MusixCrd
— Typesetting Chord Symbols with MusiXTeX—
Version 1.0

Revision: 1.7

Robert Hennig*

November 2, 2004

Contents

1 Usage 1
1.1 Syntax ......................................................... 2
1.2 Semantics ...................................................... 3

2 Implementation 4
2.1 List Macros .................................................. 4
2.2 Parsing ......................................................... 4
2.3 Chord parsing ................................................ 6
2.3.1 Vertical and Horizontal shifting ....................... 6
2.3.2 Notes and Accidentals .................................. 6
2.3.3 Chord Qualifiers ........................................ 8
2.3.4 Parsing the whole chord ............................... 9
2.3.5 Multiple chords ......................................... 10
2.4 Formatting ................................................... 10

3 Customization 12
3.1 Changing the extensions ................................. 12
3.2 change fonts ............................................... 13

4 Todo 15

1 Usage

This package was written to ease the typesetting of chord symbols for music scores. One point of focus was that the user should have not to much to type if placing the chord. So one macro will be used which takes characters as argument which describe the chord to type.

*robert.hennig@freylax.de
Thought the syntax of the chord description could easily be altered they should become somewhat stable whereas the output format can be adapted to individual needs. Further the notenames can be transposed, so transposing a music piece with chord symbols can be done easily.

The package can be used with MusiXTeX and PMX – which also gave the idea for the usage of an short chord-description ‘language’.  

The main macro which the package defines is `\c\langle chord-list\rangle`. The argument is an space terminated `\langle chord-list\rangle`.

\begin{MusiXTeX}
\begin{tikzpicture}
\node at (0,0) {C\textsuperscript{7} D\textsubscript{in} C\textsuperscript{9}/E F\textsuperscript{6}/A E\textsuperscript{7}/G F\textsuperscript{0}/A\textsubscript{b}};
\end{tikzpicture}
\end{MusiXTeX}

\nobarnumbers
\startextract\NOtes\c CM7 \hu e\c Dm7 \hu f\en\bar
\NOTEs\c Ch/E \hu g\c F6/A \hu h\en\bar
\NOTEs\c 0-1E7/G \hl i\c Fd/Af \hl j\en
\endextract

1.1 Syntax

\texttt{\langle empty\rangle ::= ‘\'}

\texttt{\langle digit\rangle ::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’}

\texttt{\langle number\rangle ::= \langle digit\rangle | ‘-’ \langle digit\rangle}

\texttt{\langle vertical-shift\rangle ::= \langle number\rangle | \langle empty\rangle}

\texttt{\langle horizontal-shift\rangle ::= \langle number\rangle | \langle empty\rangle}

\texttt{\langle note-base-name\rangle ::= ‘C’ | ‘D’ | ‘E’ | ‘F’ | ‘G’ | ‘A’ | ‘B’}

\texttt{\langle accidental\rangle ::= ‘s’ | ‘f’ | ‘ds’ | ‘df’}

\texttt{\langle note-name\rangle ::= \langle empty\rangle | \langle note-base-name\rangle | \langle note-base-name\rangle \langle accidental\rangle}

\texttt{\langle chord-qualifier\rangle ::= ‘m’ | ‘d’ | ‘h’ | ‘M’ | ‘+5’ | ‘6’ | ‘7’ | ‘-9’ | ‘+9’}

\texttt{\langle chord-qualifier-list\rangle ::= \langle empty\rangle | \langle chord-qualifier\rangle \langle chord-qualifier-list\rangle}

\texttt{\langle begin-bass-note\rangle ::= \langle empty\rangle | ‘/’}

\footnote{1If the default output functions are changed the package could also be used with \TeX{} and \LaTeX{} alone.}

\footnote{2This form of argument was choosen because it leads to a short notation inside PMX e.g.:\c\texttt{AfM \ e8 f g4 \c Gm7 \ b4 g}}
\langle chord \rangle ::= (vertical-shift) (horizontal-shift) 
\langle note-name \rangle (chord-qualifier-list) 
\langle begin-bass-note \rangle \langle note-name \rangle (chord-qualifier-list)

\langle chord-list \rangle ::= (chord-list) ', ' (chord-list) | (chord) | (empty)

1.2 Semantics

\texttt{\\textbackslash crdefaultheight} \langle vertical-shift \rangle \quad \text{Adjustment of the vertical chord position in internotes, relative to the default value defined with} \texttt{\\textbackslash crdefaultheight}. \text{You may change this default within your sheet.}
1 \texttt{\textbackslash def \textbackslash crdefaultheight(10)}

\langle horizontal-shift \rangle \quad \text{Horizontal adjustment in multiples of} \texttt{\\textbackslash elemskip}.

\langle accidental \rangle \quad \text{Allowed accidentals are: sharp, flat, double-sharp, double-flat.}

\texttt{\\textbackslash crdtranspose} \langle note-name \rangle \quad \text{The given note names are transposed by the number of quint steps given in} \texttt{\\textbackslash crdtranspose}. \text{You may change this value within your sheet.}
2 \texttt{\textbackslash def \textbackslash crdtranspose(0)}

\langle chord-qualifier \rangle \quad \text{Currently known qualifiers are:}

\texttt{m} \quad \text{minor}
\texttt{d} \quad \text{diminished}
\texttt{h} \quad \text{half-diminished}
\texttt{M} \quad \text{major}
\texttt{+5} \quad \text{augmented fifth}
6 \quad \text{6th}
7 \quad \text{7th}
9 \quad \text{9th}
-9 \quad \text{diminished 9th}
+9 \quad \text{augmented 9th}

\text{Note that the syntax is independent of the visualization of the qualifier so different chord styles could be applied.}

\langle begin-bass-note \rangle \quad \text{Use the '}/' symbol to skip the} \langle note-name \rangle \text{and} \langle chord-qualifier-list \rangle \text{to allow the notation of bass-notes without chord-notes}.

\langle chord-list \rangle \quad \text{With '},' separated chords are spread evenly within one bar. \text{Use this notation if the horizontal positions of the chords do not line up with the notes.}
\section{Implementation}

\subsection{List Macros}

For the parsing of the chord description some macros are needed which can do simple string operations.

\begin{verbatim}
\crd@append \crd@append⟨tokens-a⟩\to⟨tokens-b⟩
Append ⟨tokens-a⟩ to ⟨tokens-b⟩.
\end{verbatim}

\begin{verbatim}
\crd@prepend \crd@prepend⟨tokens-a⟩\by⟨tokens-b⟩
Prepend ⟨tokens-a⟩ by ⟨tokens-b⟩.
\end{verbatim}

\begin{verbatim}
\crd@movetoken \crd@movetoken⟨tokens-a⟩\to⟨tokens-b⟩
Move the first token of ⟨tokens-a⟩ to the front of ⟨tokens-b⟩.
\end{verbatim}

\begin{verbatim}
\makeatletter
\def\l{oo} (\l) % \crd@append\ns\to\l append:(\l) % \crd@prepend\l\by\sp prepend:(\l)\%% \swap\l\#2(\l) % \ifx\l\empty\else\expandafter\crd@moveoff\l\#1\swap\l\fi\%
\makeatother
\end{verbatim}

\begin{verbatim}
test
\crd@parse\l
\crd@parsematched\l\for\l
\end{verbatim}

\subsection{Parsing}

To describe the syntactic items which exists for a distinct semantic a ⟨syntax-table⟩ is used. For each item exists a corresponding macro which will be executed if its name matches. The name of the item consists of the ⟨syntax-table⟩ name and the reverse syntax of this item.

\begin{verbatim}
\crd@parse The \crd@parse ⟨tokens⟩ \for ⟨syntax-table⟩ macro is used to test if the first part of ⟨tokens⟩ has matches for the longest possible item described in ⟨syntax-table⟩. If an item matched its macro will be expanded and and the tokens of the item are cut of from the given ⟨tokens⟩. The conditional \crd@parsematched is true if an item matched and false otherwise.
\end{verbatim}
\newif\ifcrd@parsematched% true if parse matched
\newcount\crd@parsedepth% internal register
\def\crd@parse#1\for#2{% parse tokens #1 for occurrence of items of table #2
\crd@parsedepth=1 % default if not defined
\expandafter\ifx\csname#2depth\endcsname\relax\else%
\crd@parsedepth=\csname#2depth\endcsname%
\fi%
\def\stack{}%\def\crd@parseresult{}
\crd@parsematchedfalse% initialisation
\crd@parser#1\for#2% call the recursive part
\}
\def\crd@parser#1\for#2{% recursive part of parser
\ifx#1\empty\else% is list filled ?
\ifnum\crd@parsedepth>0 % and do we have to read more chars into stack
\advance\crd@parsedepth by-1 %
\crd@movetoken#1\to\stack%
\crd@parser#1\for#2% recursive call
\ifcrd@parsematched\else% if still not matched
\expandafter\ifx\csname#2\stack\endcsname\relax% does item match
\csname#2\stack\endcsname%
\crd@parsematchedtrue% signal success
\fi%
\fi%
\fi%
\fi%
\fi%
\fi%
\fi%
\}

For an example suppose that we want to express the semantic \langle bool \rangle by the following grammar:

\langle bool \rangle ::= \texttt{‘y’} | \texttt{‘n’} | \texttt{‘yes’} | \texttt{‘no’}

\begin{verbatim}
    bool
    (nonyyestest) false*: (nyyestest)
    (nyyestest) false: (yyestest)
    (yyestest) true: (yestest)
    (yestest) true*: (test)
    (test) : (test)
\end{verbatim}

\def\bool{bool} % syntax-table with name 'bool'
\def\booldepth{3} % max length of text to looking for is 3
\def\booly{true}
\def\booln{false}
\def\boolsey{true*} % reverse syntax !!!
\def\boolon{false*} % reverse syntax !!!
\makeatletter
\def\p#1{ (#1) \crd@parse#1\for\bool : (#1)\crd\@parseresult{}}
\makeatother
\def\l{nonyyestest} \p\l \p\l \p\l \p\l \p\l
2.3 Chord parsing

2.3.1 Vertical and Horizontal shifting

\newcount\crd@vshift%
\newcount\crd@hshift%
\def\crd@number{crd@number}%
\def\crd@numberdepth{2}%
\expandafter\def\csname\crd@number0\endcsname{\crd@numberval=0 }%
\expandafter\def\csname\crd@number1\endcsname{\crd@numberval=1 }%
\expandafter\def\csname\crd@number2\endcsname{\crd@numberval=2 }%
\expandafter\def\csname\crd@number3\endcsname{\crd@numberval=3 }%
\expandafter\def\csname\crd@number4\endcsname{\crd@numberval=4 }%
\expandafter\def\csname\crd@number5\endcsname{\crd@numberval=5 }%
\expandafter\def\csname\crd@number6\endcsname{\crd@numberval=6 }%
\expandafter\def\csname\crd@number7\endcsname{\crd@numberval=7 }%
\expandafter\def\csname\crd@number8\endcsname{\crd@numberval=8 }%
\expandafter\def\csname\crd@number9\endcsname{\crd@numberval=9 }%
\expandafter\def\csname\crd@number1-\endcsname{\crd@numberval=-1 }%
\expandafter\def\csname\crd@number2-\endcsname{\crd@numberval=-2 }%
\expandafter\def\csname\crd@number3-\endcsname{\crd@numberval=-3 }%
\expandafter\def\csname\crd@number4-\endcsname{\crd@numberval=-4 }%
\expandafter\def\csname\crd@number5-\endcsname{\crd@numberval=-5 }%
\expandafter\def\csname\crd@number6-\endcsname{\crd@numberval=-6 }%
\expandafter\def\csname\crd@number7-\endcsname{\crd@numberval=-7 }%
\expandafter\def\csname\crd@number8-\endcsname{\crd@numberval=-8 }%
\expandafter\def\csname\crd@number9-\endcsname{\crd@numberval=-9 }%

2.3.2 Notes and Accidentals

Syntax In order to allow transposition of notes we use the circle of fifths for representing notes. The syntax table \crd@quintval contains the mapping from \crd@quintval note names to the note position in the circle of fifths.

\crd@quintval
\crd@quint
\def\crd@quintval{crd@quintval}
\def\crd@quintvalA{\crd@quint=3 }% A
\def\crd@quintvalB{\crd@quint=5 }% B
\def\crd@quintvalC{\crd@quint=0 }% C
\def\crd@quintvalD{\crd@quint=2 }% D
\def\crd@quintvalE{\crd@quint=4 }% E
\def\crd@quintvalF{\crd@quint=1 }% F
\def\crd@quintvalG{\crd@quint=1 }% G

\crd@quintmod
The modification of the note position in the circle of fifths which is caused by the accidentals is coded in the \crd@quintmod syntax table.

\def\crd@quintmod{crd@quintmod}
\def\crd@quintmoddepth{2}
\def\crd@quintmoda{\advance\crd@quint by7 }% sharp
\def\crd@quintmodf{\advance\crd@quint by-7 }% flat
\def\crd@quintmodb{\advance\crd@quint by14 }% double sharp
\def\crd@quintmodfb{\advance\crd@quint by-14 }% double flat

Notenames and accidental symbols After transposition and enharmonic adaption an reverse mapping from the circle of fifths to notenames and accidentals is
needed. The mapping from circle of fifth to notenames without accidentals is specified in the \texttt{\textbackslash crd@note} table. In addition we need to now where the notes without accidentals start and end which is defined in \texttt{\textbackslash crd@notelow} and \texttt{\textbackslash crd@notehigh}.

\begin{verbatim}
79 \def\crd@note{crd@note}
80 \expandafter\def\csname\crd@note3\endcsname{A}
81 \expandafter\def\csname\crd@note5\endcsname{B}
82 \expandafter\def\csname\crd@note0\endcsname{C}
83 \expandafter\def\csname\crd@note2\endcsname{D}
84 \expandafter\def\csname\crd@note4\endcsname{E}
85 \def\crd@notelow{-1} % lowest quint without accidental
86 \def\crd@notehigh{5} % highest quint without accidental
87 \def\crd@note\{\sharp\}
88 \def\crd@flat\{\flat\}
89 \def\crd@doublesharp\{\sharp\sharp\}
90 \def\crd@doubleflat\{\flat\flat\}
\end{verbatim}

The following macros define the representation of the (default) accidentals.

\begin{verbatim}
\def\crd@parsenote#1#2{% parse input results: #1
\def#1\{}
\def#2\{}
\crd@parse\crd@input\for\crd@quintval\ifcrd@parsematched% we got an valid note
\crd@parse\crd@input\for\crd@quintmod\advance\crd@quint by\crdtranspose \relax% transposition, space is needed!
\crd@enharmonic\ifnum\crd@quint>\crd@notehigh % sharps ?
\advance\crd@quint by-7 %
\ifnum\crd@quint>\crd@notehigh % double sharp ?
\advance\crd@quint by-7 %
\ifnum\crd@quint>\crd@notehigh % too much sharps !
\relax ERROR:too much sharps%
\else#2\{\crd@doublesharp\}fi%
\else#2\{\crd@sharp\}fi%
\else#2\{\crd@doubleflat\}fi%
\else#2\{\crd@flat\}fi%
\fi%
\ifnum\crd@quint<\crd@notelow % flats ?
\advance\crd@quint by7 %
\ifnum\crd@quint<\crd@notelow % double flat ?
\advance\crd@quint by7 %
\ifnum\crd@quint<\crd@notelow % too much flats !
\relax ERROR:too much flats%
\else#2\{\crd@doubleflat\}fi%
\else#2\{\crd@flat\}fi%
\fi%
\expandafter\ifx\csname\crd@note\number\crd@quint\endcsname\relax% ERROR:note name for (\number\crd@quint) is not defined.
\end{verbatim}
To allow different enharmonic adaptations the `\crd@enharmonic` macro is provided which default behaviour is to do nothing.

\begin{example}
\makeatletter
\def\crd@enharmonic{}%
\makeatother
\parsenotes{CDfDEfEFFsGAfAsB}
\end{example}

2.3.3 Chord Qualifiers

To cover a broad range of different styles for setting chord qualifiers the design is open for extensions. For the sake of demonstration and simple usability an default implementation is provided and discussed furtherwards.

Suppose we want to distinguish 3 different kinds of qualifiers, some go down, some go up and alterations are put in brackets. We choose to use 3 lists (macros) to hold the parsing results. For initialisation of these lists the `\crd@qualinit` macro has to be implemented.

\begin{example}
\makeatletter
\def\crd@qualinit{%  
  \def\crd@lo{}% lower extensions  
  \def\crd@up{}% upper extensions  
  \def\crd@alt{}% alterations  
}%
\makeatother
\parsenotes{CDfDEfEFFsGAfAsB}
\end{example}

Now the syntax table `\crd@qual` has to be defined which fills the lists appropriatly.
2.3.4 Parsing the whole chord

We are now ready to parse the whole chord, consisting of chordnote, qualifiers and bassnote. However if one likes to set only a bassnote one needs to tell that there is no chord note to set. For this purpose the \crd@skipcrdnote syntax table defines the '/' item which does this skip.

\crd@skipcrdnote
\parsequal{mMdh+567-9+9}

\crd@parsecd
2.3.5 Multiple chords

The \texttt{\textbackslash crd@parsecrds} macro is used to read more than one chord. This can be useful if no corresponding note over which one can put the note exist. The syntax \texttt{\textbackslash crd@crddelim} is used.

\begin{verbatim}
def\crd@crddelim{\crd@crddelim}\
\expandafter\def\csname\crd@crddelim,\endcsname{}\
def\crd@parsecrds{\
crd@parsecrd\
crd@parse\crd@input\for\crd@crddelim\
if\crd@parsematched\crd@parsecrds\fi\}
\end{verbatim}

\begin{center}
\begin{tabular}{ccc}
$C_m$ & $C_M$ & $C_m^7$
\end{tabular}
\end{center}

The main entry point for the user is the \texttt{\textbackslash c\{chord-list\}} macro which calls the \texttt{\textbackslash crd@output} routine with the formatted chords.

\begin{verbatim}
def\c#1 {\def\crd@input{#1}\crd@output\crd@parsecrds}
\end{verbatim}

2.4 Formatting

To allow the use of different fonts the notion of fontsstyles is introduced. The initialisation of fontsstyles is done in different macros.

\begin{verbatim}
def\crd@fontstylea{
\font\crd@eightrm=cmr8
\font\crd@eightit=cmmi8
\font\crd@seventeenrm=cmr17
\font\crd@fourteenrm=cmr14
\font\crd@twelverm=cmr12
\font\crd@ninerm=cmr9
\let\crd@notetype=\crd@seventeenrm
\def\crd@noteflat\{aise0.6ex\hbox{\kern-0.085em\musictwenty2}}
\def\crd@notedoubleflat\{aise0.6ex\hbox{\kern-0.085em\musictwenty3}}
\def\crd@notesharp\{aise0.8ex\hbox{\musictwenty4}}
\def\crd@notedoublesharp\{aise0.8ex\hbox{\musictwenty5}}
\let\crd@basstype=\crd@fourteenrm
\end{verbatim}
The formatting of the chords is done in the \crd@formatcrd macro. The parse results are stored in the following macros: \crd@crdnote – chord note, \crd@crdacc – chord note accidental, \crd@bassnote – bass note and \crd@bassacc – bass note accidental.

\begin{verbatim}
def\crd@formatcrd{% 
  \ifx\crd@alt\empty\else\crd@numbertype(\crd@alt\crd@numbertype)\fi% 
  \crd@up% 
  \crd@dim\crd@notetype\crd@crdnote\crd@crdacc\vbox{% 
    \crd@lo}% 
  \ifx\crd@bassnote\empty\else% 
    \crd@bassnote/\crd@bassacc\vphantom{\crd@capitaltype M}\crd@bassnote\crd@basstype/\crd@bassnote\crd@bassacc\fi% 
  \lower0.5ex\hbox{\kern-0.17em \crd@bassnote\crd@bassacc}}% 
}\crd@formatcrd
\let\crd@formatcrd=\crd@formatcrd% 
\crd@fontstylea
\end{verbatim}
3 Customization

If the default implementation does not suite the needs some tips of how to change the default behaviour will be given.

3.1 Changing the extensions

Q: How do I get the symbol $Dm^{M7}$?

A: The M has to be put to the upper extensions list, so we have to change the definition in the \crd@qual table:

\makeatletter
\def\crd@qualM{\crd@append{\crd@capitaltype M}\to\crd@up}% major7
\c DmM7 %
\makeatother%

Q: How do I get the symbol $Dm^{7-5}$?

A: - The predefined symbol can be found by using ‘h’ – for half diminished.

\c Dh %

- to get the diminished fifth explicit to the upper extensions one could either change the definition of the ‘h’ to:
Dmh : $Dm^{7-5}$

\[ D_{m}^{7-5} \]

- or may introduce an mapping for the -5 and has to write: $D^{7-5}$

\[ D_{m}^{7-5} \]

3.2 change fonts

Q: How do I change the font?

A: One has to create his own fontstyle definition with a suitable formatting like for example:
\begin{verbatim}
\makeatletter
\def\crd@fontstyleb{\%
  \font\crd@newfont=cmssbx10\%
  \def\crd@notetype={\crd@newfont}\%
  \def\crd@noteflat={\raise2pt\hbox{\musixchar90}}\%
  \def\crd@notedoubleflat={\crd@noteflat\crd@noteflat}\%
  \def\crd@notesharp={\raise3.5pt\hbox{\musixchar92}}\%
  \def\crd@notedoublesharp={\crd@notesharp\crd@notesharp}\%
  \let\crd@basstype=\crd@newfont\%
  \def\crd@bassflat={\crd@noteflat}\%
  \def\crd@bassdoubleflat={\crd@notedoubleflat}\%
  \def\crd@basssharp={\crd@notesharp}\%
  \def\crd@bassdoublesharp={\crd@notedoublesharp}\%
  \let\crd@numbertype=\crd@newfont\%
  \def\crd@numberflat={\crd@noteflat}\%
  \def\crd@numbersharp={\crd@notesharp}\%
  \def\crd@numberminus={\crd@newfont-}\%
  \def\crd@numberplus={\crd@newfont+}\%
  \let\crd@capitaltype=\crd@newfont % capitals in extension
  \let\crd@smalltype=\crd@newfont % small letters in extension
  \def\crd@hdim{\crd@newfont$\circ$\kern-4.4pt\raise.9pt\hbox{\crd@newfont/}}\%
  \def\crd@dim{\crd@newfont$\circ$}\%
}\%
\def\crd@formatcrdb{\%
  \hbox{\kern\crd@hshift\elemskip\raise\crd@vshift\internote\hbox{\%
    \crd@notetype\crd@crdnote\crd@crdacc\%
    \crd@lo\%
    \raise4pt\%
    \hbox{\%
      \crd@up\%
      \ifx\crd@alt\empty\else\crd@numbertype{\crd@alt\crd@numbertype}\fi\%
    }\%
    \ifx\crd@bassnote\empty\else\%
      \crd@basstype{\crd@bassnote}\crd@bassacc\%
    \fi\%
    \}
}\%
\crd@fontstyleb\%
\let\crd@formatcrd=\crd@formatcrdb
\makeatother%
\nobarnumbers%
\startextract\NOtes\c D7, AfM7 \hu f\en\bar\endextract%
\end{verbatim}
Contributions for improving either the current font style or the definition of new ones are welcome.

4 Todo

The actual implementation is not really open for changing the input format and the way the chords are displayed. One should implement an middle layer which is fixed and offer various implementations either for the input format and the output format which are written using this layer.