- NAME
- INTRODUCTION
- BS3ML DESCRIPTION
  - BS3ML CONFIGURATION FILE
  - STRUCTURE OF A BS3ML FILE
  - BS3ML COMMANDS
  - SYNTAX OF THE BS3ML COMMANDS
- META SETUP
  - EXAMPLES:
  - HOW IT WORKS ?
- THE BS3ML SOFTWARE
  - PREREQUESITES
  - COMPUTER PLATFORMS
  - EXTERNAL MIDI SOFTWARE
  - INITIAL INSTALL
  - USING THE BS3ML SOFTWARE
- DOCUMENTATION
  - MODIFYING AND REGENERATING THE DOCUMENTATION
- LICENCE
- AUTHOR
- ORGANIZATION
- VERSION
- CREATED
- COPYRIGHT

NAME

BS3ML - A DSL for programming the controls of the CME BITSTREAM 3X MIDI controler, and its associated software.

INTRODUCTION

As of today (2023/11/02), only two ways existed to program the "**CME Bitstream 3X**" MIDI Controler (Aka "BS3X").

Through the device's on-board Menu

A method that left over many interesting options, thus making this option sub-optimal for many configuration jobs.

Through the manufacturer's external graphical editor

A method that initially offered access to all possible options but suffers today significant limitations :

- As of today (2023/11/02), this editor no longer works on newer computer platforms like Silicon MacOS, BSDxxx, OpenBSD, Linux, etc.
- This editor allows to program BS3X controls one by one, thus making this approach inefficient and time consuming for extensive setups, once one is reminded that the BS3X MIDI controler offers 21 groups of 66 controls.

BS3ML, born on November, 2, 2023 remedy the situation by offering a modern, perene and efficient method to program the BS3X MIDI controler by means of BS3X specifications writen in a very specific programming language : the **BS3X Mini Language**, or **BS3ML**.

BS3ML DESCRIPTION

Programming the BS3X through the BS3ML narrows down to writing a text file comprised of a linear sequence of the "**BS3ML commands**" expressing how the BS3X controls should behave.

BS3ML CONFIGURATION FILE

- A BS3ML file is a text file no different from any source code file in general programming.
- A BS3ML file may bear the extension ".bs3ml" for easy access.

STRUCTURE OF A BS3ML FILE

- Empty lines, or lines starting with the pound sign (#) are ignored.
- Non empty lines must contain 1 and only 1 BS3ML command per line.
- BS3ML commands **CANNOT** be split in multiple lines.

BS3ML COMMANDS

CC()

The **CC()** command allows to program any of the 1386 BS3X controlers to have them send MIDI Control Change messages.

SYSEX()

The **SYSEX()** command allows to program any of the 1386 BS3X controlers to have them send MIDI System Exclusive messages.

LCD()

The **LCD()** command allows to associate a custom label to each of the 1386 BS3X controlers, so the BS3X displays such labels on its mini LCD when these controlers are being used.

SYNTAX OF THE BS3ML COMMANDS

CC()

The general syntax of the **CC()** command is :

Without options:

```
CC( BS3CTRL, USERGROUP, MIDICHANNEL, CCNUM, VMIN, VMAX, LABEL )
```

With options:

```
CC( BS3CTRL, USERGROUP, MIDICHANNEL, CCNUM, VMIN, VMAX, LABEL, qw(OPTIONS) )
```

Where:

- **BS3CTRL**

  The BS3X controler number (0 to 66) under definition..

- **MIDICHANNEL**

  The MIDI channel number (1 to 16) used by the MIDI Control Change message.

  Note: Using the value 0 allows the CC message to dynamically use the global MIDI channel actually selected by the BS3X, as displayed on the top left corner of the mini LCDi, and which can be dynamically set by the value knob.

- **CCNUM**

  The MIDI CC number (0 to 127). Please refer to the MIDI specs for more information.

- **VMIN**

  The minimum value of the MIDI CC controler (0 to 127).

- **VMAX**

  The maximum value of the MIDI CC controler (0 to 127).

  Note 1 : VMAX must be greater or equal to VMIN. Should you want the controler to send lesser values as you move up the knob or the slider, please use a curve option such as "Inverse Linear" (INV_LINEAR) or "Inverse logarithm" (INV_LOG).

  See below **OPTIONS**

  Note 2 : When refering to CC switches (CC num 64 to 69), it is recommended that you use VMIN=0 (off) and VMAX=127(on)

- **LABEL**

  This specifies the label the BS3X will display on its mini LCD when using the BS3X controler being defined.

  NOTES

  - The label is optional.

    Should you want **NOT** to specify a label, simply use an empty string, like "", or simply omit it !

  - The label is a string of 16 or less ASCII characters.

  - The string may contain *templates* that will be subject to dynamic expansion during compilation, as follows:

    - **%bs3**

      Will expand to the value of the BS3CTRL parameter.

    - **%g**

      Will expand to the value of the USERGROUP parameter.

- **%ch**

  Will expand to the value of the MIDICHANNEL parameter.

- **%cc**

  Will expand to the value of the CCNUM parameter.

- **%vmin**

  Will expand to the value of the VMIN parameter.

- **%vmax**

  Will expand to the value of the VMAX parameter.

After expansion, the label is truncated to 16 characters.

- **OPTIONS**

  Please refer to the BS3X User guide for a detailed explanation of these options.

  It is mandatory to use the form **qw(OPTION OPTION...)** to specify the options to use.

  Options should be separated by space only : no '.' nor ',' or other separator allowed.

  Unknown options will be silently discarded.

  The available options are :

  - LINEAR
  - INV_LINEAR
  - LOG
  - INV_LOG
  - RND
  - USR0
  - USR1
  - TOGGLE
  - PUSH
  - SCENE_EXCLUDE
  - SCENE_INCLUDE
  - AUTOSEND
  - NO_AUTOSEND
  - USB_OUT
  - NO_USB_OUT
  - MIDI1_OUT
  - NO_MIDI1_OUT
  - MIDI2_OUT
  - NO_MIDI2_OUT
  - NO_MIDIDELAY
  - MIDIDELAY30
  - MIDIDELAY60
  - MIDIDELAY90
  - CHAIN<**BS3X_control_number_expression**>

    This option allows to **chain** the controler under definition with the BS3X controler whose number will result from dynamically computing the value of **BS3X_control_number_expression** during compilation (see **EXPRESSION**

EXPANSION below).

- CROSSFADE<**BS3X_control_number_expression**>

  This option allows to **crossfade** the controler under definition with the BS3X controler whose number will result from dynamically computing the value of **BS3X_control_number_expression** during compilation (see **EXPRESSION EXPANSION** below).

EXPRESSION EXPANSION

Within **CHAIN** and **CROSSFADE** options, the term **Bs3x_control_number_expression** can be any valid arithmetic expression formed using standard '+', '-' and '*' operators and templates as seen above for LABEL template expansion.

For instances, given BS3X controler equals 8 :

- CHAIN%bs3+1

  will be expanded as "CHAIN9"

- CHAIN%bs3+(2*8)+1

  will be expanded as "CHAIN25"

- etc.

EXAMPLES OF CC() COMMANDS

1. Configure the leftmost BS3X slider (40) on user group 4 to control the volume (CC 7) on fixed MIDI channel 8 and to display the string "Volume chan#8" when using the slider :

```
CC(40, 4, 8, 7, 0, 127, "Volume chan#8")
```

or

```
CC(40, 4, 8, 7, 0, 127, "Volume chan#ch%")
```

2. Configure the joystick on user group 2 to control the BS3X X-Y joystick to control the volume (vertically) and pan (horizontally) on the BS3X selected MIDI channel :

```
CC( 6, 2, 0, 7,  1, 127, "Volume")
CC( 7, 2, 0, 10, 1, 127, "Pan")
```

3. Configure the left slider to control volume on **both** MIDI channels 1 and 2 on user group 3:

```
CC( 40, 3, 1, 7, 1, 127, "Volume ch#1", qw(CHAIN41) )
                                        ^^^^^^^^^^^

CC( 41, 3, 2, 7, 1, 127, "Volume ch%2")
```

SYSEX()

The BS3ML **SYSEX()** command allows to program any of the 1386 BS3X controlers to have them send a MIDI System Exclusive message when used.

The general syntax and semantics of the **SYSEX()** command is identical to that of the **CC()** command, where the parameter CCNUM is replaced by the parameter SYSEX_STRING, that is:

Without options:

```
SYSEX( BS3CTRL, USERGROUP, CHANNEL, SYSEX_STRING, VMIN, VMAX, LABEL )
```

With options:

```
SYSEX( BS3CTRL, USERGROUP, CHANNEL, SYSEX_STRING, VMIN, VMAX, LABEL, qw(OPTIONS) )
```

Where:

- **SYSEX_STRING** is a double-quoted string of the form :

```
"SYX_CHE<lt>Midi Channel PositionE<gt>_VVE<lt>Value positionE<gt>=F0_E<lt>SYSEX HEX STRINGE<gt>_F7"
```

- Other parameter fields are semantically identical to those of the CC() command above.

EXAMPLES OF SYSEX() COMMANDS

1. Here is how to include the MIDI channel at position 6 (CH6), and the BS3X controler value on 1 byte at position 8 (VV8) of the SYSEX HEX STRING :

```
             1  2  3  4  5  6  7  8  9
             |  |  |  |  |  |  |  |  |
             v  v  v  v  v  v  v  v  v
"SYX_VV8_CH6=F0_43_10_4C_08_00_72_00_F7"
       ^   ^                ^^    ^^
       |   |                 |     |
       |    -----------------      |
        ---------------------------
```

2. Here is how to program the leftmost BS3X slider on group 3 to send the dynamic MIDI channel at position 6 (CH6), and the BS3X controler value on 2 bytes at position 8 and 9 (VV8,9) of the SYSEX HEX STRING :

```
                                1  2  3  4  5  6  7  8  9  10
                                |  |  |  |  |  |  |  |  |  |
                                v  v  v  v  v  v  v  v  v  v
SYSEX(40, 3, 0, "SYX_VV8,9_CH6=F0_43_10_4C_08_00_72_00_00_F7", 1, 127, "EQ Low ch#%ch")
                        ^^^   ^                ^^    ^^ ^^
                         |    |                 |     |  |
                         |     -----------------      |  |
                          -------------------------------
```

3. Here is how to program a Simple SYSEX that includes none of MIDI channel not the value of the BS3X controler :

```
"SYX=F0_..._F7"
 ^^^
```

LCD()

The **LCD** command allows to associate a label to any of the 1386 BS3X controlers which the BS3X will display on its mini LCD when using these controlers.

The syntax of the **LCD()** command is :

LCD(LABEL, BS3CTRL, USERGROUP, [CHANNEL, CCNUM, VMIN, VMAX])

Where:

- The semantics of the various fields is similar to that of the **CC()** command.
- The fields CHANNEL, CCNUM, VMIN, VMAX are optional.
- The field **LABEL** is subject to template expansion similar to that of the **CC()** command, given that templates associated with unsupplied parameters will be silently discarded.

  For instance :

  - `LCD("Volume %ch", 8, 0, 8)`

    will expand label to "Volume 8"

  - as

    `LCD("Volume %ch", 8, 0)`

    will expand label to "Volume" (template %ch is discarded)

META SETUP

Configuring more than a handful of BS3X controlers might still prove tedious, time consuming and error prone.

Indeed, such MIDI controlers as the BS3X are often used to perform the same set of functions across groups of controlers (eg. rows or columns), only with different MIDI channels.

BS3ML therefore offers a convenient and efficient syntax to address these situations by allowing groups of controlers to be programmed at once.

EXAMPLES:

When glancing at these examples, please pay attention at how groups of values differ from single values by using a bracket syntax.

1. ```
   CC( [40..47], 0, [1], 7, 1, 127, "Volume ch#%ch")
       ^^^^^^^^     ^^^
        BS3X      IMPLICIT SET OF MIDI
        CTRLS     CHANNELS STARTING AT 1
   ```

   will program each of the BS3X 8 sliders (40 to 48) to send the MIDI CC volume message to MIDI channels 1 to 8, exaclty as if was suppied these 8 CC() commands :

   ```
   CC(40, 0, 1, 7, 1, 127, "Volume ch#1")
   CC(41, 0, 2, 7, 1, 127, "Volume ch#2")
   CC(42, 0, 3, 7, 1, 127, "Volume ch#3")
   CC(43, 0, 4, 7, 1, 127, "Volume ch#4")
   CC(44, 0, 5, 7, 1, 127, "Volume ch#5")
   CC(45, 0, 6, 7, 1, 127, "Volume ch#6")
   CC(46, 0, 7, 7, 1, 127, "Volume ch#7")
   CC(47, 0, 8, 7, 1, 127, "Volume ch#8")
   ```

2. ```
   CC([40,42,44], 0, [ 8,10,12], 7, 10, 100, "Volume ch#%ch")
       ^^ ^^ ^^       ^^ ^^ ^^
       |  |  |        |  |  |
        --|--|--------   |  |
           --|-----------   |
   ```

```
              --------------
              Explicit set of values leading to a 1-1 mapping
```

will be like writing :

```
CC(40, 0,  8, 7, 10, 100, "Volume ch#8")
CC(42, 0, 10, 7, 10, 100, "Volume ch#10")
CC(44, 0, 12, 7, 10, 100, "Volume ch#12")
```

3. `CC([40..42], 0, 8, 7, 10, 100, "Volume ch#%ch")`

   Will have the 3 leftmost sliders be clones of each other, each one sending the same CC volume message to the MIDI channel 8, like writing :

```
CC(40, 0,  8, 7, 10, 100, "Volume ch#8")
CC(41, 0,  8, 7, 10, 100, "Volume ch#8")
CC(42, 0,  8, 7, 10, 100, "Volume ch#8")
```

4. `CC( [8,16,32,40], 0, [1..8], [93,91,10,7], 1, 127, ["Echo ch#%ch", "Reverb ch#%ch", "Pan ch#%ch", "Volume ch#%ch"])`

   In this more complex example, BS3 controlers, LABELs, and CCNUMs controlers define sets of 4 values that will be mapped 1 to 1 for each of the 8 midi channels.

   For each subsequent MIDI channels (2 to 9), only the BS3 control numeros will be incremented. CCNUMs will loop, as the generic labels thanks to the convenient available templating. No other semantics would make any sense :

```
 MIDI CHANNEL          BS3_CONTROLS            CCNUMBERS

     1        ---->     8  16  32  40         93  91  10  7
                        ^^  ^^  ^^  ^^         ^^  ^^  ^^  ^^
                        |   |   |   |          |   |   |   |
                         ---|---|---|----------    |   |   |
                             ---|---|--------------    |   |
                                 ---|------------------    |
                                     ----------------------


     2        ---->     9  17  33  41         93  91  10  7
                        ^^  ^^  ^^  ^^         ^^  ^^  ^^  ^^
                        |   |   |   |          |   |   |   |
                         ---|---|---|----------    |   |   |
                             ---|---|--------------    |   |
                                 ---|------------------    |
                                     ----------------------

    ...

     8        ---->    15  23  39  47         93  91  10  7
                        ^^  ^^  ^^  ^^         ^^  ^^  ^^  ^^
                        |   |   |   |          |   |   |   |
                         ---|---|---|----------    |   |   |
                             ---|---|--------------    |   |
                                 ---|------------------    |
                                     ----------------------
```

   This one and only **META CC()** command stands as an highly efficient syntax that does the same as writingi individually the following 32 CC() commands:

```
CC( 8, 0, 1, 93, 1, 127, "Echo ch#1")
CC( 9, 0, 2, 93, 1, 127, "Echo ch#2")
CC(10, 0, 3, 93, 1, 127, "Echo ch#3")
CC(11, 0, 4, 93, 1, 127, "Echo ch#4")
CC(12, 0, 5, 93, 1, 127, "Echo ch#5")
CC(13, 0, 6, 93, 1, 127, "Echo ch#6")
CC(14, 0, 7, 93, 1, 127, "Echo ch#7")
CC(15, 0, 8, 93, 1, 127, "Echo ch#8")

CC(16, 0, 1, 91, 1, 127, "Reverb ch#1")
CC(17, 0, 2, 91, 1, 127, "Reverb ch#2")
CC(18, 0, 3, 91, 1, 127, "Reverb ch#3")
CC(19, 0, 4, 91, 1, 127, "Reverb ch#4")
CC(20, 0, 5, 91, 1, 127, "Reverb ch#5")
CC(21, 0, 6, 91, 1, 127, "Reverb ch#6")
CC(22, 0, 7, 91, 1, 127, "Reverb ch#7")
CC(23, 0, 8, 91, 1, 127, "Reverb ch#8")

CC(32, 0, 1, 10, 1, 127, "Pan ch#1")
CC(33, 0, 2, 10, 1, 127, "Pan ch#2")
CC(34, 0, 3, 10, 1, 127, "Pan ch#3")
CC(35, 0, 4, 10, 1, 127, "Pan ch#4")
CC(36, 0, 5, 10, 1, 127, "Pan ch#5")
CC(37, 0, 6, 10, 1, 127, "Pan ch#6")
CC(38, 0, 7, 10, 1, 127, "Pan ch#7")
CC(39, 0, 8, 10, 1, 127, "Pan ch#8")

CC(40, 0, 1,  7, 1, 127, "Volume ch#1")
CC(41, 0, 2,  7, 1, 127, "Volume ch#2")
```

```
CC(42, 0, 3,  7, 1, 127, "Volume ch#3")
CC(43, 0, 4,  7, 1, 127, "Volume ch#4")
CC(44, 0, 5,  7, 1, 127, "Volume ch#5")
CC(45, 0, 6,  7, 1, 127, "Volume ch#6")
CC(46, 0, 7,  7, 1, 127, "Volume ch#7")
CC(47, 0, 8,  7, 1, 127, "Volume ch#8")
```

HOW IT WORKS ?

1) Any field meant to indicate multiple values, implicitely, or explicitely, must use the bracket [] syntax to gather the different values for that field.

Two forms of brackets are accepted :

- Explicit list of discrete values

  Like [40,41,42,43,44,45,46,47]

- Intervals

  Like [40..48], interpreted as a 1-increment between values.

2) The software then looks for all fields listing multiple values,

3) Then the values are taken in parallel one at a time for every set of values, simply incrementing by 1 any missing value, starting with the last value used.

Note: The BS3ML META syntax works identically for all BS3ML commands : CC(), SYSEX(), LCD()

THE BS3ML SOFTWARE

The BS3ML language, per se, cannot do much on its own :(

There is a software, the BS3ML software, whose only role is to take any any BS3ML specs (aka BS3ML preset) and translate it into its equivalent SysEX file to be sent to the **CME Bitstream 3X** Midi device for reconfiguring its various controlers (knobs, sliders, buttons).

The **BS3ML software** is made of 3 binaries :

1 The **BS3ML compiler**

   Entirely written in Perl, the **BS3ML compiler** works to translate any BS3ML preset into its equivalent SysEX file.

2 The **BS3ML launcher**

   The **BS3ML launcher** is a wrapper written in Bash, aimed at easing the overall process of generating generating the SysEX files and sending it to the BS3X device.

3 The **BS3ML documentation generator**

   Should you plan to correct, modify or extend the BS3ML documentation, there is a binary that regenerates all the BS3ML documentation.

PREREQUESITES

The **BS3ML software** relies on using modern versions of both the **Bash shell** and the **Perl** programming language.

The **BS3ML software** was developped against :

- Bash version 5.1.16
- Perl version v5.34.0

COMPUTER PLATFORMS

The BS3ML software should work flawlessly on any modern Un*x platform given the prerequesites are satisfied.

EXTERNAL MIDI SOFTWARE

The BS3ML software relies on simple 3rd party MIDI software to send the SysEx files to the CME/WaveIdea Bitstream 3X Midi device.

- MacOS

  sendmidi v1.1.0

  https://github.com/gbevin/SendMIDI

- Linux

  amidi command, usually from the "alsa-utils" Linux package

INITIAL INSTALL

Installing the BS3ML software is a trivial process :

0. Open a terminal session to run the Bash shell :

```
$ exec bash
```

1. Move the downloaded BS3ML zip archive wherever you plan to use the software, say "<myfolder>" :

```
$ mv bs3ml.zip <myfolder>
```

2. Position your Bash shell to the folder where you moved the archive :

```
$ cd <myfolder>
```

3. unzip the BS3ML archive

```
$ unzip bs3ml.zip
```

USING THE BS3ML SOFTWARE

Once installed, the time has come to use the BS3ML software.

Most of the BS3ML operations can be realized using the **BS3ML wrapper** : the **bs3ml** command.

- Position your Bash shell into the bs3ml root install :

  ```
  $ cd <myfolder>/bs3ml
  ```

- Bootstrap the bs3ml environment :

  ```
  $ source usebs3ml
  ```

- Help :

  ```
  $ bs3ml -h
  ```

- Create/modify a bs3ml **p**reset and (re)generate the SysEx file :

  ```
  $ bs3ml -p my_preset
  ```

- **G**enerate the sysex file from your newly defined/modified preset using the bs3ml wrapper (without sending it to the BS3X) :

  ```
  $ bs3ml -g my_preset
  ```

  This generates a SysEx file named "my_preset.syx" into the BS3ML software "presets" subfolder.

  or :

- Generate the sysex file from your newly defined/modified preset using the bs3ml wrapper and send the resulting sysex file to your BS3X :

  ```
  $ bs3ml my_preset (no option)
  ```

  or :

  ```
  $ bs3ml -ps my_preset
  ```

  This generates a SysEx file named "my_preset.syx" into the BS3ML software "presets" subfolder and sends it to your BS3X device :

- **S**end, or resend, a preset sysex to the BS3X

  ```
  $ bs3ml -s my_preset
  ```

  This generates a SysEx file named "my_preset.syx" into the BS3ML software "presets" subfolder and sends it to your BS3X device :

- **E**dit the BS3ML documentation

  ```
  $ bs3ml -e
  ```

- Regenerate the BS3ML **d**ocumentation :

  ```
  $ bs3ml -d
  ```

- Create an archive of the BS3ML sofware including all the BS3ML presets :

  ```
  $ bs3ml -z
  ```

DOCUMENTATION

The BS3ML documentation exists in multiple forms in the "doc" subfolder.

- POD

  Perl Plain Old Documentation -- This is the marked-up master documentation from which all the other formats are derived.

- README

  To read the README :

  ```
  bs3ml -t
  ```

- MAN

  This is the Un*x **m**an page of the BS3ML documentation.

  To consult :

  ```
  bs3ml -m
  ```

- PDF

  The PDF version, in the "doc/pdf" subfolder.

- HTML

  The HTML version, in the "doc/hml" subfolder.

  Use any modern web browser to read it.

MODIFYING AND REGENERATING THE DOCUMENTATION

Should you want to correct, modify, or enhance the actual documentation, simply run the following command:

```
bs3ml -e
```

To regenerate the modified documentation :

```
bs3ml -d
```

Note: Please be kind enough to send us any modified documentation.

LICENCE

The BS3ML software is distributed as a nonpaying freeware for your own use only.

Using the BS3ML software implies that you have implicitely and formally agreed with the following terms :

- You have the right to read and study the BS3ML software source code, to correct it, to modify it or enhance it to fit your wishes.

  Please be kind enough to send us any modified version.

- However, you do **not** have the right to redistribute or sell the original BS3ML software as you received it, in any form, in whole or in parts, in any medium.

- You do **not** have the right to redistribute or sell any modified version of the BS3ML software, in any form, in whole or in parts, in any medium.

- You do **not** have the right to clone the BS3ML software with the intent to make it your ownn, or to integrate any part of the BS3ML software in any other software work.

AUTHOR

Dr. Franck Porcher, Ph.D., franck.porcher@protonmail.com

ORGANIZATION

Franckys Software Engineering and Publishing - www.franckys.com

VERSION

1.0

CREATED

04.10.2023 17:46:04

COPYRIGHT

Copyright (C) 2023 - Franck PORCHER - All rights reserved