MusixCrd

— Typesetting Chord Symbols with MusiXTEX— Version 1.0

Revision : 1.7

Robert Hennig*

November 2, 2004

Contents

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 cord. So one macro will be used which takes characters as argument which describe the chord to type.

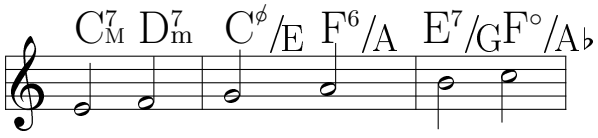
*robert.hennig@freylax.de

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'. [1]

\c The main macro which the package defines is `\c␣`⟨*chord-list*⟩`␣`. The argument is an space terminated ⟨*chord-list*⟩ [2].

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

⟨*empty*⟩ ::= ''

⟨*digit*⟩ ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

⟨*number*⟩ ::= ⟨*digit*⟩ | '-' ⟨*digit*⟩

⟨*vertical-shift*⟩ ::= ⟨*number*⟩ | ⟨*empty*⟩

⟨*horizontal-shift*⟩ ::= ⟨*number*⟩ | ⟨*empty*⟩

⟨*note-base-name*⟩ ::= 'C' | 'D' | 'E' | 'F' | 'G' | 'A' | 'B'

⟨*accidental*⟩ ::= 's' | 'f' | 'ds' | 'df'

⟨*note-name*⟩ ::= ⟨*empty*⟩ | ⟨*note-base-name*⟩ | ⟨*note-base-name*⟩ ⟨*accidental*⟩

⟨*chord-qualifier*⟩ ::= 'm' | 'd' | 'h' | 'M' | '+5' | '6' | '7' | '-9' | '+9'

⟨*chord-qualifier-list*⟩ ::= ⟨*empty*⟩ | ⟨*chord-qualifier*⟩ ⟨*chord-qualifier-list*⟩

⟨*begin-bass-note*⟩ ::= ⟨*empty*⟩ | '/'

[1] If the default output functions are changed the package could also be used with TEX and LATEX alone.

[2] This form of argument was choosen because it leads to a short notation inside PMX e.g.: `\c AfM \ e8 f g4 \c Gm7 \ b4 g`

⟨*chord*⟩ ::= ⟨*vertical-shift*⟩ ⟨*horizontal-shift*⟩
⟨*note-name*⟩ ⟨*chord-qualifier-list*⟩
⟨*begin-bass-note*⟩ ⟨*note-name*⟩ ⟨*chord-qualifier-list*⟩

⟨*chord-list*⟩ ::= ⟨*chord-list*⟩ '`,`' ⟨*chord-list*⟩ | ⟨*chord*⟩ | ⟨*empty*⟩

1.2 Semantics

`\crddefaultheight` ⟨*vertical-shift*⟩ Adjustment of the vertical chord position in internotes, relative to the default value defined with `\crddefaultheight`. You may change this default within your sheet.

```
1 \def\crddefaultheight{10}
```

⟨*horizontal-shift*⟩ Horizontal adjustment in multiples of `\elemskip`.

⟨*accidental*⟩ Allowed accidentals are: sharp, flat, double-sharp, double-flat.

`\crdtranspose` ⟨*note-name*⟩ The given note names are transposed by the number of quint steps given in `\crdtranspose`. You may change this value within your sheet.

```
2 \def\crdtranspose{0}
```

⟨*chord-qualifier*⟩ Currently known qualifiers are:

m minor

d diminished

h half-diminished

M major

+5 augmented fifth

6 6th

7 7th

9 9th

-9 diminished 9th

+9 augmented 9th

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

⟨*begin-bass-note*⟩ Use the '/' symbol to skip the ⟨*note-name*⟩ and ⟨*chord-qualifier-list*⟩ to allow the notation of bass-notes without chord-notes.

⟨*chord-list*⟩ With ',' separated chords are spread evenly within one bar. Use this notation if the horizontal positions of the chords do not line up with the notes.

2 Implementation

2.1 List Macros

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

`\crd@append` `\crd@append`⟨*tokens-a*⟩`\to`⟨*tokens-b*⟩
Append ⟨*tokens-a*⟩ to ⟨*tokens-b*⟩.

```
\def\crd@append#1\to#2{%
  \toks0=\expandafter{#1}\toks2=\expandafter{#2}%
  \edef#2{\the\toks2 \the\toks0}}
```

`\crd@prepend` `\crd@prepend`⟨*tokens-a*⟩`\by`⟨*tokens-b*⟩
Prepend ⟨*tokens-a*⟩ by ⟨*tokens-b*⟩.

```
\def\crd@prepend#1\by#2{%
  \toks0=\expandafter{#1}\toks2=\expandafter{#2}%
  \edef#1{\the\toks2 \the\toks0}}
```

`\crd@movetoken` `\crd@movetoken`⟨*tokens-a*⟩`\to`⟨*tokens-b*⟩
Move the first token of ⟨*tokens-a*⟩ to the front of ⟨*tokens-b*⟩.

```
\def\crd@movetoken#1\to#2{%
  \ifx#1\empty\else\expandafter\crd@moveoff#1\crd@moveoff#1#2\fi}%
\def\crd@moveoff#1#2\crd@moveoff#3#4{\def#3{#2}\crd@prepend#4\by#1}
```

test

(oo) append:(oons) prepend:(spoons)
movetoken:
(spoons,) (poons,s) (oons,ps) (ons,ops) (ns,oops) (s,noops) (,snoops)

```
\makeatletter
\def\l{oo} (\l) %
\crd@append{ns}\to\l append:(\l) %
\crd@prepend\l\by{sp} prepend:(\l)\\%
\def\swap#1#2{(#1,#2) %
  \ifx#1\empty\else\crd@movetoken#1\to#2\swap#1#2\fi}
\def\r{} movetoken:\\ \swap\l\r
\makeatother
```

2.2 Parsing

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

`\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
`\crd@parsematched` cut of from the given ⟨*tokens*⟩. The conditional `\crd@parsematched` is true if an item matched and false otherwise.

```
\newif\ifcrd@parsematched%  true if parse matched
\newcount\crd@parsedepth%   internal register

\def\crd@parse#1\for#2{% parse tokens #1 for occurence 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
          \crd@movetoken\stack\to#1% no match, put char back to source
        \else% match
          \csname#2\stack\endcsname%
          \crd@parsematchedtrue% signal success
        \fi%
      \fi%
    \fi%
  \fi%
}
```

For an example suppose that we want to express the semantic ⟨*bool*⟩ by the following grammar:

⟨*bool*⟩ ::= ‘`y`’ | ‘`n`’ | ‘`yes`’ | ‘`no`’

bool

(nonyyestest) false*: (nyyestest)
(nyyestest) false: (yyestest)
(yyestest) true: (yestest)
(yestest) true*: (test)
(test) : (test)

```
\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)\\}
\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 fifth for representing notes. The syntax table `\crd@quintval` contains the mapping from note names to the note position in the circle of fifth.

`\crd@quintval`

`\crd@quint` The count register `\crd@quint` is used to receive the result.

```
\newcount\crd@quint% register used to represent notes in the circle of
\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 fifth 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@quintmods{\advance\crd@quint by7 }%    sharp
\def\crd@quintmodf{\advance\crd@quint by-7 }%   flat
\def\crd@quintmodsd{\advance\crd@quint by14 }%  double sharp
\def\crd@quintmodfd{\advance\crd@quint by-14 }% double flat
```

Notenames and accidental symbols After transpositon and enharmonic adaption an reverse mapping from the circle of fifth to notenames and accidentals is

\crd@note
\crd@notelow
\crd@notehigh

needed. The mapping from circle of fifth to notenames without accidentals is specified in the \crd@note table. In addition we need to now where the notes without accidentals start and end which is defined in \crd@notelow and crd@notehigh.

```
\def\crd@note{crd@note}
\expandafter\def\csname\crd@note3\endcsname{A}
\expandafter\def\csname\crd@note5\endcsname{B}
\expandafter\def\csname\crd@note0\endcsname{C}
\expandafter\def\csname\crd@note2\endcsname{D}
\expandafter\def\csname\crd@note4\endcsname{E}
\expandafter\def\csname\crd@note-1\endcsname{F}
\expandafter\def\csname\crd@note1\endcsname{G}
\def\crd@notelow{-1} % lowest quint without accidental
\def\crd@notehigh{5} % highest quint without accidental
```

\crd@sharp
\crd@flat
\crd@doublesharp
\crd@doubleflat

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

```
\def\crd@sharp{\sharp}
\def\crd@flat{\flat}
\def\crd@doublesharp{\sharp\sharp}
\def\crd@doubleflat{\flat\flat}
```

\crd@input
\crd@parsenote

Parsing The following functions read their input from \crd@input.
The \crd@parsenote⟨*note*⟩⟨*accidental*⟩ macro parses the input \crd@input and if note and accidental could be detected the transposed note position in the circle of fifth will be calculated, transposed and adapted using the \crd@enharmonic macro. Then the position in the circle of fifth is calculated back to the ⟨*note*⟩ and ⟨*accidental*⟩ representation which is stored in the arguments.

```
\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\edef#2{\crd@doublesharp}\fi%
      \else\edef#2{\crd@sharp}\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\edef#2{\crd@doubleflat}\fi%
      \else\edef#2{\crd@flat}\fi%
    \fi%
    \expandafter\ifx\csname\crd@note\number\crd@quint\endcsname\relax%
      ERROR:notename for (\number\crd@quint) is not defined.
```

```
    \fi%
    \edef#1{\csname\crd@note\number\crd@quint\endcsname}% set note
  \fi%
}
```

\crd@enharmonic To allow different enharmonic adaptions the \crd@enharmonic macro is provided which default behaviour is to do nothing.

```
\def\crd@enharmonic{}%
```

noteparsing

CD♭DE♭EFF♯GA♭A♯B

```
\makeatletter
\def\parsenoter{\crd@parsenote\n\a%
\ifx\n\empty\else{\crd@notetype\n\a}\parsenoter\fi}%
\def\parsenotes#1{%
   \let\crd@flat=\crd@noteflat%
   \let\crd@sharp=\crd@notesharp%
   \def\crd@input{#1}\parsenoter}
\makeatother

\parsenotes{CDfDEfEFFsGAfAsB}
```

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.

\crd@qualinit 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.

```
\def\crd@qualinit{%
  \def\crd@lo{}%    lower extensions
  \def\crd@up{}%    upper extensioins
  \def\crd@alt{}%   alterations
}
```

\crd@qual Now the syntax table \crd@qual has to be defined which fills the lists appropriatly.

```
\def\crd@qual{crd@qual}%
\def\crd@qualdepth{2}
\def\crd@qualm{\crd@append{\crd@smalltype m}\to\crd@lo}% minor
\def\crd@qualM{\crd@append{\crd@capitaltype M}\to\crd@lo}% major7
\expandafter\def\csname\crd@qual5+\endcsname% aug. 5
{\crd@append{+}\to\crd@lo}
\expandafter\def\csname\crd@qual6\endcsname% 6
{\crd@append{\crd@numbertype6}\to\crd@up}
\expandafter\def\csname\crd@qual7\endcsname% dominant 7
{\crd@append{\crd@numbertype7}\to\crd@up}
\def\crd@quald{\crd@append{\crd@dim}\to\crd@up}% diminished
\def\crd@qualh{\crd@append{\crd@hdim}\to\crd@up}% half diminished
```

```
\expandafter\def\csname\crd@qual9-\endcsname% -9
{\crd@append{\crd@numbertype\crd@numberflat9}\to\crd@alt}
\expandafter\def\csname\crd@qual9+\endcsname% +9
{\crd@append{\crd@numbertype\crd@numbersharp9}\to\crd@alt}
```

qualparsing

$(\mathrm{mM}+,\circ\not{6}7,\flat9\sharp9)$

```
\makeatletter
\def\parsequal#1{\def\crd@input{#1}%
   \crd@qualinit%
   \loop\crd@parse\crd@input\for\crd@qual%
     \ifcrd@parsematched\repeat%
     (\crd@lo,\crd@up,\crd@alt)%
 }%
\makeatother

\parsequal{mMdh+567-9+9}
```

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`

```
\def\crd@skipcrdnote{crd@skipcrdnote}
\expandafter\def\csname\crd@skipcrdnote/\endcsname{}
```

`\crd@parsecrd`

```
\def\crd@parsecrd{%
  \crd@vshift=0 %
  \let\crd@numberval=\crd@vshift%
  \crd@parse\crd@input\for\crd@number%
  \crd@hshift=0 %
  \let\crd@numberval=\crd@hshift%
  \crd@parse\crd@input\for\crd@number%
  \def\crd@crdnote{}%  chord note
  \def\crd@crdacc{}%   chord note accidental
  \def\crd@bassnote{}% bass note
  \def\crd@bassacc{}%  bass note accidental
  \crd@qualinit%       initialize qualifiers
  \let\crd@flat=\crd@noteflat%
  \let\crd@doubleflat=\crd@notedoubleflat%
  \let\crd@sharp=\crd@notesharp%
  \let\crd@doublesharp=\crd@notedoublesharp%
  \crd@parsenote\crd@crdnote\crd@crdacc%    read chord note
  \loop\crd@parse\crd@input\for\crd@qual%   read qualifiers
     \ifcrd@parsematched\repeat%
  \crd@parse\crd@input\for\crd@skipcrdnote% skip eventually
  \let\crd@flat=\crd@bassflat%
  \let\crd@doubleflat=\crd@bassdoubleflat%
```

```
  \let\crd@sharp=\crd@basssharp%
  \let\crd@doublesharp=\crd@bassdoublesharp%
  \crd@parsenote\crd@bassnote\crd@bassacc%  read bass note
  \crd@formatcrd\hfil% call rendering
}
```

2.3.5 Multiple chords

\crd@parsecrds
\crd@crddelim

The \crd@parsecrds macro is used to read more then one chord. This can be usefull if no corresponding note over which one can put the note exist. The syntax table \crd@crddelim is used.

```
\def\crd@crddelim{crd@crddelim}%
\expandafter\def\csname\crd@crddelim,\endcsname{}%
\def\crd@parsecrds{%
  \crd@parsecrd%
  \crd@parse\crd@input\for\crd@crddelim%
  \ifcrd@parsematched\crd@parsecrds\fi%
}
```

multiple chords

$\mathrm{C_m}\ \mathrm{C_M}\ \mathrm{C^7_m}$

```
\c Cm,2CM,Cm7
```

\c

The main entry point for the user is the \c␣⟨*chord-list*⟩␣ macro which calls the \crd@output routine with the formatted chords.

```
\def\c#1 {\def\crd@input{#1}\crd@output\crd@parsecrds}
```

2.4 Formatting

\crd@fontstylea

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

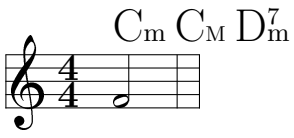
```
\def\crd@fontstylea{%
  \font\crd@eightrm=cmr8
  \font\crd@eightit=cmmi8
  \font\crd@seventeenrm=cmr17
  \font\crd@fourteenrm=cmr14
  \font\crd@twelverm=cmr12
  \font\crd@ninerm=cmr9
  \font\crd@smallninerm=cmr9 scaled 900
  \font\crd@bigninerm=cmr9 scaled 1100
  \let\crd@notetype=\crd@seventeenrm
  \def\crd@noteflat{\raise0.6ex\hbox{\kern-0.085em\musictwenty2}}
  \def\crd@notedoubleflat{\raise0.6ex\hbox{\kern-0.085em\musictwenty3}}
  \def\crd@notesharp{\raise0.8ex\hbox{\musictwenty4}}
  \def\crd@notedoublesharp{\raise0.8ex\hbox{\musictwenty5}}
  \let\crd@basstype=\crd@fourteenrm
```

```
  \def\crd@bassflat{\raise.5ex\hbox{\musicsixteen2}}
  \def\crd@bassdoubleflat{\raise0.6ex\hbox{\kern-0.085em\musicsixteen3}}
  \def\crd@basssharp{\raise1ex\hbox{\musicsixteen4}}
  \def\crd@bassdoublesharp{\raise0.8ex\hbox{\musicsixteen5}}
  \let\crd@numbertype=\crd@ninerm
  \def\crd@numberflat{\raise.5ex\hbox{\musiceleven2}}
  \def\crd@numbersharp{\raise1ex\hbox{\musiceleven4}}
  \def\crd@numberminus{\crd@ninerm-}
  \def\crd@numberplus{\crd@ninerm+}
  \let\crd@capitaltype=\crd@smallninerm % capitals
  \let\crd@smalltype=\crd@bigninerm   % small
  \def\crd@hdim{\crd@eightit$\circ$\kern-4.4pt\raise.9pt\hbox{\crd@eightrm/}}
  \def\crd@dim{\crd@eightit$\circ$}
}
```

`\crd@formatcrd` The formating 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.

```
\def\crd@formatcrda{%
  \hbox{\kern\crd@hshift\elemskip\raise\crd@vshift\internote\hbox{%
    {\crd@notetype\crd@crdnote\crd@crdacc}%
    \vbox{%
      \hbox{%
        \crd@up%
        \ifx\crd@alt\empty\else\crd@numbertype(\crd@alt\crd@numbertype)\fi%
      }%
      \nointerlineskip\vskip1pt%
      \hbox{\vphantom{\crd@capitaltype M}\crd@lo}}%
    \ifx\crd@bassnote\empty\else%
      {\crd@basstype/%
        \lower0.5ex\hbox{\kern-0.17em \crd@bassnote\crd@bassacc}}%
    \fi%\
  }}
}
\let\crd@formatcrd=\crd@formatcrda
\crd@fontstylea
```

formatting




```
\generalsignature{0}\generalmeter{\meterfrac44}
\nobarnumbers
\startextract
\NOtes\c Cm,CM,Dm7 \hu f\en\bar
\endextract
```

`\crd@output` The output function puts the formatted chords on their place.

```
\def\crd@musixOutput#1{\hbox{\zchar{\crddefaultheight}{\hbox to\elemskip{#1\hss}}}}
\let\crd@output=\crd@musixOutput
```

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:

Dm^{M7}

$\mathrm{D}_{\mathrm{m}}^{\mathrm{M7}}$

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

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

A: - The predefined symbol can be found by using 'h' – for half dimineshed.

h

D^{ϕ}

```
\c Dh %
```

- to ge the diminished fifth explicit to the upper extensions one could either change the definition of the 'h' to:

Dmh : Dm^{7-5}

$\mathrm{D}_{\mathrm{m}}^{7\text{-}5}$

```
\makeatletter%
\def\crd@qualh%
{\crd@append{\crd@numbertype 7-5}\to\crd@up}% half diminished
\makeatother%
\c Dmh %
```

- or may introduce an mapping for the -5 and has to write: D7-5

Dm7-5 : Dm^{7-5}

$\mathrm{D}_{\mathrm{m}}^{7\text{-}5}$

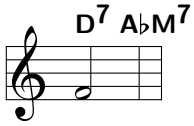
```
\makeatletter%
\expandafter\def\csname\crd@qual5-\endcsname% dimin. 5
{\crd@append{\crd@numberminus\crd@numbertype5}\to\crd@up}%
\makeatother%
\c Dm7-5 %
```

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:

change fonts




```
\makeatletter%
\def\crd@fontstyleb{%
  \font\crd@newfont=cmssbx10%
  \let\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%
```

Contributions for improving either the current fontstyle 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.