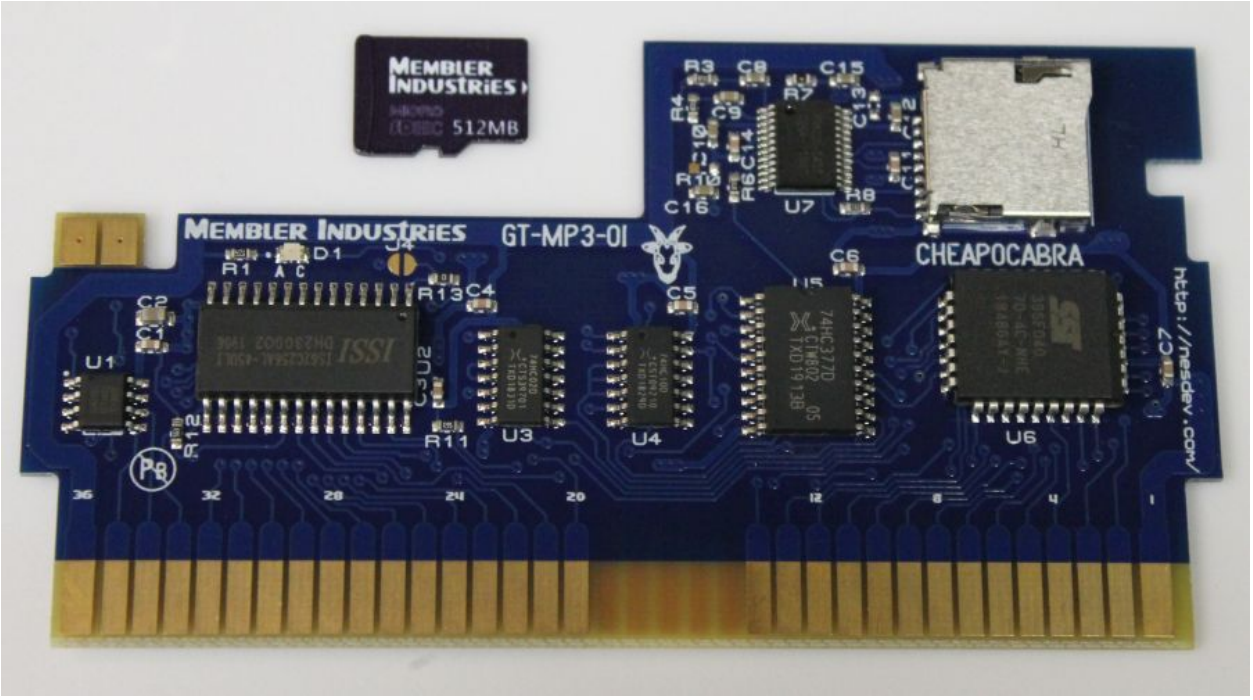
GTMP3 NES Flash Cartridge



Features

- Mapper-controlled MP3 player audio expansion
- 512MByte microSD card included, up to 32GByte supported
- 512kByte Program FlashROM, 32kB page size
- 16kByte Character Pattern RAM, 8kB page size
- 8kByte Nametable RAM, 4kB page size
- 4-screen nametable memory standard, 2-screen H/V jumper optional
- Non-volatile save data supported with FlashROM
- Inexpensive discrete logic mapper using 74HC10, 74HC02, and 74HC377
- mapper-controlled indicator LED
- Includes multi-region CIC by krikzz
- edge connector ENEPIG plating, gold and palladium, over nickel
- optional: jumpers for 32kB Pattern Table RAM, using NES internal RAM for nametables
- optional: compatibility with defective NES clones

GTMP3 is a low-end NES Flash cartridge, compatible with GTROM, that includes an MP3 player DSP as coprocessor. Price is kept low as possible to support independent developers and publishers.

MP3 Player

The board includes an inexpensive coprocessor that plays MP3 and WAV files from a microSD memory card. The NES can write to it, but with this mapper is unable to read anything back from it. To reduce costs, there is no hardware UART, but this communication can be handled by the NES CPU with timed code. It is done by writing commands to a single bit, D7 of the mapper register. Commands are 8 bytes long, and are transmitted at 9600 baud.

To send a command to the chip, the NES must disable NMI and any possible IRQs, because the process must not be interrupted. In screen-display terms, this takes a significant amount of time, about 132 scanlines worth of CPU time (for NTSC). After a command is sent, the chip will play the music with no further intervention from the NES CPU. However, it is required that the NES program always write '1' to D7 of the mapper register (ORA #$80 instruction before mapper write is one suggestion). Writing a zero to it at any time is likely to corrupt the next transmission to the chip.

File System

The MP3 player ignore filenames, except for digits at the beginning. So if you have "Bloody Tears.mp3", you must rename it to "001Bloody Tears.mp3", or "0001Bloody Tears.mp3" to allow the player to index it as track number 1. Track zero and folder zero should be omitted, as they will not be accepted by the MP3 player. The folder names are not case sensitive. If you write multiple tracks beginning with the same number, the player will only choose the one that was first written to the microSD.

There are multiple types of folders, depending on the command you use to select the file.

- Up to 99 folders named "01" through "99", up to 255 tracks per folder
- Folder named "MP3", up to 65535 tracks in one folder
- Folder named "ADVERT", up to 65535 tracks in one folder
- 15 folders named "01" through "15", up to 1000 tracks per folder (not currently supported in the library)

The provided software library allows use of most of the commands supported by the chip, however some of them may be redundant, depending on how you wish to control it. Here is a list of the most useful commands:

Select MP3 Folder

The folder named “mp3” may contain up to 65535 tracks, and this command selects among tracks within that folder. 16-bit value is passed to the command with the MSB in the X register, LSB in the Y register.

```
lda #CMD_SELECT_MP3_FOLDER
ldx #>5278
ldy #<5278
jsr mp3_command ; select track #5278
```

When a track number is selected, the device must seek through all files in the folder until it finds it. The chip manufacturer recommends not using more than 3000 files in this folder, to maintain a reasonable seek time when changing tracks.

Select File

With this command, you are allowed to select from up to 99 folders, with up to 255 tracks per folder. Load X with the folder number, and Y with the track number.

```
lda #CMD_SELECT_FILE
ldx #1
ldx #4
jsr mp3_command ; select file “004___.mp3” from folder “01”
```

Select Volume

Scales the output volume. Default is maximum, the valid range is 0-31. Load volume value into the X register. **WARNING:** This chip will happily allow you to set invalid values beyond 31, but you should not do this! The output seems to wrap, so at best you will hear annoying crackling sounds, at worst it potentially could cause audio amplifier damage. If volume 31 is not enough, you must amplify your source audio before encoding as an MP3.

Pause

Pause playback.

Play

Play is the default state when the cartridge is powered on.

Advertisement

This command allows you to interrupt the currently playing track, with another track which must be selected from the “advert” folder. When this track finishes, the previously playing track will be resumed where it left off. This allows up to 65536 tracks can be selected. The 16-bit value is passed to the command with the MSB in the X register, LSB in the Y register.

When powered up, the chip will auto-detect and open the microSD card. Be aware that this chip on power-up likes to randomly select a track, and begin playback in looped mode.

The chip takes about 200ms (about 12 frames on NTSC) to initialize, when first opening the file system of the microSD card. Do not send any commands to the chip during this time. Note that the NES console has its own reset generator for the NES CPU, the timing of which may vary by console and by ambient temperature. Be sure that the NES waits long enough. A command sent too early will be ignored.

This particular MP3 chip seems to not have any kind of built-in filters to prevent hard DC voltage jumps that cause loud pops or thumps, so take appropriate precautions if you’re powering the NES up, or resetting the MP3 player while it’s connected to a live external amplifier. □ **Please be extremely careful when using headphones!** □ However, though the reset pop is annoying, this complete lack of filtering may be to our benefit. It will makes no attempt to smooth out the hard sharp edges and full-blast staccato that is commonly found in chipmusic.

Power-up, Reset, and Timing Considerations

There are only 2 ways the NES can reset the MP3 chip. First, by powering on the NES. Secondly, by sending it a reset command. This means that if the user holds the reset button, the music will continue playing. If the NES connector is dirty, the game might not even start but the MP3 chip may autoplay a track on its own. The lockout chip might be resetting the NES, but the MP3 chip will continue with the last command it was given, if any.

One particular area of concern with this is the console’s reset button. You might choose to have separate cold and warm boot reset functions, if needed, to handle initialization in these differing cases.

The MP3 player has its own integrated oscillator, and it can not be synchronized with the NES system clock. Tolerance details are not in the specifications, but it is most likely tuned at the factory to oscillate at nominal frequency at normal room temperature, and might be predicted to vary as much as ±1% if operated at extreme ambient

temperatures. The ramifications of this, unfortunately, is that it isn't possible to synchronize MP3 playback with music playback in the NES channels for any significant amount of time. If this is the approach you wanted, a suggested alternative would be to premix NES and expansion audio together in the MP3 or WAV track.

Errata

The "chip reset" command will cause an audible pop or thump. Recommended to use only if necessary.

The randomizer in the "shuffle play" command seems to begin uninitialized, as it will always begin at the same track. Once playing, using the "next/prev track" command will then select a random track.

The "mp3" folder supports a larger number of files, but this comes at the cost of additional time for the chip to search the file system. More files in the folder will increase the search time, the manufacturer recommends keeping it under 3000 files.

After sending a track change command, the chip will not be ready to receive any more commands until it is finished finding the track. As this mapper provides no way for the NES to read status info from the chip, you will need to insert a delay in your software before sending additional commands.

Using the Mapper

The mapper control has 5 separate functions combined into one writable register. Unlike most discrete logic mappers which overlap the ROM and may cause bus conflicts, writing to this mapper requires no special precautions. Because the register is not readable, it is recommended you keep a shadow copy of the register in RAM. See application note (TBD) for code examples.

CPU Page Select lets you select a 32kB page of ROM to map into CPU memory. This is in an undetermined state during power-up and reset, see application note (TBD) for a code library supporting various boot options. It is also possible to use a 16kB page size, with the other 16kB (upper or lower) being a fixed bank, and the fixed bank must be

specified at the time the cartridge is programmed. When operating in 16kB page mode, due to interleaving, the available memory is reduced by half, from 512kB to 256kB.

PPU Pattern Select chooses one of the two 8kB pages of CHR-RAM that are available for sprite object and background tiles. The second set may be used for displaying more background tiles using a split-screen effect, or for double-buffering larger updates to CHR-RAM.

PPU Nametable Select allow you to use 2 different sets of 4 nametables each. When doing 8-way scrolling with a split screen, the two sets can be used for the separate areas of the screen. This makes updating the nametables much easier while scrolling. This cartridge disables the NES's internal 2kB VRAM and provides 4kB to the PPU, allowing all 4 nametables to be supported. Thus, there is no need for a mirroring option, and games can scroll in all directions without the usual problem of the attribute table wrapping around the edge of the mirrored axis.

Mapper Register	Type	Data Bits 76543210	Control Function
$5000-$5FFF $7000-$7FFF (mirror)	(write-only)	edcbaaaa	a = CPU $8000-$FFFF Page Select b = PPU $0000-$1FFF Page Select c = PPU $2000-$3EFF Page Select d = Red LED - 0:On 1:Off e = MP3 Player TX - 1:Idle

Option Jumpers

GTMP3 offers a compatibility mode for certain NES clone systems which were based on a defective design, making full nametable control impossible. Some clone systems had games built-in to the system, in addition to a cart slot, and the hardware used the nametable enable pin as a way of detecting when an external cartridge is inserted. Later, some genius kept using this pin arrangement on clone systems which don't have built-in games, making the console incompatible with certain NES carts (Gauntlet, Rad Racer 2) and future homebrew designs, such as GTROM/GTMP3.

To make GTMP3 compatible with that garbage, we can cripple the nametable features of the mapper, using the provided jumpers. In this case, you also must solder the H or V jumper to select the mirroring type.

The nametable memory in this compatibility mode need not go to waste. There is a switch on the board made up of a surface-mount zero-ohm resistor, and a solder point jumper. Switch those, and the mapper's 2 CHR bits will now both be used for pattern table memory, providing 4 pages of 8kB.

Programming the Cartridge

A completely blank cartridge may be programmed via a few existing methods: CopyNES, INL-Retro, software-upgraded Game Genie, Kazzo. Additionally, the FlashROM is self-programmable by the NES. A bootloader may be used in conjunction with a controller port to USB communication adapter to reprogram the cartridge with no additional hardware. In the future, the cartridge will be available pre-loaded with this bootloader. You may include this bootloader in your own program, if you want your cartridge to remain rewritable with the communication adapter.

Development and Debugging Features

When used with an NES to USB communication adapter (available separately), which connects to the NES controller port, the cartridge is usable for developing and testing NES programs. In the future, some NES code and a PC program will be provided, so you can run a command on your PC to easily load your program onto the cartridge. In most cases it may be possible to remotely reset the NES from your PC.

The LED may be used as a simple form of output during debugging. For example, you may choose to turn the LED on, off, or flash a pattern when a certain part of your program has been reached. Or you may choose to disable the LED when entering an idle loop, and enable it during program execution, and this will result in the LED brightness being modulated by your program's CPU availability.

Game Genie Bootloader Operation

32kB paging mode (native) - mapper #111

On power-up, the mapper will begin with any one of the 16 banks selected. And because the program uploaded may be smaller than 512kB, the bootloader must put some code into the unused banks to launch the user's program. This also requires the bootloader to take over a small part of the user's flash memory, chosen to be at $FFF0 through $FFF9. Your program will always begin in bank 0. Here is the recommended code for compatibility with this system:

```
; .org $FFF0
      SEI             ; $FFF0
      LDA #$00        ; $FFF1
      STA $5000       ; $FFF3
      JMP ($FFFC)     ; $FFF6
      NOP             ; $FFF9
      .word your_NMI  ; $FFFA
      .word $FFF0     ; $FFFC RESET or $FFF0
      .word your_IRQ  ; $FFFE
```

Bank zero of your program should have your actual reset vector. It is recommended that the reset vector be set to $FFF0 on all other banks, though this is not required if you provide your own reset handler.

If you want full control of the FlashROM to create your own paged reset system, you can create a full 512kB sized ROM. For any other size, you will need this code at $FFF0-$FFF9. Note that if you leave that area padded with $FF, the bootloader will attempt to fill it in, even if you have created a 512kB ROM, but I recommend against relying on this feature. You will want to include that code yourself, especially if you are using an emulator for testing.

Unused banks that are outside of the range of your program, will automatically have their NMI, RESET, and IRQ vectors set to $FFF0, and the previous code stub inserted. The rest of the banks will be blank ($FF). Vectors in your program banks will not be changed by the loader.

16kB paging mode (simulated) - mapper #2

For your convenience, the bootloader can operate in a 16kB page mode, normally with $8000-$BFFF selectable, and $C000-$FFFF fixed to the last 16kB of your program. The opposite mode with $C000-$FFFF selectable (mapper #180) can be supported, please contact Memblers if this isn't already supported in the loader.

To use 16kB mode, create your ROM like a normal UNROM program, but using the GTROM mapper register at $5000 instead of the UNROM register location. It's misleading to call this mapper #2 support, because you can't run a normal mapper #2 program on the cartridge, unless you first modify it to use GTROM's control register.

Operating in 16kB mode reduces the maximum ROM size to 256kB.

FUQ (Frequently Unasked Questions)

Can this cartridge be used for making reproductions?

No, this cartridge is designed for newly-developed software. If you do have the legal right to release someone else's game, but don't have the source code, contact me. I may be able to modify the program to suit this board, or another.

I just want to make an NROM program, is this board overkill?

While it's true that only the very largest NES games used a 512kB ROM, those were made over 20 years ago. It's quite likely that the actual silicon die of a 512kB FlashROM is smaller than the 32kB MaskROMs used in early NES releases. Sizes smaller than 512kB are not significantly cheaper. And the savings from removing the mapper entirely are little consolation when it means that you can no longer reprogram the Flash in-system. The difference in cost is expected to be negated by producing a larger quantity of one design, because it has a wider range of applications.

Will the FlashROM ever wear out?

Eventually yes, each 4kB sector is rated for 100,000 erase cycles. To put this in perspective, with one sector you could save 4kB of data 10 times a day, every day, for over 25 years. If more endurance is needed, save data could use additional sectors alternately, or even be formatted into a smaller block that fits into each sector multiple times. For example, if you have 256 bytes of save data saved into 2 sectors, you could now save 320 times per day, every day, for over 25 years.

FlashROM is expected to eventually lose it's programming after 20+ years, as the electrons gradually escape via quantum tunneling effect. This is the same retention mechanism as EPROMs, which have a similar lifespan. But the advantage of Flash over EPROM is that it can be easily programmed in-circuit. In the distant future, a faded GTROM Flash can easily be restored by using a cart reading/writing device.

How will GTMP3 affect availability of GTROM?

GTROM costs less to produce, and since the selling price of the Cheapocabra is an important factor in it existing, GTROM is expected to be available for the foreseeable future.

Could an NES program on GTMP3 read the microSD card?

No, unfortunately even if the NES had an ideal connection to the chip (including hardware UART, readable registers, etc) the MP3 player chip does not provide anything other than status information.

Cheapocabra Icon License

Document History

v1.2	9-19-2019	added MP3 player section for GTMP3 variant
v1.1	7-25-2017	added bootloader operation section
v1.0	10-31-2015	Initial release