Note Block Song (.nbs) documentation

The .nbs format (Note Block Song) was made mainly for the Minecraft Note Block Studio, and contains data about how note blocks are laid out in the program to form a song. You are free to create your own applications that handles these files.

Here's a C# VS 2010 project showing how to read .nbs files and log their contents.

All the data types found in a .nbs file are signed, and in little endian. Strings consist of a 32-bit integer, and then that many bytes for the characters. The file is made up of four parts, two of which are optional:

- #1: Header
- #2: Note blocks
- #3: Layers (Optional)
- #4: Custom instruments (Optional)

Part #1: Header

The header contains information about the file, all the data must be in the following order:

`Short: Song length`
The length of the song, measured in ticks. Divide this by the tempo to get the length of the song in seconds. The Note Block Studio doesn't really care about this value, the song size is calculated in the second part.

`Short: Song height`
The last layer with at least one note block in it, or the last layer that have had its name or volume changed.

`String: Song name`
The name of the song.

`String: Song author`
The author of the song.

`String: Original song author`
The original song author of the song.

`String: Song description`
The description of the song.

`Short: Tempo`
The tempo of the song multiplied by 100 (1225 instead of 12.25 for example). This is measured in ticks per second.

`Byte: Auto-saving`
Whether auto-saving has been enabled (0 or 1).

`Byte: Auto-saving duration`
The amount of minutes between each auto-save (if it has been enabled) (1-60).

`Byte: Time signature`
The time signature of the song. If this is 3, then the signature is 3/4. Default is 4. This value ranges from 2-8.

`Integer: Minutes spent`
The amount of minutes spent on the project.

`Integer: Left clicks`
The amount of times the user have left clicked.

`Integer: Right clicks`
The amount of times the user have right clicked.

`Integer: Blocks added`
The amount of times the user have added a block.

`Integer: Blocks removed`
The amount of times the user have removed a block.

`String: MIDI/Schematic file name`
If the song has been imported from a .mid or .schematic file, that file name is stored here (Only the name of the file, not the path).

Part #2: Note blocks

The next part contains the information about how the note blocks are placed, what instruments they have and what note. As you may or may not know, the song is divided into ticks (horizontally) and layers (vertically). Often, a majority of the ticks and layers in the song are empty, which is why we specify the amount of "jumps" to the next active tick or layer, rather than just a bunch of empty slots.

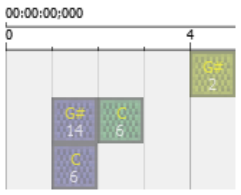
The pattern of the note block format is as follows:

Step #1: **`Short: Jumps to the next tick`**
The amount of "jumps" to the next tick with at least one note block in it. We start at tick -1. If the amount of jumps is 0, the program will stop reading and proceed to the next part.

Step #2: **`Short: Jumps to the next layer`**
Once we have found an active tick, we read the amount of vertical jumps to the next layer. We start at layer -1. If this is 0, we go back to Step #1. If not, we have found a note block!

Step #3: **`Byte: Note block instrument`**
The instrument of the note block. This is 0-9, or higher if the song uses custom instruments.
0=Piano (air), 1=Double Bass (wood), 2=Bass Drum (stone), 3=Snare Drum (sand), 4=Click (glass), 5=Guitar (wool), 6=Flute (Clay), 7=Bell (Block of Gold), 8=Chime (Packed Ice), 9=Xylophone (Bone Block)

Step #4: **`Byte: Note block key`**
The key of the note block, from 0-87, where 0 is A0 and 87 is C8. 33-57 is within the 2 octave limit. After reading this, we go back to Step #2.



For example, if the song looked like above, this is how this part of the file would look like:

```
__Tick -1__
2       Short           2 jumps to the first tick
__Tick 1__
2       Short           2 jumps down to the first note block in this tick

0       Byte            This note block have the Piano instrument
47      Byte            This note block have key #14 in Minecraft's 2 octaves (33+14)
1       Short           1 jump down to the next note block

0       Byte            Again, this have the Piano instrument
39      Byte            This note block have key #6 in Minecraft's 2 octaves (33+6)
0       Short           No more note blocks in this tick.

1       Short           1 jump to the next tick
__Tick 2__
2       Short           2 jumps down to the first note block in this tick

1       Byte            This note block have the Double Bass instrument
39      Byte            This note block have key #6 in Minecraft's 2 octaves (33+6)
0       Short           No more note blocks in this tick.
```

Here's some pseudocode for reading this part:

```
short tick = -1;
short jumps = 0;
while (true) {
    jumps = file.readShort();
    if (jumps == 0)
        break;
    tick += jumps;
    short layer = -1;
    while (true) {
        jumps = file.readShort();
        if (jumps == 0)
            break;
        layer += jumps;
        byte inst = file.readByte();
        byte key = file.readByte();
        addNoteBlock(tick, layer, inst, key);
    }
}
```

Part #3: Layers

This part is optional. You can choose to stop writing here and the Note Block Studio will still load the song. However, you cannot simply jump to the next part.

Anyways, here the information about the layers are stored, which in this case are the layer names and their volumes. These two values are repeated the same number of layers there are (the song height, specified at the beginning of the file):

`String: Layer name`
The name of the layer.

`Byte: Layer volume`
The volume of the layer (percentage). Ranges from 0-100.

Part #4: Custom instruments

Finally, the custom instruments of the song are stored. Like the previous part, this is optional. You can stop writing here and the song will still be loaded.

A song can have a maximum of 9 custom instruments, each with a name and sound file assigned to it. The sound file must be located in the /Sounds folder of the Minecraft Note Block Studio directory.

Before we begin, we need to know the amount of custom instruments:

`Byte: Custom instruments`
The amount of custom instruments (0-9)

The next four values are repeated the number of custom instruments:

`String: Instrument name`
The name of the instrument.

`String: Sound file`
The sound file of the instrument (just the filename, not the path).

`Byte: Sound pitch`
The pitch of the sound file. Just like the note blocks, this ranges from 0-87. Default is 45.

`Byte: Press key`
Whether the piano should automatically press keys with this instrument when the marker passes them (0 or 1).