MMC3

From NESdev Wiki

The **Nintendo MMC3** is a mapper ASIC used in Nintendo's TxROM Game Pak boards. Most common TxROM boards, along with the **NES-HKROM** board (which uses the Nintendo MMC6), are assigned to iNES Mapper 004.

MMC3 TxROM	
Company	Nintendo, others
Games	300 in NesCartDB (https://nescartdb.com/search/advanced?ines=4)
Complexity	ASIC
Boards	TSROM, others
Pinout	MMC3 pinout
PRG ROM capacity	512K
PRG ROM window	8K + 8K + 16K fixed
PRG RAM capacity	8K
PRG RAM window	8K
CHR capacity	256K
CHR window	2Kx2 + 1Kx4
Nametable mirroring	H or V, switchable, or 4 fixed
Bus conflicts	No
IRQ	Yes
Audio	No
iNES mappers	004, 118, 119

NESCartDB
iNES 004 (https://nescartdb.com/search/advanced?ines=4)
TxROM (https://nescartdb.com/search/advanced?unif_op=LIKE+`%25%40%25`&unif=-T%25ROM)

Some less common MMC3 boards required alternative iNES mapper implementations:

- iNES Mapper 118 - TKSROM and TLSROM
- iNES Mapper 119 - TQROM

This chip first appeared in the fourth quarter of 1988.

Contents

Banks

- CPU $6000-$7FFF: 8 KB PRG RAM bank (optional)
- CPU $8000-$9FFF (or $C000-$DFFF): 8 KB switchable PRG ROM bank
- CPU $A000-$BFFF: 8 KB switchable PRG ROM bank
- CPU $C000-$DFFF (or $8000-$9FFF): 8 KB PRG ROM bank, fixed to the second-last bank
- CPU $E000-$FFFF: 8 KB PRG ROM bank, fixed to the last bank
- PPU $0000-$07FF (or $1000-$17FF): 2 KB switchable CHR bank
- PPU $0800-$0FFF (or $1800-$1FFF): 2 KB switchable CHR bank
- PPU $1000-$13FF (or $0000-$03FF): 1 KB switchable CHR bank
- PPU $1400-$17FF (or $0400-$07FF): 1 KB switchable CHR bank
- PPU $1800-$1BFF (or $0800-$0BFF): 1 KB switchable CHR bank
- PPU $1C00-$1FFF (or $0C00-$0FFF): 1 KB switchable CHR bank

Registers

The MMC3 has 4 pairs of registers at $8000-$9FFF, $A000-$BFFF, $C000-$DFFF, and $E000-$FFFF - even addresses ($8000, $8002, etc.) select the low register and odd addresses ($8001, $8003, etc.) select the high register in each pair. These can be broken into two independent functional units: memory mapping ($8000, $8001, $A000, $A001) and scanline counting ($C000, $C001, $E000, $E001).

Bank select ($8000-$9FFE, even)

```
7  bit  0
---- ----
CPMx xRRR
|||   |||
|||   +++- Specify which bank register to update on next write to Bank Data register
|||          000: R0: Select 2 KB CHR bank at PPU $0000-$07FF (or $1000-$17FF)
|||          001: R1: Select 2 KB CHR bank at PPU $0800-$0FFF (or $1800-$1FFF)
|||          010: R2: Select 1 KB CHR bank at PPU $1000-$13FF (or $0000-$03FF)
|||          011: R3: Select 1 KB CHR bank at PPU $1400-$17FF (or $0400-$07FF)
```

```
 |||          100: R4: Select 1 KB CHR bank at PPU $1800-$1BFF (or $0800-$0BFF)
 |||          101: R5: Select 1 KB CHR bank at PPU $1C00-$1FFF (or $0C00-$0FFF)
 |||          110: R6: Select 8 KB PRG ROM bank at $8000-$9FFF (or $C000-$DFFF)
 |||          111: R7: Select 8 KB PRG ROM bank at $A000-$BFFF
 ||+------- Nothing on the MMC3, see MMC6
 |+-------- PRG ROM bank mode (0: $8000-$9FFF swappable,
 |                                $C000-$DFFF fixed to second-last bank;
 |                             1: $C000-$DFFF swappable,
 |                                $8000-$9FFF fixed to second-last bank)
 +--------- CHR A12 inversion (0: two 2 KB banks at $0000-$0FFF,
                                four 1 KB banks at $1000-$1FFF;
                             1: two 2 KB banks at $1000-$1FFF,
                                four 1 KB banks at $0000-$0FFF)
```

CHR Banks

CHR map mode →	**$8000.D7 = 0**	**$8000.D7 = 1**
PPU Bank	**Value of MMC3 register**	
$0000-$03FF	R0	R2
$0400-$07FF		R3
$0800-$0BFF	R1	R4
$0C00-$0FFF		R5
$1000-$13FF	R2	R0
$1400-$17FF	R3	
$1800-$1BFF	R4	R1
$1C00-$1FFF	R5	

2KB banks may only select even numbered CHR banks. (The lowest bit is ignored.)

PRG Banks

Bit 6 of the last value written to $8000 swaps the PRG windows at $8000 and $C000. The MMC3 uses one map if bit 6 was cleared to 0 (value & $40 == $00) and another if set to 1 (value & $40 == $40).

PRG map mode →	**$8000.D6 = 0**	**$8000.D6 = 1**
CPU Bank	**Value of MMC3 register**	
$8000-$9FFF	R6	(-2)
$A000-$BFFF	R7	R7
$C000-$DFFF	(-2)	R6
$E000-$FFFF	(-1)	(-1)

- (-1) : the last bank
- (-2) : the second last bank

Because the values in R6, R7, and $8000 are unspecified at power on, the reset vector must point into $E000-$FFFF, and code must initialize these before jumping out of $E000-$FFFF.

Bank data ($8001-$9FFF, odd)

```
7  bit  0
---- ----
DDDD DDDD
|||| ||||
++++-++++- New bank value, based on last value written to Bank select register (mentioned above)
```

R6 and R7 will ignore the top two bits, as the MMC3 has only 6 PRG ROM address lines. Some romhacks rely on an 8-bit extension of R6/7 for oversized PRG-ROM, but this is deliberately not supported by many emulators. See iNES Mapper 004 below.

R0 and R1 ignore the bottom bit, as the value written still counts banks in 1KB units but odd numbered banks can't be selected.

Nametable arrangement ($A000-$BFFE, even)

```
7  bit  0
---- ----
xxxx xxxM
        |
        +- Nametable arrangement (0: horizontal (A10); 1: vertical (A11))
```

This bit has no effect on cartridges with hardwired 4-screen VRAM. In the iNES and NES 2.0 formats, this can be identified through bit 3 of byte $06 of the header.

PRG RAM protect ($A001-$BFFF, odd)

```
7  bit  0
---- ----
RWXX xxxx
||||
||++------ Nothing on the MMC3, see MMC6
|+-------- Write protection (0: allow writes; 1: deny writes)
+--------- PRG RAM chip enable (0: disable; 1: enable)
```

Disabling PRG RAM through bit 7 causes reads from the PRG RAM region to return open bus.

Though these bits are functional on the MMC3, their main purpose is to write-protect save RAM during power-off. Many emulators choose not to implement them as part of iNES Mapper 4 to avoid an incompatibility with the MMC6.

See iNES Mapper 004 and MMC6 below.

IRQ latch ($C000-$DFFE, even)

```
7  bit  0
```

```
---- ----
DDDD DDDD
|||| ||||
++++-++++- IRQ latch value
```

This register specifies the IRQ counter reload value. When the IRQ counter is zero (or a reload is requested through $C001), this value will be copied to the IRQ counter at the NEXT rising edge of the PPU address, presumably at PPU cycle 260 of the current scanline.

IRQ reload ($C001-$DFFF, odd)

```
7  bit  0
---- ----
xxxx xxxx
```

Writing any value to this register clears the MMC3 IRQ counter immediately, and then reloads it at the NEXT rising edge of the PPU address, presumably at PPU cycle 260 of the current scanline.

IRQ disable ($E000-$FFFE, even)

```
7  bit  0
---- ----
xxxx xxxx
```

Writing any value to this register will disable MMC3 interrupts AND acknowledge any pending interrupts.

IRQ enable ($E001-$FFFF, odd)

```
7  bit  0
---- ----
xxxx xxxx
```

Writing any value to this register will enable MMC3 interrupts.

Hardware

The MMC3 most commonly exists in a 44-pin TQFP package. Three revisions are known to exist - MMC3A, MMC3B, and MMC3C. No major behavioral differences are known, except for the IRQ counter.

MMC3 chips manufactured by NEC are implemented within a 13x56 CMOS gate array. The implementation method used by Sharp is not yet known.

IRQ Specifics

The MMC3 scanline counter is based entirely on PPU A12, triggered on a rising edge after the line has remained low for three falling edges of M2.[1]

The counter is based on the following trick: whenever rendering is turned on in the PPU, it fetches nametable and BG pattern tiles from dots 0-255 and 320-340 of a scanline and fetches sprite patterns from dots 256-319, even if no sprites are visible. Because of this, if BG uses the left pattern table ($0000), and if sprites always use the right pattern table ($1000), this filtered copy of A12 will remain low during all nametable and BG pattern fetches, and high during all sprite pattern fetches, causing it to oscillate exactly one time per scanline and 241 times per frame. It may oscillate more if the program uses registers $2006 and $2007 to access PPU $1000-$1FFF during vblank, but this is rare because very few games have MMC3 and CHR RAM (two on TQROM and three on TGROM among NES games, and a couple more Famicom-only games). The scanline counter partially works when the BG uses the right pattern table ($1000) and the sprites use the left pattern table ($0000), but the missing dot on the 2C02 causes the very first scanline to sometimes count twice. The MMC3 IRQ has two revisions that work slightly different [1] (http://forums.nesdev.org/viewtopic.php?p=62546#p62546).

Counter operation:

- When the IRQ is clocked (filtered A12 0→1), the counter value is checked - if zero **or** the *reload flag* is true, it's reloaded with the IRQ latched value at $C000; otherwise, it decrements.
- If the IRQ counter is zero *and* IRQs are enabled ($E001), an IRQ is triggered. The "alternate revision" checks the IRQ counter transition 1→0, whether from decrementing or reloading.

Regarding PPU A12:

- When using 8x8 sprites, if the BG uses $0000, and the sprites use $1000, the IRQ counter should decrement on PPU cycle 260, right after the visible part of the target scanline has ended.
- When using 8x8 sprites, if the BG uses $1000, and the sprites use $0000, the IRQ counter should decrement on PPU cycle 324 of the *previous* scanline (as in, *right before* the target scanline is about to be drawn). However, the 2C02's pre-render scanline will decrement the counter twice every other vertical redraw, so the IRQ will shake one scanline. This is visible in **Wario's Woods**: with some PPU-CPU reset alignments the bottom line of the green grass of the play area may flicker black on the rightmost ~48 pixels, due to an extra count firing the IRQ one line earlier than expected.
- When using 8x16 sprites *PPU A12 must be explicitly tracked*. The exact time and number of times the counter is clocked will depend on the specific set of sprites present on every scanline. Specific combinations of sprites could cause the counter to decrement up to four times, or the IRQ to be delayed *or early* by some multiple of 8 pixels. If there are fewer than 8 sprites on a scanline, the PPU fetches tile $FF ($1FE0-$1FFF) for each leftover sprite and discards its value. Thus if a game uses 8x16 sprites with its background and sprites from PPU $0000, then the MMC3 ends up counting each scanline that *doesn't* use all eight sprites.

Important points:

- The scanline counter cannot be stopped. It will continue to decrement and reload as long as PPU A12 on the PPU bus toggles.
- There is no direct access to the counter! The best that can be done is update the reload value and immediately request a reload.
- Writing to $E000 will only prevent the MMC3 from generating IRQs - the counter will continue to run.
- Writing to $E001 will simply allow the MMC3 to generate IRQs - the counter remains unaffected.
- Writing to $C001 will cause the counter to be cleared, and set *reload flag* to **true**. It will be reloaded on the NEXT rising edge of filtered A12.
- Writing to $C000 does not immediately affect the value within the counter - this value is only used when the counter is reloaded, whether from reaching 0 or from writing to $C001.

- The exact number of scanlines between IRQs is N+1, where N is the value written to $C000. 1 (Sharp MMC3B, MMC3C) or 2 (MMC3A, Non-Sharp MMC3B) to 256 scanlines are supported.
- The counter will not work properly unless different pattern tables for background and sprite data are in use. The standard configuration is to use PPU $0000-$0FFF for background tiles and $1000-$1FFF for sprite tiles, whether 8x8 or 8x16.
- The counter is clocked on each rising edge of filtered A12, no matter what caused it, so it is possible to (intentionally or not) clock the counter by writing to $2006, regardless of whether PPU is/is not rendering.

There's a slight discrepancy with what happens when $C000 is set to $00, and so far, two behaviors are known to exist:

- MMC3 made by NEC generates only a single IRQ when $C000 is $00. This version of the MMC3 generates IRQs when the scanline counter is *decremented* to 0. In addition, writing to $C001 with $C000 still at $00 will result in another single IRQ being generated. In the community, this has been known as the "alternate" or "old" behavior.
- MMC3 made by Sharp generates an IRQ on each scanline while $C000 is $00. This version of the MMC3 generates IRQs when the scanline counter is *equal* to 0. In the community, this has been known as the "normal" or "new" behavior.

Even on Sharp style MMC3, writing $C001 on consecutive scanlines can cause pathological behavior that Blargg and N-K encountered. See MMC3 IRQ Tests by blargg (https://github.com/christopherpow/nes-test-roms/blob/master/mmc3_irq_tests/readme.txt) and test ROM by N-K (https://forums.nesdev.org/viewtopic.php?p=261236#p261236). Because this behavior is not fully understood, emulators do not yet emulate it, though it'd be nice for a debugging emulator to break on $C001 writes without two intervening timer clocks.

- Many MMC3 games do not write $C001 in the IRQ handler, instead updating only $C000 to be applied at start of the *next* IRQ. By updating $C000 one interrupt ahead, this issue with $C001 can be avoided.

Acclaim's MC-ACC chip is their own variant of the MMC3, that they used for their own boards (for industrial money-saving purposes). It comes in a standard 600 mil 40-pin DIP package. It is not known if it has SRAM support. The only known difference is that the scanline counter uses a ÷8 prescaler instead of filtering A12 (https://forums.nesdev.org/viewtopic.php?p=242427#p242427), clocked by *falling* edges (https://forums.nesdev.org/viewtopic.php?p=116691#p116691).

iNES Mapper 004 and MMC6

The unfortunate conflation of MMC3 and MMC6 into the same iNES mapper can be resolved by the use of NES 2.0 submapper 004:1.

Since the new header is not yet well adopted among emulators or ROM sets, an approach that supports both MMC3 and MMC6 may be desired when an NES 2.0 header is not used.

The MMC6 has a smaller PRG-RAM, and a different register scheme for write protecting it.

Because write protection is generally only used to guard against corruption during power off, many implementations of iNES Mapper 004 simply omit the write protection. Leaving PRG-RAM always write-enabled removes most of the incompatibility between MMC3 and MMC6, and is sufficient to support the

popular MMC6 games StarTropics and StarTropics II. These games do not rely on the smaller PRG-RAM size of the MMC6, so the larger 8k RAM addressed by the MMC3 is not a problem.

The less well known game *Low G Man* is problematic. It used an MMC3 board with no PRG-RAM. Because of a bug in its music code, it relies on open-bus behaviour in the RAM's address range to function correctly. The game does use the MMC3 mechanism to disable RAM, so it may function on an MMC3 board with PRG-RAM, but implementing the MMC3 RAM disable may conflict with the effort to support MMC6 games. Alternatively, NES 2.0 could be used to specify a PRG-RAM size of 0, or the problem can be resolved by patching the Low G Man ROM to work around the conflict: patch (http://www.romhacking.net/hacks/2512/).

Some romhacks attempt to increase the available PRG-ROM size beyond the MMC3's hard 512k limit. Full 8-bit banking registers could theoretically support up to 2048K PRG-ROM for an oversized MMC3, but very few emulators implement this extension. Known examples:

- 2005 Translation of Final Fantasy III (https://www.romhacking.net/translations/1590/) - A later 2019 translation (https://www.romhacking.net/translations/4760/) fits in 512k.
- Rockman 4 Minus Infinity (https://www.romhacking.net/hacks/910/) - An MMC5 alternative page is also available.

Variants

Board wiring variants

The TKSROM and TLSROM boards, assigned to INES Mapper 118, connect the upper bank select line directly to VRAM A10, allowing more flexible control over nametable arrangement.

The TQROM board, assigned to INES Mapper 119, uses both CHR ROM and CHR RAM simultaneously, using the 2nd-highest CHR bank select line to choose between them (on a per-bank basis).

Nintendo made at least two MMC3-based multicart mappers: mappers 37 and 47.

MMC3-like chips variants

The DxROM board, assigned to iNES Mapper 206 has a custom mapper developed by Namco before the MMC3 existed. It has no IRQ nor control over nametable arrangement. Tengen used it for some of their games under the name MIMIC-1. It exists both in a 400 mil 28-pin Shrink-DIP (found in licensed DxROM boards) and in a larger 600 mil 28-pin DIP (found in unlicensed Tengen cartridges).

The Namco 108/MIMIC-1 does the basic ROM banking exactly like the MMC3, but it only implements the low 3 bits of $8000 and the low 6 bits of $8001. Compared to the MMC3, it lacks nametable control, SRAM support and an IRQ counter. The TEROM and TFROM boards have been developed with backward compatibility with DxROM in mind, featuring solder pads to have hardwired H/V nametables instead of controlled by the MMC3, and allow the hardware to disable IRQs.

Nintendo MMC3 chip variants

IRQ behavior when reload is set to 0 differs among different MMC3 chips. Sharp MMC3 chips generate an IRQ every scanline. (These include at least MMC3B chips bearing a bold S before the date code.) NEC MMC3 chips cease to generate IRQs. (These include MMC3B chips lacking the S and having a date code of the form nnnnPKnnn, and an MMC3A 8940EP chip.) Some games have been manufactured with both versions. A few later games, such as *Star Trek: 25th Anniversary (https://nescartdb.com/profile/view/249/star-trek-25th-anniversary)*, rely on the Sharp behavior (source: Nestopia 1.30 changelog).

There's an anecdotal report that *Felix the Cat* needs MMC3A (https://forums.nesdev.org/viewtopic.php?p=124478#p124478), but it's necessarily not related to the IRQ behavior since the game works correctly in emulators that only implement the Sharp behavior.

Pirate variants

- Mapper 004 submapper 5 is a Waixing variant with a software-controlled T9552 chip that scrambles PRG/CHR address lines. Known games all initialize the chip with the same canonical pattern, so they can be dumped as MMC3 compatible ROMs, but the submapper designation allows accurate emulation of the scrambling. This was formerly assigned as mapper 249 (now deprecated), with the banks in scrambled order instead.
- Mapper 245 increases PRG to 1024K by losing CHR ROM.
- Mappers 205, 52, 49, 45, and 44 force unmodified games together in a multicart.
- Mapper 189 loses the MMC3's 8+8+16F banking scheme in exchange for 32k-at-a-time banking like AxROM, BxROM, or GxROM
- Mapper 182's registers are only in a different order.
- Mapper 115 increases CHR to 512K by losing PRG RAM and contains an UxROM emulation mode.
- Mappers 194, 192, 191, and 74, are like TQROM in that they combine CHR ROM and CHR RAM by replacing some CHR pages with CHR RAM.

See also

- MMC3-like mappers
- NES Mapper list (https://www.romhacking.net/documents/362/) by Disch.
- Nintendo MMC3 (https://nesdev.org/mmc3.txt) by goroh.
- Comprehensive NES Mapper Document (https://nesdev.org/mappers.zip) by \Firebug\. Information on mapper's initial state is innacurate.

References

1. The last page in Furrtek's reverse-engineered MMC3C schematic (https://github.com/furrtek/VGChips/tree/master/Nintendo/MMC3C) contains the A12 filtering circuit. /C16 here is the IRQ counter clock.