The abc package*

Enrico Gregorio
Enrico dot Gregorio at univr dot it

2016/05/15

1 Introduction

There are several ways to set music using TEX, notably MusixTEX and Lilypond. Both are very powerful and, consequently, a bit difficult to learn and to use.

From the point of view of notation, the ABC system[1] is much simpler. A recent extension of this language, called ABC Plus[2], allows for setting multiple staves and polyphony. One of the best programs for converting these notations into sheet music is `abcm2ps`, which can take an ABC or ABC Plus file and transform it into a PostScript™ file.

The purpose of this package is to allow LATEX users to include in their documents small excerpts of music written directly in ABC (Plus). It exploits the `\write18` technique available with the Web2C implementation of the TEX system and free utilities like `ps2eps`, `ps2epsi` and `epstopdf`. From version 2.0b we check the existence of the shellesc package in order to be compatible with LuaTEX.

This package can be used both with LATEX and pdfLATEX, without any change in the user's source file. It employs also the package keyval by David Carlisle and ideas from the verbatim package in the LATEX tools.

Martin Tarenskeen wrote me about possible support of Mup–Music Publisher[3] and actually it was easy to add it along with some improvements partly suggested by him. Therefore the package now comes along with a new mup package; see section **??**. I owe many thanks to Martin for testing the new version.

We are studying whether it is feasible to extend support also to other music printing programs like Lilypond.

2 Usage

`abc` The only environment provided by this package is `abc` with the following syntax:

```
\begin{abc}[⟨keyword⟩=⟨value⟩...]
⟨ABC Plus material⟩
\end{abc}
```

*This document corresponds to abc v2.0b, dated 2016/05/15.

[1] `http://staffweb.cms.gre.ac.uk/~c.walshaw/abc/`

[2] `http://abcplus.sourceforge.net`

[3] `http://www.arkkra.com`

The environment should be used only in LR-mode. Its output is set in a 'center' environment. We give a list of the available keywords.

`name=`⟨*name*⟩: ⟨*name*⟩ is a name for the temporary file which will be output and massaged by `abcm2ps`. *Warning*: the extension `.abc` is automatically added to the file name; esistent files with the same name will be silently erased. If the keyword is specified without a value, then the output file receives a unique name.

`options=`⟨*options*⟩: ⟨*options*⟩ are command line parameters to the `abcm2ps` utility; the default are '`-O= -c`'.

`postoptions=`⟨*postoptions*⟩: ⟨*postoptions*⟩ are possible command line options which, in `abcm2ps` syntax, go after the file name.

`program=`⟨*program*⟩: ⟨*program*⟩ is used to specify an alternative program to `abc2mps` (if existent). In this case users must specify completely the command line options, directly in ⟨*program*⟩ or with ⟨*options*⟩ and ⟨*postoptions*⟩. For example

```
\begin[name=song,program=abctoeps,options={-a -b}]
\begin[name=song,program={abc3ps -a -b}]
```

(assuming there is an `abc3ps` program).

`width=`⟨*width*⟩: ⟨*width*⟩ should be a dimension; it is best to express it as a fraction of `\abcwidth`.

`center`: This is a boolean, either true or false; the default is true, if left flush alignment is preferred, specify `center=false`.

`extension`: This keyword seems practically useless, but is needed if one needs to input both ABC and Mup files in one and the same document. See later on.

`\abcinput` It is also possible to input an available ABC file. The syntax is

`\abcinput[`⟨*options*⟩`]{`⟨*name*⟩`}`

where ⟨*name*⟩ is the name of the file, without the extension which should be `.abc`. In the optional argument users can put keyword-value pairs as for the environment. Of course the keyword `name` is ignored here.

`\abcwidth` Users have a minimum control (at least in this version) on how to include some lines of music. The only parameter they can modify is `\abcwidth` with `\renewcommand`. Its normal value is `\linewidth`. The best thing to do is

`\renewcommand{\abcwidth}{`⟨*fraction*⟩`\linewidth}`

where ⟨*fraction*⟩ is some number between 0 and 1. Changing this parameter affects every subsequent music inclusion, obeying to the usual scoping rules. The width can be changed locally for the environment or the command with the method explained before.

`\normalabcoutputfile` The name for the generic output files is "`out-abc`". In the (improbable) case that some file `out-abc.`⟨*ext*⟩ is present in the directory, users can redefine `\normalabcoutputfile` as they like.

3 Package options

There are some package options, to control what is to be passed for subsequent processing.

3.1 `shellescape` and `noshellescape`

The option `shellescape` (default) means that some external programs will be called by LaTeX to get the inclusion of the music lines. If you don't trust the automatic generation, or your TeX implementation does not allow the trick with `\write18`, then use `noshellescape`. In this case, a name should be specified for every 'abc' environment, because otherwise ABC output files will be overwritten, since they receive the same generic name, by default. A file named `out-abc.sh` is produced, containing the commands to give for elaborating the music files.

3.2 `generate` and `nogenerate`

With the `generate` option (default) the ABC lines will be processed by the external programs. The `nogenerate` option can be used when the ABC sources have not changed, in order to increase speed. Also in this case, however, a name should be specified for all output ABC files.

3.3 `ps2epsi`, `ps2epsidos` and `ps2eps`

The option `ps2eps` (default) means that the `ps2eps` Perl script will be used for generating the EPS file for graphic inclusion. Specify the `ps2epsi` option if you prefer the `ps2epsi` program; specify `ps2epsidos` if you are on a system where the utility `ps2epsi` generates a file with three letter extension `.epi`.

3.4 `nosaveall` and `saveall`

The first is the default, but the second is called implicitly when the `noshellescape` is given. When the `saveall` option is active, every 'abc' environment produces a unique output file; in other words, the `name` keyword, without value, is given for every environment. So the `noshellescape` option along with `generate` will not overwrite the output from unnamed environment. It is possible that, during the document's preparation, the numbers added to the default file name are out of synch, the process should converge. Note that with LuaTeX the `nosaveall` option can produce errors, so `saveall` is enabled by default, with this engine.

4 Compiling documents

Users must give the option `-shell-escape` when compiling their documents, unless they chose the `nogenerate` package option or the `noshellescape`. Thus one of

```
latex -shell-escape ⟨TEX file name⟩
pdflatex -shell-escape ⟨TEX file name⟩
simpdftex latex --extratexopts "-shell-escape"
```

should be used from the command line (or equivalent way, depending on operating system and distribution).

5 An example file

```
\documentclass[a4paper,12pt]{article}
\usepackage[generate,ps2eps]{abc}
\usepackage{mathptmx}

\begin{document}

\title{Example of ABC Plus in \LaTeX{}}
\author{Guido Gonzato}
\date{}
\maketitle

This is a short piece.

\medskip

\begin{abc}
X:4
T:Cronin's Hornpipe
R:hornpipe
S:Keenan and Glackin
E:7
M:C|
L:1/8
K:G
BA|GABc dBde|gage dega|bage dBGB|cABG A2BA|!
GABc dBde|gage dega|bage dBAB|G2G2 G2:|!
fg|afd^c d2ga|bged e2ga|(3bag  (3agf gedB|(3cBA AG AcBA|!
GABc dBde|~g3e dega|bage dBAB|G2G2 G2:|!
\end{abc}

\medskip

This is another short piece, but we would like to keep the
ABC source in our directory.

\begin{abc}[name=jacky]
X:9
T:Jacky Tar
R:hornpipe
M:4/4
L:1/8
K:Edor
(Bd) | "Em"  e2 ed efge | "G" d2 B2 B2 (dB) | "D" ABde faef |  d2 A2 A2 (Bd) |
       "Em" e2 ef g2 fe | "G" dB GB d2 (cB) | "D" AGFE DEFA | "Em" G2 E2 E2  :|
(GA) | "Em" BGEG   BGEG |     BAGF  E2 (FG) | "D" BGEG BGEG |   AGFE D2 (EF) |
       "G" GFGB g2 (fe) |     dBGB  d2 (cB) | "D" AGFE DEFA | "Em" G2 E2 E2  :|
\end{abc}

\clearpage

And, finally, we want to set also the last piece; its ABC code
is already in our directory.

\medskip

\abcinput{poll}
```

```
\end{document}
%
```

This file is accompanied by a file `poll.abc`

```
X:12
T:Poll Ha'penny
T:Garra\'i na bhF\'eile\'og
R:hornpipe
H:The Irish title means "The Garden of Honeysuckles"
D:Mary Bergin: Feadoga Stain
D:Noel Hill agus Tony McMahon: I gCnoc na Grai
Z:id:hn-hornpipe-26
M:C|
L:1/8
K:Amix
(3GAB | =cAAG A2 (3AB=c | (3d=cB (3AGF G2 (3B^cd | ed^cA    d^cAG | ~A3 G A2 de |
        ~f3 d     ~e3 c | d2     (3Bcd      efge | aged (3=cBA GB | ~A3 G     A2:|
|: ef | ~g3 f      gfef | g2 ga              gedg | eaag      a3 g | eaag  a2 ag |
        ~f3 d     ~e3 c | d2     (3Bcd      efge | aged (3=cBA GB | ~A3 G    A2 :|
%
```

6 Mup support

Everything we have said about ABC translates verbatim for Mup. Simply call the package with

```
\usepackage{mup}
```

with options just like before, and substitute the string 'mup' to every occurrence of 'abc' in the preceding sections. The only differences are in the default command line options when calling the external program (they are `-F` for 'mup').

It is even possible to use both external programs in the same document. If this is desired, call the `abc` package and define a new environment for Mup inclusion as follows

```
\newenvironment{mup}[1][]
  {\renewcommand{\normalabcoutputfile}{out-mup}%
   \abc[program=mup,options={-F},extension=mup,#1]}
  {\endabc}
\newcommand{\mupinput}[2][]{%
  \abcinput[program=mup,options={-F},extension=mup,#1]{#2}}
```

Here is an example with Mup.

```
\documentclass[a4paper,12pt]{article}
\usepackage[generate,ps2eps]{mup}
\usepackage{mathptmx}

\begin{document}

\title{Example of MUP in \LaTeX{}}
\author{Martin Tarenskeen}
\date{}
\maketitle

This is a short piece.
```

```
\medskip

\begin{mup}
1: a-;b-;c;d;
bar
1: e;f;g;a;
endbar
\end{mup}

\medskip

This is another short piece, but we would like to keep the
MUP source in our directory.

\begin{mup}[name=mymup]
score
staffs=2
time=6/8
beamstyle=4.,4.

staff 2
clef=bass

music
1: 8c;d;e;f;g;a;
2: 4.ceg;cfa;
repeatend

\end{mup}

\clearpage

And finally, we want to set also the last piece; its MUP code
is already in our directory.

\medskip

\mupinput{simple}

\end{document}
%
```

This file is accompanied by a file `simple.mup`

```
score
staffs=2
time=6/8
beamstyle=4.,4.

staff 2
clef=bass

music
1: 8c;d;e;f;g;a;
2: 4.ceg;cfa;
repeatend
%
```

7 The implementation

After the usual stuff of package presentation, here are the actual macros. To begin with the option declarations and the defaults. The first options are boolean.

```
⟨*package⟩
\RequirePackage{ifluatex}
\newif\ifabc@shellescape
\newif\ifabc@generate
\newif\ifabc@warning
\newif\ifabc@saveall
\newif\ifabc@mup
\DeclareOption{mup}{\abc@muptrue}
\DeclareOption{noshellescape}{\abc@shellescapefalse\abc@warningtrue
  \abc@savealltrue}
\DeclareOption{shellescape}{\abc@shellescapetrue}
\DeclareOption{nogenerate}{\abc@generatefalse}
\DeclareOption{generate}{\abc@generatetrue}
\DeclareOption{nosaveall}{\abc@saveallfalse}
\DeclareOption{saveall}{\abc@savealltrue}
```

The following options control the external programs to use.

```
\def\abc@epsext{eps}
\DeclareOption{ps2eps}{\def\abc@pscmd{ps2eps -f}}
\DeclareOption{ps2epsi}{\def\abc@pscmd{ps2epsi}\def\abc@epsext{epsi}}
\DeclareOption{ps2epsidos}{\def\abc@pscmd{ps2epsi}\def\abc@epsext{epi}}
```

Now we declare the default options and call the user specified ones. Since LuaTEX doesn't accept loading different PDF files with the same name, when this engine is used, the `saveall` option is enabled by default.

```
\ExecuteOptions{generate,shellescape,nosaveall,ps2eps}
\ifluatex
  \ExecuteOptions{saveall}
\fi
\ProcessOptions\relax
```

Then we have to load some packages we need. The first one is to do verbatim output to a file without reinventing the wheel. Then the package for implementing keyword-value options; we have to take care of graphics inclusion, and to control whether we are using LATEX with DVI or PDF output.

```
\RequirePackage{verbatim}
\RequirePackage{keyval}
\RequirePackage{graphicx}
\RequirePackage{ifpdf}
```

Next we define some internal commands. First of all a boolean for issuing messages if necessary and a counter to assign unique names to output files

```
\newif\ifabc@unprocessedfiles
\newcounter{abc@count}
```

We choose to give explicitly the extensions to the graphics files, since some user could prefer `ps2epsi`. Everything is doubled for Mup support.

mup
\mupinput

```
\ifabc@mup
\newcommand{\abc@cmd}{mup}       % virtually no choice
```

```
\newcommand{\abc@parm}{-F}       % -F MUST stay
\newcommand{\abc@epstopdfcmd}{epstopdf}
\newcommand{\abc@pdfext}{pdf}
\def\normalabcoutputfile{out-mup}
\def\normalmupoutputfile{\normalabcoutputfile}
\def\mup{\abc}
\def\endmup{\endabc}
\def\mupinput{\abcinput}
\def\abc@ext{.mup}
\def\abc@packagename{mup}
\else
\newcommand{\abc@cmd}{abcm2ps}       % virtually no choice
\newcommand{\abc@parm}{-O= -c}       % -O= MUST stay
\newcommand{\abc@epstopdfcmd}{epstopdf}
\newcommand{\abc@pdfext}{pdf}
\def\normalabcoutputfile{out-abc}
\def\abc@ext{.abc}
\def\abc@packagename{abc}
\fi
\def\abc@tempfile{\normalabcoutputfile}
\def\abc@opt{}
\let\abc@postopt\@empty
\ifpdf
  \let\abc@finalext\abc@pdfext
\else
  \let\abc@finalext\abc@epsext
\fi
\newif\ifabc@center
\abc@centertrue
```

The following is the only parameter the user is authorized to tamper with; it has an alias for Mup.

```
\newcommand{\abcwidth}{\linewidth}  % only fractions of \linewidth
\let\mupwidth\abcwidth
```

`\abc@startgen` `\abc@finishgen`

Now something directly borrowed from the package verbatim. We declare an output stream and define two macros which will be called by the `abc` environment or by the `\abcinput` command in case we are generating the graphics files. The macro `\abc@startgen` then passes the control to `\abc@process` which is different, according to the options given to the package.

```
\newwrite\abc@out
\def\abc@startgen{%
  \@bsphack
  \immediate\openout\abc@out\abc@tempfile\abc@ext
  \let\do\@makeother\dospecials
  \catcode`\^^M\active \catcode`\^^I=12
  \def\verbatim@processline{%
  \immediate\write\abc@out
    {\the\verbatim@line}}%
  \verbatim@start}
\def\abc@finishgen{%
  \immediate\closeout\abc@out
  \@esphack
  \abc@process
```

```
}
```

`\abc@doshellcommand`
`\abc@nodoshellcommand`

We define a macro for the external massaging of the ABC files and another one for the case the user doesn't trust or have available the `\write18` trick; the second one spits out a very simple shell script which can be used to take care of the compilation; this file is probably compatible with all systems having a command line interface. Then we check the options again and define the commands that really do the job.

```
\ifluatex
  \IfFileExists{shellesc.sty}
    {\RequirePackage{shellesc}\let\abc@shell\ShellEscape}
    {\def\abc@shell{\immediate\write18}}
\else
  \def\abc@shell{\immediate\write18}
\fi
\def\abc@doshellcommand{%
  \abc@shell{%
    \ifabc@mup
      \abc@cmd\space
      \abc@opt\space
      \abc@parm\space
      \abc@tempfile\abc@ext\space
    \else
      \abc@cmd\space
      \abc@parm\space
      \abc@opt\space
      \abc@tempfile\abc@ext\space
    \fi
    \ifx\abc@postopt\@empty
    \else\space\abc@postopt\fi
  }%
  \abc@shell{%
    \abc@pscmd\space\abc@tempfile.ps
  }%
  \ifpdf
    \abc@shell{%
      \abc@epstopdfcmd\space\abc@tempfile.\abc@epsext
    }%
  \fi
}
\def\abc@nodoshellcommand{%
\immediate\write\abc@outsh{%
  \abc@cmd\space
  \abc@parm\space
  \abc@opt\space
  \abc@tempfile\abc@ext\space
  \ifx\abc@postopt\@empty
  \else\space\abc@postopt\fi}%
\immediate\write\abc@outsh{%
  \abc@pscmd\space\abc@tempfile.ps}%
\ifpdf
  \immediate\write\abc@outsh{%
    \abc@epstopdfcmd\space\abc@tempfile.\abc@epsext}%
\fi
```

```
}
```

We use a conditional to emit a message at the end of the compilation if some file has not been found and the `nogenerate` option was chosen.

```
\AtEndDocument{%
  \ifabc@warning\ifabc@unprocessedfiles
    \PackageWarningNoLine{\abc@packagename}{%
      \ifabc@shellescape
        You have set the 'shellescape' option, but you ran%
        \MessageBreak
        (pdf)latex without the '-shell-escape' command line%
        \MessageBreak
        option. Fix it either with the 'noshellescape' option%
        \MessageBreak
        in your document or the correct call of (pdf)latex%
      \else
        Remember to generate the [eps,pdf] files before compiling%
        \MessageBreak
       again. Use the file \abc@tempfile.sh for a list or as a script%
      \fi}%
  \fi\fi}
```

Now we define the macro responsible for the massaging of the ABC files. This is a good moment for inizializing the writing of the shell script, when needed.

`\abc@process`

```
\ifabc@shellescape
  \let\abc@process\abc@doshellcommand
\else
  \newwrite\abc@outsh
  \immediate\openout\abc@outsh\abc@tempfile.sh
  \AtEndDocument{\closeout\abc@outsh}
  \let\abc@process\abc@nodoshellcommand
\fi
```

`\abc@start` `\abc@finish`

We now define how to start and finish; if no generation is required, the ABC lines are skipped like a comment (thanks again to the verbatim package).

```
\ifabc@generate
  \let\abc@start\abc@startgen
  \let\abc@finish\abc@finishgen
\else
  \let\abc@start\comment
  \let\abc@finish\endcomment
\fi
```

`abc` `\abcinput`

Finally, we define the environment and the command. Some commands are initialized here; the name of the temporary file is, by default, "out-abc" which should not clobber any existing file.

```
\def\abc{\@makeother\%\@ifnextchar[\abc@grab{\abc@grab[]}}
\define@key{abc}{name}[]{%
  \if!#1!\stepcounter{abc@count}%
    \edef\abc@tempfile{\normalabcoutputfile-\@arabic\c@abc@count}%
  \else
    \def\abc@tempfile{#1}%
```

```
  \fi
}
\define@key{abc}{options}{\def\abc@opt{#1}}
\define@key{abc}{postoptions}{\def\abc@postopt{#1}}
\define@key{abc}{program}{\def\abc@cmd{#1}\let\abc@parm\@empty}
\define@key{abc}{width}{\def\abc@width{#1}}
\define@key{abc}{center}[true]{\csname abc@center#1\endcsname}
\define@key{abc}{extension}{\def\abc@ext{.#1}}
\def\abc@grab[#1]{\let\abc@width=\abcwidth
  \ifabc@saveall
    \setkeys{abc}{name,#1}%
  \else
    \setkeys{abc}{#1}%
  \fi\abc@start}
```

The final part of the environment; we do the processing, if required and then include the graphics file. If none is found, the simple name is used, to recall that some processing is to be done.

```
\def\endabc{%
  \abc@finish
  \trivlist\item[]\ifabc@center\centering\fi
  \IfFileExists{\abc@tempfile.\abc@finalext}
    {\includegraphics[width=\abc@width]{\abc@tempfile.\abc@finalext}}%
    {\global\abc@warningtrue\fbox{\abc@tempfile}%
     \global\abc@unprocessedfilestrue}%
  \endtrivlist
}
```

The command version is similar. The only difference is that we issue a warning if the named file does not exist.

```
\def\abcinput{\@ifnextchar[\abc@grabinput{\abc@grabinput[]}}
\def\abc@grabinput[#1]#2{\let\abc@width=\abcwidth\setkeys{abc}{#1}%
  \begingroup\def\abc@tempfile{#2}%
  \IfFileExists{\abc@tempfile\abc@ext}
  {%
    \abc@process
    \begin{center}
    \IfFileExists{\abc@tempfile.\abc@finalext}
      {\includegraphics[width=\abc@width]{\abc@tempfile.\abc@finalext}}%
      {\fbox{\abc@tempfile}}%
    \end{center}%
  \endgroup
  }
  {\PackageWarning{\abc@packagename}{No file \abc@tempfile\abc@ext\space found}}%
}
```

⟨/package⟩

```
⟨*package-mup⟩
\ProvidesPackage{mup}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{abc}}
\ProcessOptions\relax
\RequirePackage[mup]{abc}
```

⟨/package-mup⟩