Music Box User's Guide

Version 1.03.00 - 24/05/99 - 26/02/01
(C) Black Box <http://blackbox.resource.cx/> 2001

Table of Contents

Introduction to Music Box

Music Box is an advanced musicsystem, soundeffect system and music editor for the Nintendo GameBoy Color, Nintendo GameBoy, Nintendo GameBoy Pocket, Nintendo Super GameBoy, Nintendo Super GameBoy 2, Nintendo GameBoy EL Blacklight, Nintendo Wide Boy, Nintdendo GameBoy Advance and all the compatible console machines, software and hardware emulators. Hereafter referred as the GameBoy family.

Music Box Performance

It is balanced on the middle. Music Box isn't that simple audio system as most of the audio systems used in most of the games available on these systems, but also isn't the most superior one crunching out absolutely everything from the GameBoy family soundchip. (As used sometimes in demonstrations, but are useless in games, due to the heavy CPU needs.) We can do a less advanced system and a really superior system too, however a too simple system sounds bad and a really superior system needs too much CPU time and developers cannot make really advanced games with it. So we think it is a quite good (able to play great music with small CPU usage) system, great for games and demos, but if you think you need a more special one (including a GameBoy Advance system), you can hire us to make it for you.

Composing Outdoors, The Fun Factor

Music Box is a bit strange in a way. The Music Box 1 editor is also running on the destination system what isn't a common thing on the consoles. It is fun to compose music outdoors, even kids like it as a game (like Music 2000 from Codemasters on the PlayStation). It is a development tool, but this plus is interesting and only on the destination machine can a composer set and test absolutely 100% accurately the intruments, hearing exactly the same the end consumer will while playing the game.

Music Box 2

Some developers may find the Music Box 1 solution (small screen, limited controls) a bit uncomfortable, the already available Music Box 2 system is for them, what is perfectly working together with Music Box 1. The combinations are unlimited, you can use several MIDI sequencers, MOD or XM trackers, convert these to MB1 format, adjust the intruments in Music Box 1 and retouch the whole thing in Music Box 2. Or compose the music directly in Music Box 2 and test it on the real GameBoy Color connected directly to a PC by downloading the works runtime into the Music Box 2 GameBoy Color player. Music Box 2 with tons of extra tools and advanced (but backwards compatible) player is available now. It is even possible to compose 128 kilobits large musics with Music Box 2, what is twice the Music Box 1 limit. More instruments, more soundeffects, more samples, more advanced player and fileformat.

DMG Compatibility

As it was mentioned above, the Music Box family can play music on every Nintendo GameBoy machine, including the oldest greyscale DMG machines and up to GameBoy Advance. But not only the player and final product is compatible, the Music Box 1 editor is also running perfectly on all these machines, so even a really old second hand DMG machine is good enough to compose music on it.

Getting Started

This booklet is about the Music Box 1 tracker running on the Nintendo GameBoy and compatible console systems, so this section is covering only the setup of this tool. You can find the player setup in the Music Box Programmer's Guide also available in this package and the Music Box 2 setup in the Music Box 2 package.

Starting Music Box on GameBoy

The best way to execute Music Box is to execute it on the real machines. The most recommended machine is always the exact target machine, nowadays usually the GameBoy Color. Probably Super GameBoy, Super GameBoy 2, Wide Boy and GameBoy Advance sounds a little bit different, maybe even different DMG and GameBoy Color models. So test your music exactly on the target machine and even on several different models.

There are different technical methods to reach this goal, below we list the most common ones. (Note, this system is mostly for professional developers who are experienced with these things, however we get a lot of questions from amateur programmers, this section is for them, others can skip it.)

Music Box on Flash Cards

The most common method to execute GameBoy programs not available in shops on ROM cards are the Flash Cards. These are available from Nintendo for licensed developers, also available for developers and individuals from many (mostly asian) companies. Even can be done at home by people who are more familiar with the soldering iron. We get a tons of questions about these daily, anyway we cannot list here all the hardware and software solutions due to the reason this documentation is changing rarely and the manufacturers of these Flash Cards are changing names and quitting from business often (and new ones are also coming often with new products), also card specific and general Flash Card programmer softwares are getting written every day.

If you are a commercial developer you can buy Flash Cards from Nintendo together with the programmer hardware (the debugger can program too) and software. If you can't buy this, you can buy commercial Flash Cards. The most reliable, oldest and most common cards were manufactured by Bung, these are no longer manufactured, but at this very moment are still available at several UK and German resellers, probably at other sources too. Other cards are available from UFO, CCL and other companies, still in the business. We recommend you to visit the following links before buying Flash Cards. On the Subport <http://www.subport.org/> site you can find links to the official sites where you can buy cards, unbiased reviews of these cards, also all the programmer software. (Some cards are not compatible with Music Box, see below, and some cards are harder to use with a PC but better for Mac and other machine owners, for

example where you have to copy your GB ROM to a floppy disk and the programmer is flashing the card from that floppy disk.) For other reviews, tools, compatibility tests (which card can be combined with which programmer software and hardware) and a guide how to make cheap Flash Cards, EPROM Cards and programmers at home, visit Reiner Ziegler's site at <http://www.reinerziegler.de/>.

Music Box on Hardware Debugger

Music Box can work on the commercial developer hardware debugger too available from Nintendo (and Intelligent Systems) to registered developers. This is similiar to the Flash Card solution, however less portable, so not that recommended for Music Box as a Flash ROM Card. You can get the hardware, software and support from Nintendo or Intelligent Systems. This one also can be done at home by really advanced hardware hackers, but requires a lot of skills and does not worth the work in the case of Music Box. (We don't have links to such a schematics.)

Music Box with Software Emulators

Usually we really don't recommend software emulators for Music Box. Not the same as the real hardware and if you want to compose on the PC we recommend you Music Box 2. Anyway people without Music Box 2 and without real GameBoy machines, Flash Cards and other solutions, especially hobby programmers like this solution too. Until this time emulator programmers didn't programmed really good soundchip emulators into their GameBoy emulators, even if it is the most emulated machine, the audio in all the emulators was terrible. However there was one totally free software emulator what was OK on the audio side in our opinion, Rusty Wagner's Virtual GameBoy Color, what is no longer available online at the author's site: <http://www.graphicsstudio.net/> unfortunately.

Hardware Requirements

Music Box 1 doesn't has really high hardware requirements. It is working with all the GameBoy machines from the very first DMG to the next generation GameBoy Advance (in GameBoy Color mode). Working with most of the development tools and software and hardware emulators. It is using 32 kilobytes of ROM and 8 kilobytes of Battery Backed SRAM. Doesn't need any memory bank controller chip, so is compatible with all of them (MBC1, MBC2, MBC3, MBC5, HuC-1, HuC-3, plain EEPROM, Flash ROM, EPROM, ROM), however it may not be compatible with absolutely everything. There are even official Flash Cards without any SRAM and Battery Backed SRAM, what is needed, so if a card (from Nintendo, asian third party manufacturer or made at home) doesn't has 8 kilobytes of Battery Backed SRAM, that isn't compatible with Music Box 1. Same goes for a few non-standard memory bank controllers, for example Music Box 1 is absolutely not compatible with the Bung and Mr Flash PocketVoice card, because this card doesn't has any SRAM and its memory bank controller works on a really strange way (however we have a way to work around this problem, the other cannot be solved). Also some software emulators doesn't emulate the SRAM (or the soundchip at all), so some problems can come up with those too. Also be sure to save your works often to the PC and replace the old weak batteries when needed. (In every 1-2 years.)

Consult The Manuals

We are sure this line doesn't has too much meaning, anyway as we get a really lot of questions about how to start Music Box on real machine (as you can read above, the most recommended way is a Flash Card, also you can find links to shops, manufacturers, reviews and programmers), we also get a really lot of technical questions too, for example how to upload Music Box to a specific Flash Card. All the Flash Cards, Flash Card programmers and other solutions (including the schematics to the homemade cards) are available together with docs, please read those as everything works a bit different way, but everything is easy to use. If it doesn't help, check everything, especially the reviews on the mentioned sites and if it still doesn't solve the problem, get in touch with the manufacturers and programmers of the software. Only Music Box 1 and Music Box 2 is our creation, we can give answers about these, we don't even own a wide range of other solutions, so we don't know them.

Managing Music Files

After reading the previous sections, the manuals of your Flash Card or hardware debugger or other, it is probably clear for you how to manage the music files.

Music Box 1 music files are fixed 8 kilobytes (64 kilobits) large files, stored in the Battery Backed SRAM of the card (Flash Card or anything else). Only one is stored at a time (we could add more slots, but it is just a development tool, it is enough and this way it is compatible with more cards). You can read/write this file to/from a PC or other machine with your programmer hardware and software, so consult with the docs and makers of those if you have problems. Maybe there are some software and hardware which doesn't support the SRAM transfer, but we're sure there's always an alternative for the cards with SRAM, visit the recommended sites. The extension and name of this file doesn't matter, usually the programmer software is giving the .SAV extension for it and usually the ROM name, so MUSICBOX.SAV is the default, but you can rename it. Don't forget to backup often.

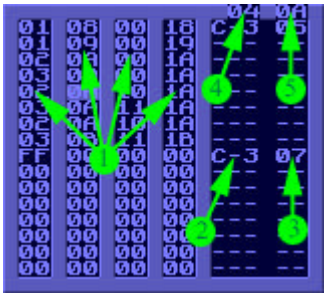
It is even easier in the cases of the software emulators which are supporting the Battery Backed SRAM. These are usually saving the ROMNAME.SAV file, so in this case MUSICBOX.SAV into a directory or their root directory. There's even less risk in these cases for some damage (no real SRAM, higher voltage programmer hardware, etc), anyway it is still recommended to backup this file to another directory and rename it often, to avoid some common problems. (A collegue is deleting the file or you liked the music better before you rewrote half of it.)

Using the Music Box Tracker

When you start Music Box on a GameBoy Color or compatible machine, you can see an introscreen like this in deep blue colors. When you start it on a DMG GameBoy machine, GameBoy Pocket or compatible greyscale machine, you can see the same with the usual yellow colors.

Pattern Editor

After skipping the intro screen with any button, the Pattern Editor is coming up with the last used patterns (or empty patterns at the first start if no music was loaded into the SRAM).



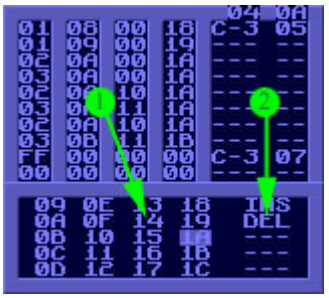
1. Pattern Order Editor
2. Note column
3. Instrument column
4. Actual row in actual window
5. Actual pattern

Control in Pattern Editor

If you are familiar with the tracker programs (and if you are a composer you surely saw at least Protracker and Fasttracker II) you can find out easily how the Pattern Order Editor and Pattern Editor is working from the picture above. There are four pattern columns in the Pattern Order Editor, one for every channel, you have 64 rows ($00-$3f) to use the 56 patterns ($00-$37, $ff in the first column is the restart command, $fe is the stop). In the Pattern Editor you have the usual 64 rows too ($00-$3f), you can use notes between C-3 and B-8 and 32 instruments (note: between $01-$20, $00 is the empty line). You can move the cursor with the **direction pad** both in the Pattern Order Editor and Pattern Editor (and between the Note and Instrument columns too), and you can switch between the two windows with the **A** button. You can see your actual row in the upper right corner, together with the active pattern. **B** button is the input button everywhere, **select** is switching to the Instrument Editor and back, **start** is switching to the Player Window and back.

Numeric Input

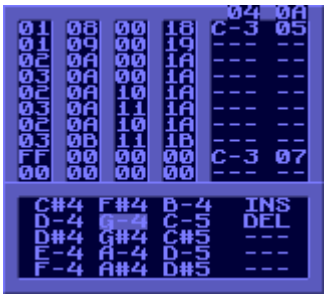
When you press the **B** button in the Pattern Order Editor or in the Pattern Editor's Instrument column, the Numeric Input window is coming up. You can move between the selections with the **direction pad**, up/down is moving slower by one position, left/right is faster by one column, you can enter your selection with the **B** button or cancel it with **A**. With **select** you can move to the Insert/Delete Menu where you can insert or delete a row with the **A** button or get back to the Numeric Input by pressing **A** on an empty menu.



1. Numeric Input
2. Insert/Delete Menu

Note Input

B button in the Pattern Editor's note column will bring up the Note Input window. This one is working totally the same way as the Numeric Input window, only with the valid notes and empty line.



Note Input

Instrument Editor

The **select** button is switching between the Pattern Editor and Instrument Editor. **B** button is the input as usual and **A** button is cycling between the ABC Editor, Sample Editor, Arpeggio Editor and 3 Bytes Editor.

1. ABC Editor
2. Sample Editor (channel 3)
3. Arpeggio Editor
4. 3 Bytes Editor
5. Instrument row
6. Sample/Arpeggio row

ABC Editor

This is the most complicated part of Music Box. Every row ($01-$20) is one instrument and every two digit is forming one byte. The first one is A and the fifth one is E. There are no rules which instrument can be played on which channel, but as every channel is working totally differently on the GameBoy, every musician has to set his/her rules while filling this table. For example if he/she wants to use a lot of instruments on channel 1, he/she can think to the $01-$10 instruments as channel 1 instruments. (And use these instruments in the patterns will be played in the first column of the pattern order.) Every instrument can be used on every channel technically, but the composer has to setup every instrument totally different way, depending on which channel he/she want to play it. We hope this is clear enough and these open rules are better than some preprogrammed ones. It is wise to divide the available instruments to smaller packs like $01-08 = channel 1, $09-$10 = channel 2 and so on, easier to remember, but totally open to the composer. (Note: if hexadecimal numbers are new for you, you really have to read the FAQ now.)

You can use the following tables the following way. At every byte from A to E you have several choises from several lists of options. Choose one from every list (for example one slide speed, one slide direction and one slide step) and add them up (easier than counting with binary

numbers, even beginners can use it) with a calculator can work with hexadecimal numbers (you sure have one built into your operation system). When you have finished with all the lists below a letter, enter your hexadecimal number to the place of your letter. Number calculated for A to the first column, number calculated for B to the second one, etc. The only problems can be for the people doesn't know the hexadecimal numbers where we didn't listed all the options only the first few ones and the last, but with basic hexadecimal knowledge and a little thinking (and with the aid of the FAQ), even those can count the hexadecimal slope and guess the missing numbers and options.

For the programmers and more technical composers we listed the official register names and register addresses, they can look up in the developer docs the exact meaning of each bit and fill the bytes that way (like the ms exact timing of the slide and exact formula, etc).

Note also that channel 1 and 2 are quite similiar channels, pulse waveform modulators. Channel 3 is digital sample channel (4 bit one with 16 bytes long wavetable) and channel 4 is a noise generator mostly can be used for percussion instruments. There's no pitch for the channel 4, so in the patterns will be played on channel 4 the notes doesn't count anything.

Channel 1 ABC Bytes

- Byte A - Frequency Slide - register NR10 ($ff10)
  - Speed of slide (choose one):
    - $00 - No slide
    - $10 - Sweep very swiftly
    - $20 - Sweep swiftly
    - ...
    - $70 - Sweep slowly
  - Direction of slide (choose one):
    - $00 - Slide up
    - $08 - Slide down
  - Step of slide - note that too little steps may result in no slide (choose one):
    - $00 - No slide
    - $01 - Biggest steps
    - $02 - A bit smaller steps
    - ...
    - $07 - Smallest steps
- Byte B - Waveform/Sound Length - register NR11 ($ff11)
  - Waveform (choose one):
    - $00 - 12.5% (- _ _ _ _ _ _ _)

- $40 - 25.0% (- - _ _ _ _ _ _)
- $80 - 50.0% (- - - - _ _ _ _)
- $c0 - 75.0% (- - - - - - _ _)

➢ Sound Length - after this time the sound will be muted, see also Byte D (choose one):

- $00 - Shortest
- $01 - A little bit longer
- $02 - Two little bits longer
- ...
- $1f - Longest

❖ Byte C - Volume Envelope - register NR12 ($ff12)

➢ Initial volume (choose one):

- $00 - Silence
- $10 - Very, very quiet
- $20 - Very quiet
- ...
- $f0 – Loudest

➢ Direction of volume change (choose one):

- $00 - Fade away
- $08 – Amplify

➢ Speed of change (choose one):

- $00 - No change
- $01 - Change very swiftly
- $02 - Change swiftly
- ...
- $07 - Change slowly

❖ Byte D - Sound Length Mode - counter/consecutive selection bit (bit 6 of NR14 ($ff14))

➢ $00 - Ignore the length in Byte B, note will sound till the next note

➢ $40 - The note sounds as long as specified in Byte B

❖ Byte E - Arpeggio Pointer

Channel 2 ABC Bytes

100% same as Channel 1 ABC Bytes out of that Byte A is unused. So this is more or less

compatible with channel 1. Channel 2 instuments can be played back on channel 1 and the result will sound the same if Byte A is $00 for channel 2. And channel 1 instruments also can be played back on channel 2, if Byte A is $00 for channel 1, the result will sound the same, if it is different, still it will sound similiar and possibly good.

Channel 3 ABC Bytes

- Byte A - Waveform Pointer
  - Points to the waveform table. $00 = offset $00, $01 = offset $10, $0e is the last value. $ff is the channel 3 mute command.
- Byte B - Sound Length - register NR31 ($ff1b)
  - The length of the sound. All values ($00-$ff) are legal. The bigger the value the shorter the sound. See also Byte D.
- Byte C - Sample Volume - register NR31 ($ff1c)
  - $00 - Mute
  - $20 - Full volume
  - $40 - Half volume
  - $60 - Quarter volume
- Byte D - Sound Length Mode - counter/consecutive selection bit (bit 6 of NR34 ($ff1e))
  - $00 - Ignore the length in Byte B, note will sound till the next note
  - $40 - The note sounds as long as specified in Byte B
- Byte E - Arpeggio Pointer

Channel 4 ABC Bytes

- Byte A - Noise Mode - register NR43 ($ff22)
  - Pitch of noise (choose one):
    - $00 - Highest
    - $10 - Very, very high
    - $20 - Very high
    - ...
    - $d0 - Lowest
    - $e0 & $f0 - Illegal values
  - Noise mode (choose one):
    - $00 - Noise 1
    - $08 - Noise 2

- Another pitch of noise (choose one):
  - $00 - Highest
  - $01 - Very, very high
  - $02 - Very high
  - ...
  - $0f – Lowest

- Byte B - Sound Length - register NR41 ($ff20)
  - The length of the sound. Values $00-$3f are legal. The bigger the value the shorter the sound. See also Byte D.

- Byte C - Volume Envelope - register NR42 ($ff21)
  - Initial volume (choose one):
    - $00 - Silence
    - $10 - Very, very quiet
    - $20 - Very quiet
    - ...
    - $f0 – Loudest
  - Direction of volume change (choose one):
    - $00 - Fade away
    - $08 – Amplify
  - Speed of change (choose one):
    - $00 - No change
    - $01 - Change very swiftly
    - $02 - Change swiftly
    - ...
    - $07 - Change slowly

- Byte D - Sound Length Mode - counter/consecutive selection bit (bit 6 of NR44 ($ff23))
  - $00 - Ignore the length in Byte B, note will sound till the next note
  - $40 - The note sounds as long as specified in Byte B

- Byte E - Unused

Waveform Editor

The waveform table contains 240 bytes from $00 to $ef, 4 bit digital samples. These can be edited the usual way with the **B** button. 15 times 16 bytes are stored here (the size of the Wave RAM) and the selected one (Byte A of Channel 3 ABC Table, $00 = $00-$0f, $01 = $10 - $1f … $0e = $e0-$ef, $ff = mute) is uploaded to the Wave RAM at channel 3 instrument change.

Arpeggio Editor (Frequency Manipulator Sequencer)

The Arpeggio Editor is more than how it sounds. It is a programmable Frequency Manipulator Sequencer, out of the Arpeggio, Portameno, Vibrato and other frequency change effects can be programmed here.

This window can be edited the same way as the Waveform Editor. The Arpeggio Pointers in Byte E of Channel 1, 2 and 3 ABC Tables are pointing to direct hexadecimal bytes in this table, from $00 to $ff. This is a really open sequencer, you can point to anywhere into it from any instrument, share Arpeggios or just Arpeggio parts between instruments and channels.

There are four commands in the Arpeggio Table and the pointer is making one step for all the three channels at every player call. The actual value in the table will be added to the played base note as halfnote, so you have to watch out to not to create false sounding music with wrong Arpeggios, Portamentos and Vibratos. This addition command is between $01 and $7f. It is possible to subtract halfnotes the same way with the commands between $81 and $ff ($ff = subtract one halfnote). $00 is the nop (no operation) command can be used for delay, timing or just for empty Apeggio sequences. $80 is the jump command, this is always followed by another byte, but that isn't a transpose, but the destination of the jump in the Arpeggio table.

If it is not clear the following example is explaining it, we start with C-5 base note and the instrument's Apreggio pointer is $05. On the left column we can see the offset and value in the Arpeggio Table, on the right the currently played note.

```
Offset $05: $00 – C-5 (base note)
Offset $06: $04 – E-5 (transpose base by 4 halfnotes)
Offset $07: $07 – G-5 (transpose base by 7 halfnotes)
Offset $08: $80 – jump command, take the destination from the next byte
Offset $09: $05 – jump to Offset $05, take $00 from there, play C-5 again
```

Also note that when you insert and delete in the Arpeggio Editor, the Arpeggio Pointers are getting fixed automatically.

3 Bytes Editor

The 3 bytes on the bottom are the main control bytes for the music.

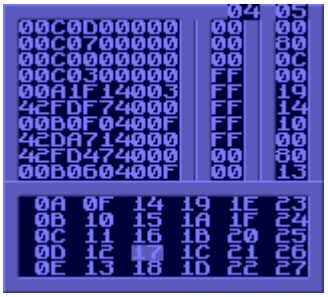
The first one is the tempo, it is dividing the maximum BPM. The higher the value, the slower the music (multiply this for double and quattro replay speed).

The second one (NR50 or $ff24 register) is the left/right global volume controller. The first digit is controlling the left volume and the second is controlling the right volume. Volume can be controlled only between $0 and $7, so $00 is silence, $77 is max volume, $73 is a heavy left balance.

The third byte (NR51 or $ff25 register) is controlling the output for every channel. The channels (channel 1, 2, 3 and 4) can be played on the left, right, both or none of the channel outputs. From the table below add the chosen outputs and you get your value to set here.

- $80 - Channel 4 is output to the left
- $40 - Channel 3 is output to the left
- $20 - Channel 2 is output to the left
- $10 - Channel 1 is output to the left
- $08 - Channel 4 is output to the right
- $04 - Channel 3 is output to the right
- $02 - Channel 2 is output to the right
- $01 - Channel 1 is output to the right

Numeric Input Light



Numeric Input

Not too much can be told about this, same as Numeric Input, just without Insert/Delete Menu, what is not needed in the windows of the Instrument Editor, out of the Arpeggio Editor, where the previous Numeric Input is coming up.

Player Window

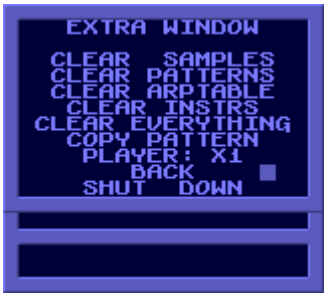
The Player Window can be accessed from the major windows with the **start** button.

1. Starting pattern position
2. Restarting (loop to) pattern position
3. Actual pattern position
4. Actual track position
5. Actual pattern numbers
6. A small bonus scroller, credits

The picture and the description above can be easily understood. There are only a few commands on this screen. With the **direction pad** the starting pattern position can be changed with its left/right directions and the restarting pattern position (where the pattern sequencer will jump to after the $ff command in the first column of the pattern order) with its up/down directions. The **start** button is leaving this screen again and the **select** button will bring up the Extra Window. The **A** button will start the music and while playing, the **B** button is fast forward, everything else is stop.

Extra Window

This can be reached only from the Player Window by pressing the **select** button.

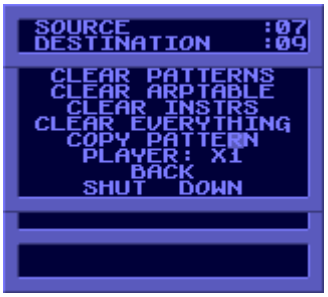


Clear samples (wavetable)
Clear patterns (plus pattern orders)
Clear arpeggio, vibrato, portamento table
Clear instruments
Clear everything (all above)
Copy pattern
Player type (1X, 2X, 4X speed)
Shut down

The extra functions are listed on this window. Most of them are speaking for themselves. The samples, patterns (including pattern orders), arpeggio table and instruments or all of them can be cleared here. There's no undo so watch out what are you doing. **B** is the activation button here. Back will go back to the previous menu.

The Player type is an interesting point. The player can be speeded up to twice and four times its speed, called two and four times per screen refresh instead the usual one. It can be used to gain some more tuned instruments and higher speed, useful for techno music, also more accurate timing can be reached by this. Unfortunately this speed isn't stored in the music (at least not in Music Box 1), so if the speed isn't 1X, you have to note it somewhere and always set the right speed here when you start Music Box by cycling between the three possibilities with the **B** button.

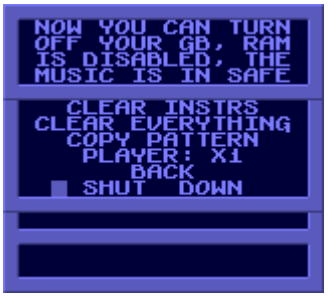
Copy Pattern



Source pattern number
Destination pattern number

This is the Copy Pattern screen, speaks for itself. The **direction pad** can be used to change the values, use left/right to change the source pattern, up/down to change the destination pattern, **B** to copy, **A** to cancel.

Shut Down Window



Shut down window

When you choose this option, the machine will stop. This doesn't sounds really interesting and useful and maybe it isn't that. This is the quit option, something like in Windows and Linux and other modern operation systems and is here for the same reasons, not for fun.

The music is always stored in the Battery Backed SRAM and data can get damaged there, especially when the SRAM isn't disabled when somebody is turning off the machine. It doesn't happen that often (it only happened once for us yet), anyway this problem is well documented and there's a possibility to avoid it (to disable the SRAM access). So this is the absolutely safest way to quit. First go here and choose this menu and only after this turn off the machine. And backup your SRAM often, your many hours of work is in risk. Of course if you just play a few notes and test music, you can turn off the machine without this, SRAM doesn't get corrupt every day.

Frequently Asked Questions

Q: *Can I send donations?*

A: Yes, of course. This is what keeping us doing and releasing such a free tools (at least free for non-commercial usage) like Music Box 1. Contact us if you like our tools and would support us with some bucks or some hardware or anything.

Q: *Will a PC tracker be available in the future?*

A: We've got this question often and it was the reason for the development of Music Box 2. What can be perfectly used together with Music Box 1 and its tools and what is backwards compatible, but based around a PC tracker with many PC tools. Music Box 2 is already available, however not to the public, only for customers. Contact us if you are interested in it.

Q: *Can I buy Music Box 1 on a card?*

A: Unfortunately Music Box 1 is not a licensed GameBoy program what will be available on the shelves of the shops. (That would be great for you and us too.) Nor we don't have the possibility to manufacture cards in small quantity. So if you don't have any development tools (nor official or unofficial ones), the best solution to run Music Box 1 on a real machine is a mentioned mass manufactured Flash Card. These aren't available in our country at all and we don't have the budget to buy a lot to resell them, so it isn't an option to buy such a cards from us with Music Box 1 flashed into it already. (However in this case programmer tools weren't important for you to buy, anyway those are needed to save, backup, transfer and reload the music files from and to PC too, so we recommend them.) Our best advice is to buy a Flash Card like this yourself with a programmer and flash Music Box 1 into it (or anything else, even we have some free legal GameBoy programs on our site and you can reflash the card as many times as you want) and send some donation to us.

Q: *What's wrong with the emulators? Can I use them to compose music? I have no other choises.*

A: Well, GameBoy is the most frequently emulated system, anyway even if it has a simple soundchip, almost no emulator emulated the soundchip good enough. There are many inaccurate emulators with wrong timing (too fast/too slow, so music cannot be composed on it), many without any sound at all and many with weak sound (missing channels, simple AdLib emulation, etc). We can only name one emulator with proper soundchip emulation, Rusty Wagner's Virtual GameBoy Color, what was free, but isn't available on the web anymore from the author's site: <http://www.graphicsstudio.net/> unfortunately.

Q: *How can I use the files with .1X, .2X and .4X extensions?*

A: These are simple .SAV files, so are working like those, simple binary SRAM images without any header, can be stored in a card's SRAM through a programmer hardware and software or can be used with an emulator. Sometimes you have to rename them manually to .SAV, some tools are recognising these files only with this extension. As playing speed (single, double, quattro) isn't stored in MB1 format, we sign the playing speeds of the songs this way sometimes (or in additional text files). The .1X files are singleplayer files, for .2X and .4X you have to set the speed to double and quattro player in the Extra Window every time you start Music Box.

Q: *How can I transfer the .SAV, .1X, .2X and .4X files into a card's SRAM?*

A: This is a really common question. The answer is partly in the previous answer and in the "Getting Started" section, especially in its "Managing Music Files" section. Also read the manuals of your programmer hardware, software and Flash Card or other development tool, these questions are surely answered there too, or get in touch with the manufacturers and programmers of those.

Q: *How can I save a music and link into my sourcecode, game, demo, etc?*

A: You can see in the previous answers and elsewhere in this booklet that Music Box music is stored in SRAM on the card and can be transferred to a PC with the Flash ROM programmer hardware and software or hardware debugger or other development tools. On the PC these are stored as 8 kilobytes large binary SRAM image files without any headers, with .SAV, .1X, .2X or .4X extensions. And also can be transferred back to the SRAM the same way. These files can simply be included into a source file as binary data and can be used with the player. More details are available in the examples and the Music Box Programmer's Guide.

Q: *Can I get the player and/or editor source?*

A: Sources of these tools are not available to the public. However for a high enough bid we may consider about selling these.

Q: *Where can I download Music Box 2?*

A: Music Box 2 is not freely available, not even to hobby programmers and musicians like Music Box 1, not even as a crippled shareware version. Contact us if you want to purchase it and if you can't afford it, you still can use Music Box 1 freely in noncommercial projects.

Q: *Can I be a betatester of the Music Box productline or other Black Box products?*

A: We are usually working with a smaller betatester team of ours, usually we aren't looking for betatesters and aren't releasing public beta versions. Anyway check out our site often at <http://blackbox.resource.cx/>, things can always change and infos about possible future beta projects and positions will be listed there.

Q: *Your site is really slow for me, I don't have hours to download your pages and files, what's wrong?*

A: Our site is stored on a smaller server in Europe. It is fast for us, but reported to be slow from other countries. Especially when it is getting many hits at the same time, especially from countries far away from here, like Australia, Japan, USA, etc. If you send us donations, we can improve it in the future, or try to access our site a bit later. Also the downloads from our site are mostly available at the Subport site too <http://www.subport.org/>.

Q: *Will be a better control available in the future versions?*

A: We know the current control is a bit hard to use (and twice harder under emulators), but the GameBoy controller is really limited. Still there are a few alternative possibilities what we could use, for example inputing each digit (in the notes and numbers) seperately from a keyboard image, or from PC through serial port, or from an infrared keyboard, or even from remote controllers or a phone like image, or with morse code and many more possibilities. But most of them would be available for limited people and all of them would be slower than the current one. Usually you have to repeat the previous number or note, or input a close one (what needs only a few keypresses with the current method, no more), so even if it is strange at first, still the fastest solution for Music Box in our opinion. The short answer is we will not change it as we don't plan major modifications on Music Box 1.

Q: *M2MB and/or XM2MB tool doesn't convert well, what can I do?*

A: These are third party tools, weren't done and aren't supported by Black Box. If you have problems with those or questions about them, read their documentations first (there are many limitations in both) and contact the authors of these tools with questions and suggestions. By the way, XM2MB never worked for us, so probably won't work for you either.

Q: *Can I convert NES/SID/AdLib/etc music?*

A: These machines/soundchips don't have one specific music format. There are several thousands of different music formats on Commodore 64 (SID soundchip) and probably hundreds on Nintendo Entertainment System and PC (even if we count only the AdLib specific ones). As there's no direct format, there's no possibility for these. However there are several tools available to convert many NES/SID/AdLib songs to MIDI/MOD/XM and MIDI can also be converted to MOD and XM, so with many conversion steps and third party tools there is a little chance. Anyway we don't know such a converters, so we can't help, use websearchers and experiment with the converters. Also Music Box is mostly a developer tool for programmers and musicians, there's not too much reason to convert when you can create original work.

Q: *Will be there a possibility to change the default colours?*

A: The actual colours are perfectly visible, so we can't see any reason to add a colour changer menu into the tracker. This is not a mass produced program available in shops, in that case there would be a sense to add such a things and other customizations, but this is just a developer tool for musicians and composers, not the tool itself with its outfit is important, but the music people are creating with it. Anyway for the customers and people sending higher donations we will change the colours in personal versions if they really need different colours.

Q: *Will it be possible to store more music files in the SRAM?*

A: This is a good question, unfortunately our answer is no. We know it would be a great option, but it is missing for two reasons and we don't plan to add major things to Music Box 1 in the future so will not be added. The first reason is simple, mostly we forgot about such an options. And the second reason is that this way Music Box 1 is compatible with more cards, every card with at least 8 kilobytes of SRAM. Larger SRAM requirement would limit the possible cards heavily. So backup your SRAM to a PC often, it is a bit annoying, but still a really quick process, takes around 20-30 seconds.

Q: *What is hexadecimal and binary, what the $ sign means at numbers?*

A: These are not really Music Box 1, Music Box 2 or GameBoy related, but we get too many questions about these. First, check out a programmer book or tutorial for beginners (also tons of these are available on the web) and you surely can find these in it. Or ask for help in personal from a friend, almost everybody knows these, we are sure nowadays it is taught in the elementary schools. Anyway we try to descibe these in short. Usually we are counting with decimal numbers, we have 10 numbers from 0 to 9 at each digit (position) and when we turn the counter in a digit (reaching the maximum), we increase the next number on the left (and unlimited numbers can be on the left). This is what everybody knows who can read. Binary works the same way, but the possible numbers at each digit are 0 and 1 only. And hexadecimal also works the same way, but it has 16 possible numbers, we can squeeze these into one digit by using the decimal numbers and after 9 we use the A, B, C, D, E and F letters. The sign of hexadecimal is the $ prefix in this documentation and the sign of binary is the % prefix. (In some programming languages other signs are also used.) You don't have to think a lot when you want to change a number between binary, decimal and hexadecimal, you sure have a built in calculator can do it in your operation system and in your filemanager, also many of these calculator programs can be downloaded from the net freely. Anyway everybody can figure out easily how to change these manually, here is an example. These are the same numbers: decimal: 223, hexadecimal: $df, binary: %11011111. Music Box is using only hexadecimal numbers.

Tips

The following quick tips can be really useful in the first steps with Music Box 1 and Music Box 2.

Of course first, always backup your files, especially from the SRAM. And in the case of multispeed musics (doubleplayer and quattroplayer) it is wise to note the playback speed somewhere, for example in the extension, .1X, .2X and .4X. If the speed is not single speed (the default) it have to be changed in the Extra Window every time the tracker is restarted.

Always organize your instruments between the four channels. You can setup some rules for yourself what you will use with Music Box 1 always (and you can remember to these later), or setup rules for every song. Rules like instrument $01-$08 goes for channel 1, instrument $09-$10 goes for channel 2 ... etc. But you can give less instruments for percussion (mostly channel 4) if you are using less percussion instruments usually and you even can share many instruments between channel 1 and channel 2 as these are really similiar, however channel 1 is a bit more advanced. These are all up to you.

It is always good to keep a null instrument with $00 filled values. You can use this always on all channels as note off or you can mute some looping soundeffects in games too with this method. It is wise to keep it always on the first ($01) instrument, but depends on you.

A null Arpeggio is also useful, what can be used with the instruments where no Arpeggio is needed. We mostly recommend to place this to the beginning of the Arpeggio Table, in this case only the following three bytes are needed: $00, $80, $00. (Play the base note and a jump command to the $00 position of the Arpeggio Table.) In this case the default $00 Arpeggio pointer will point to this one from the instrument desciption bytes (ABC Bytes).

How to Buy Music Box

Music Box 1 can be freely used in any non-commercial projects, including mp3 music, live performance and free GameBoy demos and games. (In these cases all we need is your free work, the credit for us and a donation if you can afford and you think we deserve for our work.)

It also can be licensed into commercial projects (games, demos and tools), now available only together with Music Box 2. (So the package is including two trackers for one price, one of them is running on the GameBoy, one on PC, with many tools, converters, examples, etc. Also with full and fast customer support and help. We answer every questions, make more examples and help out with every problems on request. We even make minor modifications and customizations to the system if some customers need it.) For details, contact us.

Another licensing possibility into commercial projects (games, demos and tools) is, to license only the musicplayer for a lower price and hire us to make all the audio works in the game including musics and soundeffects. Also contact us for details.

Copyright

Thanks to Ádám Ábrahám, Ákos Balázs, David S. Berkompas, Mr EMP, Collin van Ginkel, Jukka-Pekka Luukkonen, Balázs Oszvald, Martijn Reuvers, Attila Szõke, Péter Tihanyi and all the forgotten people for their help.

Ray Nemes and Zsolt Minier, <blackbox@resource.cx>