# MRO (SharpEye) to SCORE Converter

Christian Fremerey (fremerey@iai.uni-bonn.de)

March 3, 2009

## 1 General Information

MRO to Score Converter is a Java command-line application that converts symbolically encoded musical scores from the MRO (as used in the SharpEye Music Reader) format to file formats that can be used with the SCORE engraving software.

**Version:** 0.2.1
**Date:** 03.03.2009
**Input Format:** MRO (version 3100, as output by the liszt OMR engine used in SharpEye 2.68)
**Output Formats:** MUS (binary format for SCORE 3.11), PMX (ascii format for SCORE 4???)

Special thanks to Craig Sapp and Andreas Kornstädt for their great help and assistance in creating and revising this piece of software.

## 2 Licence



## 3 Requests / Bug Reports

- (in `sonata04-2_Page_1.mro`:) Stem length for chords with staff offsets is not accurate. It seems to be a mixture of multiple problems:
  1. The value for the flagend in the MRO data is inaccurate (not a constant offset). (The SharpEye Editor uses default stem lengths for this case.)
  2. Despite problem 1, the additional stem length added to make up for the y-distance between the regular staff and the offset staff is also inaccurate. It seemed a bit too long in the examples. This might be caused by either the value for scalestep_inch to be inaccurate, or by the distance of the staffs to not come out as it should (it should match the distance specified by the MRO).

# 4 Change Log

Version 0.2.1:

- options -od and -ofs now do not enable the use of multiple input files (using the default inputFileMask if not specified otherwise) anymore
- duration parameter for grace notes set to 65.0f
- fixed bug that caused the vertical position of grace note beams to be off by +100.0 if the note vertical position was 0.0f.

Version 0.2.0:

- changed project name from `MRO_to_SCORE_Converter` to `mro2score`
- make staff bracket optional, and only offer it when there are two or more staffs that are not linked (braced)
- added option `-noSystemBracket`
- fixed bug that caused notes with flags always have a duration of 1.0
- make clefs cue size if they are not at the beginning of a staff
- added option `-noCueClefs`
- added option `-appendToEndOfPMXFiles=[String]`
- slur curvature is now converted
- added horizontal alignment of notes/rests with similar horizontal position (default: on)
- added option `-noHorizontalAlignmentForNotesAndRests`
- fixed bug causing accidental offsets to alter the type of accidental (and the offset to be wrong)
- fixed bug in the P4 value of grace notes: notes with negative vertical position now get -100 added instead of +100
- added option `-version`
- fixed bug slur-staff mapping that would cause slurs always to be mapped to the top staff of a system
- primary beams may have more than 1 beam even in presence of secondary beams
- fixed bug that caused missing secondary beams at the end of a note group
- fixed bug that caused missing displacement of tertiary beams
- added support for notes with staff offset
- for notes with more than one flag (16th or shorter), SCORE seems to add the additional flags on top of the stem, making the stem longer as specified (about 1.5 scalesteps per flag). compensation added.
- fixed bug in horizontal displacement parameter for notes with flipped position

- horizontal displacement from the horizontal alignment position of x-clusters is only used if it significant (larger than 3 scoreunits).
- fixed grace note beams
- all stems in beamed groups should now correctly attach to beams (except for chords with staff offset), and it should work with and without stem length quantization
- stretch staff lengths to 0-200 for staffs from 6-194, added option to disable this
- barline snapping for bars within 2 scoreunits of staff ends, added option to disable this
- no traling 0.0 parameters in file export
- shorter option names for commonly used options
- fixed bug in accent enumeration
- durations of whole rests now follow the current time signature

# 5 Usage

MRO to Score Converter is a Java command-line application compiled with Java 1.6. To run the converter, you need a Java Runtime Environment 1.6 or higher. The application consists of one file named `mro2score-0.2.1.jar`.

Executing the application without any parameters will print the usage screen.

```
Usage: java -jar mro2score-0.2.1.jar [options]

Example: java -jar mro2score-0.2.1.jar -inputFile=test.mro

Note: You must at least specify either -inputFile=[String] to convert
a single MRO File, or one of -inputDir=[String] or -inputFileMask=[String]
to batch-convert multiple input files.

Options:
  -inputFile=[String] or -if=[String]
      Path and filename of a single input MRO file
  -dpi=[Integer]
      Resolution of the original image file in dots per inch
      Default: 600
  -inputDir=[String] or -id=[String]
      Path to directory for input MRO files. The inputFileMask will be
      used to find input MRO files from there.
      Default: "."
  -inputFileMask=[String] or -ifm=[String]
      Pattern for selecting input files from inputDir.
      Use "*" as wildcard, e.g. "*.mro" will select all MRO files
.      Default: "*.mro"
  -outputDir=[String] or -od=[String]
      Path to directory for writing output files.
      Default: "."
  -outputFilenameScheme=[String] or -ofs=[String]
```

```
    Pattern that specifies the filenames of the output files.
    Use the following escape sequences:
        %f = filename of the input MRO file without extension
        %p = number of page in current MRO file (starts with 001)
        %c = page counter for all input MRO files (starts with 001)
    The extension ".mus" or ".pmx" are be appended to the end of
    the scheme.
    Default: "%f_p%p"
-stemLengthQuantization=[Float] or -slq=[Float]
    Specify a stem length quantization in [scalesteps].
    Default: "0.5"
-appendToEndOfPMXFiles=[String] or -ap=[String]
    String to append to the end of each PMX File.
    Use the following escape sequences:
        \n = new line
    Default: ""
-version or -v
    Output converter version number and date.
-noPMX
    Do not write PMX output files.
-noMUS
    Do not write MUS output files.
-noStemLengthQuantization or -n1
    Disable stem length quantization.
-noHorizontalAlignmentForNotesAndRests or -n2
    Disable horizontal alignment of notes and rests with similar horizontal position in the MRO
-noSystemBracket or -n3
    Disable output of they system bracket for each system.
-noCueClefs or -n4
    Disable making clefs cue size if they do not stand at the beginning of a staff.
-noNormalizeStaffLengths or -n5
    Disable normalization of staffs to 0-200.
-noSnapOuterBarlines or -n6
    Disable snapping of outer barlines to the staff borders.
```

# 6 Conversion Details

It is assumed that input MRO files are created by applying OMR on input image files. Each image file is considered as one page. MRO files can contain more than one page. The converter tries to create SCORE data that closely matches the original layout and symbol positions of the original image file.

## 6.1 Horizontal Positions, Paper Size and Global Scale

To be able to reconstruct the paper size of the original input image file, the converter takes the image file resolution in `[dpi]`(dots per inch) as an input parameter.

The origin of the coordinate system of a page in SCORE is the bottom left corner. Horizontal positions are measured in a unit that in this converter are called *scoreunits*. When not changing the default value of 1.0 for the global scale, scoreunits convert into inch by means of

$$200\texttt{[scoreunits]} = 7.5\texttt{[inch]}.$$

The absolute horizontal position on the paper at 0 [scoreunits] is determined by the left margin, which is specified in [inch] (can be set in the printing options, default value is 0.5).

Many functions in SCORE are designed to work with the staff lines going from 0 [scoreunits] to 200 [scoreunits]. In fact, it is not possible to put any objects on a staff in SCORE at positions $> 218$ [scoreunits]. To be able to have staffs range from 0 to 200 [scoreunits] but not having a width of 7.5 [inch], SCORE offers a global scale parameter (called *size* in the printing options and often written as size_factor in the converter). This global scale parameter uniformely scales the complete output, but it does not affect the page margins.

The converter calculates page margin and global scale parameters such that the leftmost staff starts at 0 [scoreunits] and the rightmost staff ends at 200 [scoreunits] while the original width of the staffs is preserved. Since neither the MUS nor the PMX format have fields for saving these parameters, they are integrated into the files using a custom mechanism explained in the following section.

## 6.2 Custom Non-Standard Information in the Output Files.

Reconstructing the paper size and layout of the original input image file requires several parameters to be saved that are not part of the default file formats. Currently, these parameters are

- the image resolution in [dpi],
- the page width in [inch]
- the page height in [inch]
- the left margin of the page in [inch]
- the bottom margin of the page in [inch]
- the global scale of the page (unitless).

To get the original page layout, these values have to be used as parameters when printing in the SCORE software. Note that SCORE Preview 3.11 did not seem to put information about the page width and height into output *.eps files. The page dimensions can be set manually when viewing the *.eps in GhostScript.

In the PMX format, the values are stored at the top of the file using one text line starting with 99 for each parameter. Example:

```
99 dpi=600.0
99 pageWidth_inch=8.538333
99 pageHeight_inch=11.68
99 marginLeft_inch=0.57485807
99 marginBottom_inch=0.75
99 size_factor=1.0045098
```

In the MUS format, the values are stored at the beginning of the trailer at the end of the file. To indicate the existance and beginning of the custom fields, a "magic number" is written followed by a version number and the values of interest. In detail, the trailer written by the converter looks like this:

1. `CUSTOM_TRAILER_MAGIC_NUMBER`
   (32-bit float little-endian, value = −36.001)
2. `CUSTOM_TRAILER_VERSION`
   (32-bit float little-endian)
3. `dpi`
   (32-bit float little-endian)
4. `pageWidth_inch`
   (32-bit float little-endian)
5. `pageHeight_inch`
   (32-bit float little-endian)
6. `marginLeft_inch`
   (32-bit float little-endian)
7. `marginBottom_inch`
   (32-bit float little-endian)
8. `size_factor`
   (32-bit float little-endian)
9. `trailer_entry_1`
   (32-bit float little-endian, default value 0.0)
10. `trailer_program_serial_number`
    (32-bit unsigned integer, default value 4002041)
11. `trailer_program_version`
    (32-bit float little-endian, default value 3.0)
12. `trailer_measurement_code`
    (32-bit float little-endian, default value 0.0 for `[inch]`)
13. `trailer_num_32bit_entries`
    (32-bit float little-endian)
14. `file_end_marker`
    (32-bit float little-endian, value = −9999.0)

Note: The custom fields must be added at the top of the trailer. They cannot be added at the bottom of the trailer, because it seems that SCORE reads the trailer in reverse order and assumes that the first four fields it sees are the default fields.

## 6.3 Vertical Positions and Sizes

The vertical positions of the bottom staff is determined by the bottom margin of the page plus an offset parameter P4 that specifies an offset in scalesteps relative to the staff height. Unfortunately, the SCORE reference manual is not very clear about the definition of this parameter and it says it should not be used to alter the position of the bottom staff. Therefore, the parameter P4 is not used in the converter. Instead, the vertical position of the bottom staff is handled by setting the bottom margin of the page to the offset of the bottom of the bottom staff from the bottom of the page. The positions of the remaining staffs are set using the P10 parameter, which allows specifying an absolute offset from the bottom staff.

The default staff height in SCORE is 9 millimeter. It can be changed by setting parameter P5 of the staff object. The formula for the staff height in millimeters is

$$\texttt{staff\_height[mm]} = (10 \cdot \texttt{P5} - 1) \cdot \texttt{size\_factor}.$$

In the conversion, the `P5` values are calculated to result in staffs with the same height as in the original image. Note that the calculation is not 100% accurate because in both SCORE and MRO it remains unclear how exactly the staff height is defined. The uncertainty is about the order of one staffline thickness, because the staff height might or might not include the staffline thickness. The same uncertainty applies to the vertical positions of the staffs.

In SCORE, vertical positions relative to staffs are measured in *scalesteps*. A scalestep is one step in the vertical raster of the staff, i.e. the amount a notehead is moved up or down when the vertical position is changed one step in the raster. The converter estimates the scalestep unit by the calculation

$$\texttt{scalestep[inch]} = \texttt{staff\_height[inch]}/8.$$

Note that due to the uncertainty in the staff height, the accuracy of the scalestep estimation is also uncertain.

## 6.4 Symbol and Text Position and Size

For many types of symbols, the MRO contains information about the center position, but it doesn't contain information about the width and height. This is the case for clefs, key signatures and dynamic symbols. For text, a height value is given, but it remains unclear what unit it is. The documentation claims it is a point value [pt], but that doesn't seem to be right. In fact, SharpEye has no chance to calculate text height in [pt] without knowing the resolution of the image files (which it doesn't). There is no information about the width of text.

The converter divides the value acquired from the MRO by a factor of 3.4 to get an estimated point height. Actually the resulting height is a bit smaller than the original, which is preferred to avoid horizontal overlap of text. In SCORE, the text size depends on the staff size of the staff it is bound to. The formula here is:

$$\texttt{text\_size[pt]} = (\texttt{staff\_size\_P5} \cdot 10 + 3.2) \cdot \texttt{text\_P6}.$$

Since in SCORE horizontal symbol locations are specified by means of their left border, the correct values have to be guessed by estimating the symbol width. Note that this might lead to inaccuracies. For example in key signatures, the width depends on the amount and width and spacing of the accidentals.

## 6.5 Key Signatures

In the MRO format, key signatures are encoded by a center point and an integer for specifying the key. There is no explicit way to encode existance and amount of naturals. The converter tracks the key signatures throughout each staff. In case the key signature changes within a staff from $p > 0$ accidentals to 0 accidentals, the converter will put $p$ naturals at the corresponding position.

## 6.6 Clefs

In it's current version, the liszt recognition engine can recognize treble, alto, tenor and bass clefs. Other clefs like baritone are not recognized as such. The MRO format has a pitch position parameter for clefs that is recommended to use in preference of the vertical center position.

The converter uses the pitch position from the MRO to distinguish between alto and tenor clefs. In other cases, the value is ignored. The converter never creates clefs with a vertical offset.

The MRO format also lists treble clefs with octavation up and down. Since the SCORE reference manual doesn't list a treble clef with octavation up, a plain treble clef is used instead.

## 6.7 Accidental Offsets

Accidental Offsets in the MRO are measured as distances from the left edge of the notehead to the center of the accidental. To convert this to an offset value in SCORE, the converter has to estimate the accidental width and the default note-accidental spacing.

The fractional part of P5 for notes works like this:

- .0000 to .9000 is an offset to the left (away from the notehead).
- .90001 to 0.9999 is an offset to the right of the default accidental position, with 0.9999 touching the notehead.

## 6.8 Slurs / Ties

Slurs in the MRO are approximated as arcs with given start point, end point and radius. The converter uses the radius value to determine a curvature for the slurs/ties in SCORE. The converter does not discriminate between slurs and ties.

In the MRO format, slurs are mapped to systems. In SCORE however, slurs are mapping to staffs. The converter maps slurs to staffs by using the following algorithm:

- If the slur starts below the bottom staff of the system it is mapped to, it is mapped to the bottom staff of that system.
- If the slur starts above the top staff of the system it is mapped to, it is mapped to the top staff of that system.
- If the slur starts inside a staff, it is mapped to that staff.
- If the slur starts in between two staffs of the system it is mapped to, it gets mapped to the staff it "faces".

## 6.9 Stems

In upstem chords, only the bottom note gets a stem. In downstem chords, only the top note gets a stem. The end position of stems are encoded in the MRO as locations relative to the top left corner of the staff. In SCORE, the stem length is declared as a value added to some "regular" stem length in `[scalesteps]`. The SCORE reference manual says that the regular stem length is one octace, which probably means 7 scalesteps measured from the vertical center of the note head to the tip of the stem. The parameter for the stem length is based on this assumption. For grace notes, the assumed regular staff length is 4.5 scalesteps.

The converter allows stem lengths to be quantized with a customizable resolution. In the default settings, this feature is turned on. The default value is 0.5 `[scalesteps]`.

## 6.10 Braces and Brackets

MRO files do not contain explicit information about braces or square brackets used to group staffs into systems. However, braces are implicitly encoded as staffs being linked-up in the MRO. If the converter finds a group of linked-up staffs in the MRO, it will put a brace to the left side of the staffs. Square brackets and other indicators for staff groups in the original image cannot be reconstructed from the MRO data.

The following heuristic is used:

- If the system has only one staff, don't add any brace or bracket. Also don't add a plain barline at the system start.
- If the system has two staffs, assume it is a piano system. Add a brace but no bracket. Add a plain barline at the system start.
- If the system has three or more staffs, put a brace to staffs that are marked as "joined" in the MRO. Unless all staffs are joined together as one, add square brackets to visually indicate the system boundaries.

## 6.11 Beams

The encoding of beams is quite different in MRO and SCORE. In the MRO, for each member of the beamed group there are two integer numbers specifying the amount of strokes going from the stem of that member to the left and right. In SCORE, one has to use a secondary or tertiary beam to represent beamed groups with different numbers of strokes.

The converter handles this case by the following rules:

- If the number of strokes is constant, use a single beam object without secondary or tertiary beam.
- If the number of stroke changes is 1, use a single beam object with a secondary beam.
- If the number of stroke changes is 2, use a single beam object with a secondary and tertiary beam.
- If the number of stroke changes is $> 3$, use more than one beam object. Here, every beam object created has an identical primary beam connecting all the members. Each secondary and tertiary beam is then used to account for one stroke change.

Beam offsets in SCORE are supported by the converter. However, there are situations, where the beam offsets are not flexible enough to account for the beam in the MRO data. That is because for upstem chords, the beam offset always goes to the left, and for downstem chords, the beam offset always goes to the right. If for example an offset to the left is needed in a downstem chord, this offset will be created using the horizontal position parameter instead.

Staff offset is supported for notes heads by the converter. However, there is a problem with determining the correct staff length in this situation (see Section 3).

### 6.12 Text Encoding

The converter assumes that the input MRO files are encoded using the `ISO-8859-1` encoding. The characters in the output PMX and MUS files are encoded using the `US-ASCII` encoding. The converter replaces many (but not all) of the special characters to SCORE escape sequences according to the SCORE reference manual.

For the PMX format, text is encoded inline as a type 16 object:

- Parameter 12 = String length of the text
- Parameter 13 = Parameter 12
- Parameter 14 = The text String (trimmed)

In the MUS format, text is also encoded as a type 16 object. It consists of 13 32-bit float parameters plus the text represented by 8-bit per character (using the US-ASCII encoding). If the text length is not a multiple of 4, then zero-bytes are added at the end to fill up, such that the total object length is an integer multiple of 32-bit words.