction

Welcome to **ABC2MTEX**!

This is a package designed to notate tunes stored in an ascii format or `abc` notation. It was designed primarily for folk and traditional tunes of Western European origin (such as Irish, English and Scottish) which can be written on one stave in standard classical notation. However, it should be extendible to many other types of music.

The package is small (about 60 KByte), easy to use and features, amongst other things, the ability to transpose both the music and the `abc` notation. It will also create an index of all the tunes you have transcribed.

The ability to write tunes in `abc` notation means that they can be easily and portably stored or transported electronically. This package allows you to typeset them easily too. In addition the notation is not specifically tied to MusicTEX and so similar packages could be written to translate to your favourite typesetting program.

This package has, however, been written on top of MusicTEX, Daniel Taupin's music typesetting package, itself written on top of TEX, Donald Knuth's typesetting package. To run it you will need TEX, MusicTEX (available by ftp from a number of sources), a C compiler and access to a postscript printer.

To get a copy of the package send an email request to `C.Walshaw@gre.ac.uk` . Also mail me here with any comments or suggestions for future versions

1.1 ABC2MTEX: what you get

The following files come in the package:–

Makefile – a makefile!

abc2tex.c – C code which converts the `abc` notation into MusicTEX. This code is fairly fragile and care should be taken if changes are made to it.

output.c – C code which formats the information fields for each tune (see section 2.1). This code is fairly robust and can be changed without too much difficulty.

userguide.tex – this document in LaTEX format.

header.tex – a header file for the MusicTEX output.

zmacros.tex – local macros built on top of the MusicTEX macros.

XE, *XIJ*, *XIR* & *XSSs* – example tune files.

1.2 MusicTEX: what you will need

You will need the macro packages *musictex.tex*, *musicnft.tex*, *musictrp.tex* and *musicvbm.tex* together with the **.tfm* files and **.pk* files. You will also need the macro packages *musicpre.tex* and *musicpos.tex* although only to run LaTEX on this document. Note that it only seems to work if the **.pk* files are changed to **.300pk* files. The version of MusicTEX that I have is 4.98 from the file *musictex-498.tar.Z* available at various ftp sites.

To set this up with TEXversion 3.14 or 3.141 you should set the following environment variables:–

```
setenv TEXFONTS  $TEXFONTS\:<tfm_directory>
setenv TEXINPUTS $TEXINPUTS\:<macro_directory>
setenv TEXPKS    $TEXPKS\:<pk_directory>
setenv XDVIFONTS $XDVIFONTS\:<pk_directory>
```

where `<macro_directory>` is the pathname of the directory the macros are stored in, etc. For older versions of TEX, such as 2.93 this is a much messier procedure.

If you have problems setting up MusicTEX or modifying it for your own use, don't mail me – I'm no expert. Instead there is a mailing list which you can subscribe to by sending an email to `mutex-request@stolaf.edu` .

1.3 Acknowledgements

I would like to thank John Walsh (`walsh@math.ubc.ca`) for all his helpful advice and suggestions when getting this package together.

2 Tune Files

The music can either be stored one tune per file with a name that reflects the contents, e.g. the tune 'Paddy O'Rafferty' in the file *Paddy_ORafferty*, or grouped together in files according to some common theme, e.g. *XIR* contains Irish reels and *XE* has English tunes of all types. If the tunes are grouped then an index is useful and indeed is provided here (see section 3.3). However to use this indexing facility you must use the following naming convention for the files. The first letter should be *X*, the second should be the origin of the tunes (*I* – Irish, *E* – English, *S* – Scottish, etc.) and the third & fourth (optional) the type of tune (*R* – Reel, *J* – Jig, *SJ* – Slip Jig, *Ss* – Strathspey, etc.). Each file is then structured as follows:

```
M:6/8

X:1
T:First tune name, The
      :
    optional information fields
      :
K:D
ABC DEF|abc def| ...

X:2
T:Second tune name, The
      :
    optional information fields
      :
K:D
ABC DEF|abc def| ...

 :
```

Each file should start with a default meter (e.g. `M:6/8` for jigs) and is followed by the tunes separated from each other by a blank line. Each tune should start with an `X` (reference number) field followed by a `T` (title) field followed by optional fields (see below). The last field should be a `K` field and then the `abc` notation should follow immediately after.

If, on the other hand, you choose to store tunes one per file then you need the `M`, `X` & `T` fields (in that order), followed by the optional fields, followed by the `K` field followed by the `abc` notation. No indexing is provided as a simple `ls` command should reveal all.

2.1 Information fields

The following information fields can be used:–

- `X` – tune reference number.
- `T` – tune title. Some tunes have more than one title and so this field can be used more than once per tune – the first time will generate the title whilst subsequent usage will generate the alternatives in small print. For the purposes of indexing definite & indefinite articles should be placed at the end after a comma – e.g. 'The Tenpenny Bit' should be entered as `T:Tenpenny Bit, The`, although this will appear as 'The Tenpenny Bit' on

the printed music sheet. The normal TeX accents (such as in `caf\'e` for producing café or in `makazi\u ce` for producing makazīče) can be used but will be stripped out of the index.

`C` – composer (optional). Most of my tunes have `C:Trad`.

`S` – source, i.e. where the tune came from (optional).

`N` – notes, i.e. anything else of interest (optional).

`M` – meter, i.e. `M:4/4`, `M:6/8`, ... (optional – default meter given on the first line of the tune file). In addition `M:C` or `M:C|` give common time and cut time respectively. This field can be also used in the middle of a tune for changing meter (see section 2.2.10).

`O` – origin (1 letter), e.g. `O:I` – Irish, `O:E` – English, `O:S` – Scottish, ... (optional - default generated by the 2nd letter of the tune file name).

`R` – tune type (1 or 2 letters), e.g. `R:R` – reel, `R:J` – jig, `R:SJ` – slip jig, ... (optional - default generated by the 3rd & 4th letters of the tune file name).

`Z` – transcription notes (optional). This field generates a message when TeX is running but no other output.

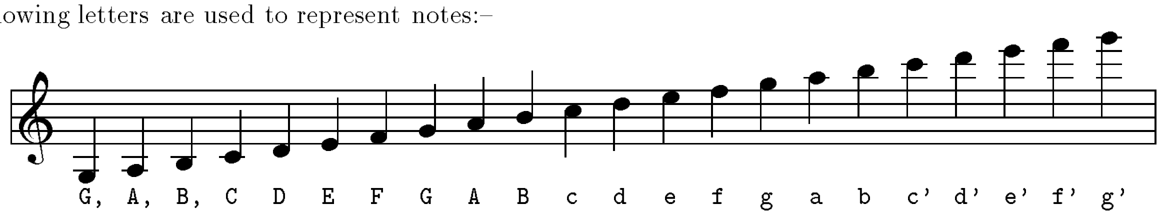
`E` – elemskip (optional). This changes the internote spacing, see section 2.2.17.

`K` – key, e.g. `K:D`, `K:G`, `K:Am`, ... This field signals the start of the tune and so must be the last field used. It can, however, also be used in the middle of a tune for changing key (see section 2.2.9).

2.2 `abc` tune notation

2.2.1 Notes

The following letters are used to represent notes:–



They can be modified in length – see section 2.2.3.

2.2.2 Rests

Rests are generated with a `z` and can be modified in length in exactly the same way as notes can (see below).

2.2.3 Note lengths

Each meter has a default note length and a single letter in the range `A-G`, `a-g` will generate a note of this length. For example, in 3/4 the default note length is a quaver and so `abc` represents 3 quavers. The following meters are supplied (the 's' and 'l' modifiers respectively double or half the default note lengths):–

Meter	Default Note Length	Maximum Notes Per Beam
`2/4`	semi-quaver	4
`2/4s`	quaver	2
`3/2`	crotchet	1
`3/2l`	quaver	3
`3/4s`	crotchet	1
`3/4`	quaver	3
`3/4l`	semi-quaver	6
`3/8s`	quaver	3
`3/8`	semi-quaver	6
`4/4s`, `Cs`, `C|s`	crotchet	1
`4/4`, `C`, `C|`	quaver	4
`4/4l`, `Cl`, `C|l`	semi-quaver	4
`5/4`	semi-quaver	4
`6/8`	quaver	3
`6/8l`	semi-quaver	6
`7/4`	quaver	4
`9/8`	quaver	3
`9/8l`	semi-quaver	6
`12/8l`	semi-quaver	6

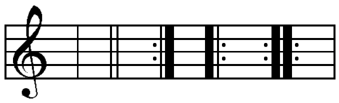
Other notes can then be obtained by simply putting a multiplier after the letter. Thus in 2/4, `A` is a semi-quaver, `A2` a quaver, `A3` a dotted quaver, `A4` a crotchet, `A6` a dotted crotchet, `A8` a half note, and so on, whilst in 3/4 `A` is a quaver, `A2` a crotchet, `A3` a dotted crotchet, `A4` a half note, ...

2.2.4 Beams

To group notes together under one beam they should be grouped together without spaces in the tune file. Thus in 2/4 `A2BC` will produce a quaver followed by two semi-quavers under one beam, whilst `A2 B C` will produce the same notes separated. The beam slopes and the choice of upper or lower staffs are generated automatically. Note that each meter has a maximum number of notes that can be grouped under one beam (specified in the above table). Thus in 6/8, where the maximum number is 3 quavers, `abc def` will produce exactly the same output as `abcdef` – two groups of 3 quavers.

2.2.5 Repeat/bar symbols

The symbols '`|`', '`||`', '`:|`', '`|:`' & '`::`' are used to generate the following bars and repeats respectively:–



2.2.6 First & Second Repeats

First & second repeats can be generated with the symbols `[1` & `[2`. These should come after the bar or repeat symbol, i.e. `faf gfe|[1 dfe dBA:|[2 dfe dcB||`.

2.2.7 Accidentals

Just as in MusicTeX the symbols `^`, `=` and `_` are used (before a note) to generate respectively a sharp, natural or flat.

2.2.8 Line Breaking

Generally one line of `abc` notation will produce one line of music. If the music is too long it will overflow onto the next line and you will have to either change the internote spacing (see section 2.2.17) or the tune file. If, however, you wish to use two lines of input to generate one line of music (see, for example, the 'Untitled Reel' in *XIR*) then simply put a \ at the end of the first line. This is also useful for changing meter or key in the middle of a line of music.

2.2.9 Changing key mid-tune

To change key simply put in a line with a key (`K:`) field, e.g.

```
ed|cecA B2ed|cAcA E2ed|cecA B2ed|c2A2 A2:|
K:G
AB|cdec BcdB|ABAF GFE2|cdec BcdB|c2A2 A2:|
```

To do this without generating a new line of music, put a \ at the end of the first line, i.e.

```
ed|cecA B2ed|cAcA E2ed|cecA B2ed|c2A2 A2:|\
K:G
AB|cdec BcdB|ABAF GFE2|cdec BcdB|c2A2 A2:|
```

2.2.10 Changing meter mid-tune

This is accomplished in exactly the same way as changing key, only with a meter (`M:`) field (see 'Old Sir Simon the King' in *XE* for an example of changing meter for a bar.

2.2.11 Ties

You can tie two notes together across a bar with a `-` symbol immediately before the bar, e.g. `abc-|cba`.

2.2.12 Triplets and Quadruplets

These can be simply coded with the notation `(3abc` for a triplet or `(4abcd` for a quadruplet. At the moment they don't have a tie marker, just a `3` or `4` by the beam, but I'm working on it.

2.2.13 Gracings

With regard to gracings, I fall in the Irish music camp which says that you transcribe gracings as little as possible and leave it up to the players to make their own interpretation. Thus the only gracing I tend to use is to put a tie/slur marker under a note which will generally mean a roll, cran or stacatto triplet. This is achieved by putting a `~` before the note.

However, to explicitly write out every grace note, just put them in curly braces, {}. For example, a taorluath on the Highland pipes would be written {`GdGe`}. To make things simpler the common Highland gracings could then have macros written for them, for example `{\T}` for a taorluath instead of {`GdGe`}, or `{\B}` for a birl or {`AGAG`}. However, I leave the task of setting these macros to someone with more experience of Highland ornamentation than myself. The tune 'Athol Brose' (number `2` in the file *XSSs*) has an example of complex Highland pipe gracing in all its glory.

2.2.14 Staccato Notes

Staccato marks (a small dot above or below the note head) can be generated by a dot before the note, i.e. a staccato triplet is written as `(3.a.b.c` . Rolls and staccato markers cannot be combined on one note.

2.2.15 Chords

By this I mean putting more than one note head per stem. This seems to occur so rarely in traditional music tune books that I haven't made any explicit provision for it. It is *possible* by using the facility to directly input TEX (see section 2.2.18) and the MusicTEX macro `\zh`. However, it is very messy and the resulting `abc` notation is not at all easy to read. See the tune 'Kitchen Girl' in the file *XIR* for an example.

2.2.16 Order of Symbols

The order of symbols for one note is `<roll/staccato marker><accidental><note><octave><note length>`, i.e. `~^c'3` or `.=G,2`

2.2.17 Internote spacings

The internote spacing is set by the MusicTEX command `\elemskip`. However this can be stored with the rest of the tune using the information field `E`. As the format is set up now, I use `E:8` & `E:7` to squeeze long tunes up a bit and `E:10` and above to stretch short tunes. Using `E:6` really looks too cramped to my eye.

2.2.18 TEX Input

If there is a line beginning with a \ inside a tune, it is put directly into the output file (*music.tex*).

2.3 Examples

Examples are provided for most of the possibilities above. They can be found in:–

Example	File & Number	Relevant subsection
First & Second Repeats	`XIJ:1`	2.2.6
Line Breaking	`XE:1`	2.2.8
Changing key mid-tune	`XIR:1`	2.2.9
Changing meter mid-tune	`XE:2`	2.2.10
Triplets and Quadruplets	`XSSs:1`	2.2.12
Gracings	`XSSs:2`	2.2.13
Chords	`XIR:2`	2.2.15
Internote spacings	`XSSs:2`	2.2.17
TEX Input	`XIR:2`	2.2.18

3 Running the format program

To make the format program first type `make`. Then run it with the command `format`. You will be prompted to choose a format by:–

```
select format:
        x - xbook
        m - music
        t - music (transposable)
        i - index
        n - notes
```

The formats `x`, `m` and `t` are essentially the same, all producing MusicTEX output in the file *music.tex*. However, the `x` format produces tune numbering and the details in the `S:` & `N:` fields, while the `t` format prompts to transpose every tune (see section 3.2). The `i` format makes an index of all the *X* tune files in the file *index*. The `n` format does nothing at present; at some stage it would be nice to have an index of `abc` notation, so that if you know a tune begins `abc def` you can look up the title and the rest of the tune. However, I haven't got round to this yet.

As an alternative to making this selection you can simply put your chosen format as a command line input, e.g. run the program with the command `format x` for the `x` format.

Having selected the format you will then be prompted to choose tunes with the line

```
select tunes:
```

Typically you might respond with `XE:1-3,5,9-` which will select tunes numbered 1, 2, 3, 5 and 9 onwards from the file *XE*. Just entering `XE` or `XE:-` will select all the all the tunes from *XE*. Note that there should be no spaces anywhere in this input.

When the program has finished formatting the chosen tunes (a matter of seconds) it will prompt you again with a `select tunes`. To finish the program simply hit return at this prompt.

3.1 TEX Input

If at the prompt

```
select tunes:
```

you enter a line beginning with a \, it is put directly into the output file (*music.tex*). This is a good way of putting things like page headings directly into the document, e.g. something like

```
\headline{\sl ENGLISH TUNES \hfil \folio}
```

will generate a header and page number for each page.

3.2 Transposing Tunes

If you select the format `t`, you will be prompted at every tune by

```
  Transpose <tune name>:
```

You should enter here the change in the number of sharps and flats you wish to make. For example, if the tune is in G (1 sharp) and you wish to transpose it up to A (3 sharps) you would enter `+2` or just `2`. Flats are counted negatively (as in MusicTEX) so that to transpose from G to F (1 flat or -1) you would enter `-2`.

If you have chosen to transpose the tune, i.e. entered anything other than `0` you will then be prompted by

```
  Note offset:
```

which is just the number of line and spaces that each note must move. Thus going from G to A is just moving every note up one and so you would enter `+1` or `1` whilst G to F would require `-1`. Note that something like changing from E$\flat$ (3 flats) to E (4 sharps) has note offset `0`.

As well as generating transposed MusicTEX output in the file *music.tex*, the `T` format also generates transposed `abc` notation to the screen. This is useful if you want to save the transposed version in the tune file. The only drawback of this process is that it won't transpose accidentals (as I haven't come up with a good algorithm for doing this yet) although it generates a warning if any are present.

3.3 Making an Index

To make an index of all the tunes stored in X files simply type `make index`. The result is stored in the file *index*.

3.4 Tidying Up

To tidy up at the end of a session simply type `make clean`, although be warned that this will remove the files *music.tex* and *index*.

4 Bugs and Features

4.1 Things missing when running TEX

If you get a complaint from TEX looking a bit like

```
 missing \enotes \ mesure 1 bar
```

or even

```
! Missing } inserted.
<inserted text>
                }
<to be read again>
                   \end
l.25 \end

?
```

you have probably forgotten to end the previous line of `abc` notation (possibly in the previous tune) with a bar, double bar or repeat symbol.

4.2 Undefined control sequences when running TEX

The format program generates output in the form of `\Z` macros which are then defined in terms of MusicTEX macros in the file *zmacros.tex* (which is automatically read in when processing *music.tex*). However my approach to defining the `\Z` macros has been to do it as and when needed and so some combinations of notes and accidentals do not yet been written. If you get a complaint from TEX looking a bit like

```
! Undefined control sequence.
l.24 \notes\Zl0a0a0a0a
                      {^j}{^j}{^j}{^j}\enotes
?
```

then it is probably that the `\Z` macro hasn't been defined yet. This is likely to happen with complicated gracings occurring within a beam and also with beams which have a large number of accidentals (where a large number is anything greater than or even equal to one).

There are two solutions to this problem – either get in there and define it for yourself (and let me know so I can include it in later releases), or mail me and I'll do it (as soon as I get round to it).

4.3 Other Bugs

If (when) you find any other bugs, let me (`C.Walshaw@gre.ac.uk`) know and I'll try to sort them out.