**Documented Design**

Fully or nearly fully articulated design for a real problem, that describes how all or almost all of the key aspects of the solution/investigation are to be structured/are structured.

**The Problem and Solution**

The problem is to combine multiple music macro language text files into a single multi-channel MIDI file. My chosen approach to this problem is to write two terminal programs that would be used in conjunction. The first program, named “mmltomidi”, would take a single MML file and output a single single-channel MIDI file. The second program, named “catmidi” would take multiple single-channel MIDI files and combine them to create a single multi-channel MIDI file.

**Target Hardware**

The programs are will be written for Unix based operating systems such as OSX and Linux, as this is what I will be developing them on, and I am also used to working with these systems. Also, the programs will be written to run on little endian processors – this is relevant because the endianess of many values are flipped in the programs.

**Overall System Design**

The following tables are IPSO charts for the mmltomidi and catmidi programs respectively:

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Processes | Storage | Outputs |
| MML text file | MML text | Single-channel MIDI file | Success message |

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Processes | Storage | Outputs |
| Single-channel MIDI files | MIDI file contents | Multi-channel MIDI file | Success message |

The figure below shows the same information as in the tables above, but as flow chart:

F:\School work\A Level\Computer Science\MML-To-Midi-Project\Project Documentation\Diagrams\File_Data_Flow.png

**Music Macro Language Design**

**Introduction**

The music macro language (MML) is a music description language that has been in use since 1978, although this was an early version. There has never been an official specification, so each implementation varies slightly, and over the years the language has evolved. The MML to MIDI converter uses a version of the MML derived largly from “Classical MML” with some “Modern MML” features present. Some new specific commands are included also, and some commands are changed where necessary. This section outlines this version of the MML used by the program, and contrasts it to existing syntax where it significantly differs.

All commands in this language have their own line and are terminated by a new line (“\n”, “\r “or “\r\n”).

**Comments**

Comments are started with two hash characters at the beginning of a new line . This makes the remainder of the line a comment; any more hashes found on the line have no effect. Two hashes are used because single hash starts a meta command.

**Playing Notes**

The “play” command is used to play a series of notes and macros. Spaces can be intermingled with the notes to improve the clarity of the code. An example usage of this command is shown below:

play c5e5g5

**Note Syntax**

Notes are written as the note name followed optionally by the length of the note as a digit – each value for this digit represents a musical note length, which can be seen in the table below. If a length is not given, the default value is used, which is initially 5, but can be changed with the “l” command detailed shortly. A rest is represented by the note name “r”. To play an accidental note a “+” or “-”, respectively, is added after the note name and before the note length. Accidentals applied to rests are ignored.

|  |  |  |
| --- | --- | --- |
| MML Note Value Number | Musical Note | |
| American Notation | Name |
| 0 | 1/32 | Demisemiquaver |
| 1 | 1/16 | Semiquaver |
| 2 | 1/16 + 1/32 | Dotted semiquaver |
| 3 | 1/8 | Quaver |
| 4 | 1/8 + 1/16 | Dotted quaver |
| 5 | 1/4 | Crochet |
| 6 | 1/4 + 1/8 | Dotted crochet |
| 7 | 1/2 | Minim |
| 8 | 1/2 + 1/4 | Dotted minium |
| 9 | 1 | Semibreve |

To alter how each note is played, there are some of commands entered with the notes. These are listed below (where square brackets and their contents are not literal):

* o[digit] Set the octave each following note is played in. The digit represents the scientific pitch notation (SPN) number of the desired octave. All notes entered before this command is entered are played in the 4th SPN octave (“A” will be 440 Hz.)
* < Shift the octave down by one.
* > Shift the octave up by one.
* v[digit] Set the volume of the following notes. By default, notes will play at 100% volume.
* p[number from 0 to 11] Transpose all the following notes up by the number following ‘p’ semitones. The default setting is 0.
* l[digit] Set the default length of the following notes to the digit. The initial default length is 5. Note that this does not affect the ‘v’ or ‘o’ commands.

In modern MML there is also a “t” command, which sets the tempo. This is not included, as a more obvious command on it’s own line is favoured for ease of reading.

**Meta Commands**

These commands are entered on their own lines only once and are all preceded by a single hash. They tell the converter how the rest of the file should be played and add information to the MIDI file.

* #tempo [BPM] – set the tempo in BPM of the track (where a beat is a crochet.) This should be set the same in each MML track file when combining them into one MIDI file. The default tempo is 120 BPM.
* #instrument [general MIDI patch number] – set the instrument the rest of the file should be played with. The default instrument is a piano (GM patch number 0.) This command is not present in other MML versions because it is only useful if the file is being converted to a MIDI file.
* #name [name] – set the name of the track. This is put verbatim into the MIDI file in a track name meta event, and can be very useful when altering the MIDI file directly. Only one instance of this command should be in a MML file, otherwise a syntax error will occur.

**Macros**

A macro in this version of MML is written as below (on it’s own line):

$c v9o4c5

The dollar sign shows that this is a macro definition, and the letter following this is the “name” of the macro. The text after the dollar sign and letter replaces any other instance of the macro name found. A limitation of this notation is that there are only 26 possible macro names, but it is done this way to be more compatible with other versions of MML. Macros can be defined more than once.

**Full Example**

To conclude the section, a short example MML file is shown below.

|  |
| --- |
| ##Example comment  #name test\_track  #instrument 0  #tempo 120  $c l3o4cdefgab>c9  play v8$c |

**mmltomidi – Key Algorithms**

**mmltomidi – Main Data Structures**

**mmltomidi - User Interface**

**catmidi – Key Algorithms**

**catmidi – Main Data Structures**

**catmidi - User Interface**