

Everynes - Nocash NES Specs

Everynes Hardware Specifications

[Tech Data](#)

[Memory Maps](#)

[I/O Map](#)

[Picture Processing Unit \(PPU\)](#)

[Audio Processing Unit \(APU\)](#)

[Controllers](#)

[Cartridges and Mappers](#)

[Famicom Disk System \(FDS\)](#)

[Hardware Pin-Outs](#)

[CPU 65XX Microprocessor](#)

[About Everynes](#)

Tech Data

Overall Specs

CPU 2A03 - customized 6502 CPU - audio - does not contain support for decimal

The NTSC NES runs at 1.7897725MHz, and 1.7734474MHz for PAL.

NMIs may be generated by PPU each VBlank.

IRQs may be generated by APU and by external hardware.

Internal Memory: 2K WRAM, 2K VRAM, 256 Bytes SPR-RAM, and Palette/Registers

The cartridge connector also passes audio in and out of the cartridge, to allow for external sound chips to be mixed in with the Famicom audio.

Original Famicom (Family Computer) (1983) (Japan)

60-pin cartridge slot, with external sound-input, without lockout chip.

Two joypads directly attached to console, Joypad 1 with Start/Select buttons, Joypad 2 with microphone, but without Start/Select. 15pin Expansion port for further controllers. Video RF-Output only.

"During its first year, people found the Famicom to be unreliable, with programming errors and freezing rampant. Yamauchi recalled all sold Famicom systems, and put the Famicom out of production until the errors were fixed. The Famicom was re-released with a new motherboard."

Original NES (Nintendo Entertainment System) (1985) (US, Europe, Australia)

Same as Famicom, but with slightly different pin-outs on cartridge slot, and controllers/expansion ports:

Front-loading 72-pin cartridge slot, without external sound-input on cartridge slot, without microphone on joypads, with lockout chip.

Newer Famicom, AV Famicom (1993-1995)

60-pin cartridge slot, with external sound-input, without lockout chip.

Includes NES-style joystick connectors, plus the original 15pin Famicom Expansion port. Doesn't have microphone. Video AV-Output only.

Newer NES (1993-1995)

Top-loading 72-pin cartridge slot, without external sound-input, without lockout chip. Poorer video signal than old NES. Video WHAT?-output only.

VS Unisystem

Arcade Machine. Coin-detect inputs, eight Dip-switches, different palette.

Play Choice 10

Arcade Machine with 10 cartridge slots. Uses Z80 as second CPU.

Memory Maps

Internal Memory: 2K WRAM, 2K VRAM, 256 Bytes SPR-RAM, and Palette/Registers

CPU Memory Map (16bit buswidth, 0-FFFFh)

0000h-07FFh	Internal 2K Work RAM (mirrored to 800h-1FFFh)
2000h-2007h	Internal PPU Registers (mirrored to 2008h-3FFFh)
4000h-4017h	Internal APU Registers
4018h-5FFFh	Cartridge Expansion Area almost 8K
6000h-7FFFh	Cartridge SRAM Area 8K
8000h-FFFFh	Cartridge PRG-ROM Area 32K

CPU Reset vector located at [FFFC], even smaller carts must have memory at that location. Larger carts may use whatever external mappers to access more than the usual 32K.

PPU Memory Map (14bit buswidth, 0-3FFFh)

0000h-0FFFh	Pattern Table 0 (4K) (256 Tiles)
1000h-1FFFh	Pattern Table 1 (4K) (256 Tiles)
2000h-23FFh	Name Table 0 and Attribute Table 0 (1K) (32x30 BG Map)
2400h-27FFh	Name Table 1 and Attribute Table 1 (1K) (32x30 BG Map)
2800h-2BFFh	Name Table 2 and Attribute Table 2 (1K) (32x30 BG Map)
2C00h-2FFFh	Name Table 3 and Attribute Table 3 (1K) (32x30 BG Map)
3000h-3EFFh	Mirror of 2000h-2EFFh
3F00h-3F1Fh	Background and Sprite Palettes (25 entries used)
3F20h-3FFFh	Mirrors of 3F00h-3F1Fh

Note: The NES contains only 2K built-in VRAM, which can be used for whatever purpose (for example, as two Name Tables, or as one Name Table plus 64 Tiles). Palette Memory is built-in as well. Any additional VRAM (or, more regularly, VROM) is located in the cartridge, which may also contain mapping hardware to access more than 12K of video memory.

SPR-RAM Memory Map (8bit buswidth, 0-FFh)

00-FF	Sprite Attributes (256 bytes, for 64 sprites / 4 bytes each)
-------	--

Sprite RAM is directly built-in in the PPU chip. SPR-RAM is not connected to CPU or PPU bus, and can be accessed via I/O Ports only.

I/O Map

I/O Map

2000h	- PPU Control Register 1 (W)
2001h	- PPU Control Register 2 (W)
2002h	- PPU Status Register (R)
2003h	- SPR-RAM Address Register (W)
2004h	- SPR-RAM Data Register (RW)
2005h	- PPU Background Scrolling Offset (W2)
2006h	- VRAM Address Register (W2)
2007h	- VRAM Read/Write Data Register (RW)
4000h	- APU Channel 1 (Rectangle) Volume/Decay
4001h	- APU Channel 1 (Rectangle) Sweep
4002h	- APU Channel 1 (Rectangle) Frequency
4003h	- APU Channel 1 (Rectangle) Length
4004h	- APU Channel 2 (Rectangle) Volume/Decay

4005h - APU Channel 2 (Rectangle) Sweep
 4006h - APU Channel 2 (Rectangle) Frequency
 4007h - APU Channel 2 (Rectangle) Length
 4008h - APU Channel 3 (Triangle) Linear Counter
 4009h - APU Channel 3 (Triangle) N/A
 400Ah - APU Channel 3 (Triangle) Frequency
 400Bh - APU Channel 3 (Triangle) Length
 400Ch - APU Channel 4 (Noise) Volume/Decay
 400Dh - APU Channel 4 (Noise) N/A
 400Eh - APU Channel 4 (Noise) Frequency
 400Fh - APU Channel 4 (Noise) Length
 4010h - APU Channel 5 (DMC) Play mode and DMA frequency
 4011h - APU Channel 5 (DMC) Delta counter load register
 4012h - APU Channel 5 (DMC) Address load register
 4013h - APU Channel 5 (DMC) Length register
 4014h - SPR-RAM DMA Register
 4015h - DMC/IRQ/length counter status/channel enable register (RW)
 4016h - Joypad #1 (RW)
 4017h - Joypad #2/APU SOFTCLK (RW)

Additionally, external hardware may contain further ports:

4020h - VS Unisystem Coin Acknowledge
 4020h-40FFh - Famicom Disk System (FDS)
 4100h-FFFFh - Various addresses used by various cartridge mappers

Picture Processing Unit (PPU)

[PPU Control and Status Registers](#)

[PPU SPR-RAM Access Registers](#)

[PPU VRAM Access Registers](#)

[PPU Scrolling](#)

[PPU Tile Memory](#)

[PPU Background](#)

[PPU Sprites](#)

[PPU Palettes](#)

[PPU Dimensions & Timings](#)

Based on "Nintendo Entertainment System Documentation" Version 2.00 by Jeremy Chadwick aka YOSHi aka JDC. Which was itself based on "Nintendo Entertainment System Architecture" by Marat Fayzullin.

PPU Control and Status Registers

2000h - PPU Control Register 1 (W)

Bit7	Execute NMI on VBlank	(0=Disabled, 1=Enabled)
Bit6	PPU Master/Slave Selection	(0=Master, 1=Slave) (Not used in NES)
Bit5	Sprite Size	(0=8x8, 1=8x16)
Bit4	Pattern Table Address Background	(0=VRAM 0000h, 1=VRAM 1000h)
Bit3	Pattern Table Address 8x8 Sprites	(0=VRAM 0000h, 1=VRAM 1000h)
Bit2	Port 2007h VRAM Address Increment	(0=Increment by 1, 1=Increment by 32)
Bit1-0	Name Table Scroll Address	(0-3=VRAM 2000h, 2400h, 2800h, 2C00h)
	(That is, Bit0=Horizontal Scroll by 256, Bit1=Vertical Scroll by 240)	

2001h - PPU Control Register 2 (W)

Bit7-5	Color Emphasis	(0=Normal, 1-7=Emphasis) (see Palettes chapter)
Bit4	Sprite Visibility	(0=Not displayed, 1=Displayed)
Bit3	Background Visibility	(0=Not displayed, 1=Displayed)

Bit2	Sprite Clipping	(0=Hide in left 8-pixel column, 1=No clipping)
Bit1	Background Clipping	(0=Hide in left 8-pixel column, 1=No clipping)
Bit0	Monochrome Mode	(0=Color, 1=Monochrome) (see Palettes chapter)

If both sprites and BG are disabled (Bit 3,4=0) then video output is disabled, and VRAM can be accessed at any time (instead of during VBlank only). However, SPR-RAM does no longer receive refresh cycles, and its content will gradually degrade when the display is disabled.

2002h - PPU Status Register (R)

Bit7	VBlank Flag	(1=VBlank)
Bit6	Sprite 0 Hit	(1=Background-to-Sprite0 collision)
Bit5	Lost Sprites	(1=More than 8 sprites in 1 scanline)
Bit4-0	Not used	(Undefined garbage)

Reading resets the 1st/2nd-write flipflop (used by Port 2005h and 2006h).

Reading resets Bit 7, can be used to acknowledge NMIs, Bit 7 is also automatically reset at the end of VBlank, so manual acknowledge is normally not required (unless one wants to free the NMI signal for external NMI inputs).

Status Notes

VBlank flag is set in each frame, even if the display is fully disabled, and even if NMIs are disabled. Hit flag may become set only if both BG and OBJ are enabled. Lost Sprites flag may become set only if video is enabled (ie. BG or OBJ must be on). For info about the "Not used" status bits, and some other PPU bits see:

[Unpredictable Things](#)

PPU SPR-RAM Access Registers

2003h - SPR-RAM Address Register (W)

D7-D0: 8bit address in SPR-RAM (00h-FFh)

Specifies the destination address in Sprite RAM for use with Port 2004h (Single byte write), and Port 4014h (256 bytes DMA transfer).

This register is internally used during rendering (and typically contains 00h at the begin of the VBlank period).

2004h - SPR-RAM Data Register (Read/Write)

D7-D0: 8bit data written to SPR-RAM.

Read/write data to/from selected address in Sprite RAM.

The Port 2003h address is auto-incremented by 1 after each <write> to 2004h.

The address is NOT auto-incremented after <reading> from 2004h.

4014h - Sprite DMA Register (W)

Transfers 256 bytes from CPU Memory area into SPR-RAM. The transfer takes 512 CPU clock cycles, two cycles per byte, the transfer starts about immediately after writing to 4014h: The CPU either fetches the first byte of the next instruction, and then begins DMA, or fetches and executes the next instruction, and then begins DMA. The CPU is halted during transfer.

Bit7-0 Upper 8bit of source address (Source=N*100h) (Lower bits are zero)

Data is written to Port 2004h. The destination address in SPR-RAM is thus [2003h], which should be normally initialized to zero - unless one wants to "rotate" the target area, which may be useful when implementing more than eight (flickering) sprites per scanline.

Notes

SPR-RAM should be accessed during VBlank only. SPR-RAM is dynamic memory, refreshed during rendering, it does no longer receive refresh cycles (and will lose its content) when the display is disabled (by clearing both Bit 3 and 4 in Port 2001h).

PPU VRAM Access Registers

Registers used to Read and Write VRAM data, and for Background Scrolling.

The CPU can Read/Write VRAM during VBlank only - because the PPU permanently accesses VRAM during rendering (even in HBlank phases), and because the PPU uses the VRAM Address register as scratch pointer. Respectively, the address in Port 2006h is destroyed after rendering, and must be re-initialized before using Port 2007h.

1st/2nd Write

Below Port 2005h and 2006h require two 8bit writes to receive a 16bit parameter, the current state (1st or 2nd write) is memorized in a single flipflop, which is shared for BOTH Port 2005h and 2006h. The flipflop is reset when reading from PPU Status Register Port 2002h (the next write will be then treated as 1st write) (and of course it is also reset after any 2nd write).

2005h - PPU Background Scrolling Offset (W2)

Defines the coordinates of the upper-left background pixel, together with PPU Control Register 1, Port 2000h, Bits 0-1).

```
Port 2005h-1st write: Horizontal Scroll Origin (X*1) (0-255)
Port 2005h-2nd write: Vertical Scroll Origin (Y*1) (0-239)
Port 2000h-Bit0: Horizontal Name Table Origin (X*256)
Port 2000h-Bit1: Vertical Name Table Origin (Y*240)
```

Caution: The above scroll reload settings are overwritten by writes to Port 2006h. See PPU Scrolling chapter for more info.

2006h - VRAM Address Register (W2)

Used to specify the 14bit VRAM Address for use with Port 2007h.

```
Port 2006h-1st write: VRAM Address Pointer MSB (6bit)
Port 2006h-2nd write: VRAM Address Pointer LSB (8bit)
```

Caution: Writes to Port 2006h are overwriting scroll reload bits (in Port 2005h and Bit0-1 of Port 2000h). And, the PPU uses the Port 2006h register internally during rendering, when the display is enabled one should thus reinitialize Port 2006h at begin of VBlank before accessing VRAM via Port 2007h.

2007h - VRAM Read/Write Data Register (RW)

The PPU will auto-increment the VRAM address (selected via Port 2006h) after each read/write from/to Port 2007h by 1 or 32 (depending on Bit2 of \$2000).

```
Bit7-0 8bit data read/written from/to VRAM
```

Caution: Reading from VRAM 0000h-3EFFh loads the desired value into a latch, and returns the OLD content of the latch to the CPU. After changing the address one should thus always issue a dummy read to flush the old content. However, reading from Palette memory VRAM 3F00h-3FFFh, or writing to VRAM 0000-3FFFh does directly access the desired address.

PPU Scrolling

The PPU allows to scroll the background pixelwise horizontally and vertically. The total scroll-able area is 512x480 pixels (though the full size can be used with external memory only, see Name Tables chapter), of which circa 256x240 pixels are displayed (see visible screen resolution).

Vertical offsets 240-255 (aka Tile Rows 30-31) will cause garbage Tile numbers to be fetched from the Attribute Table (instead of from Name Table), after line 255 it will wrap to line 0, but without producing a carry-out to the Name Table Address.

Scroll Pointer and Reload Registers

Scrolling relies on a Pointer register (Port 2006h), and on a Reload register (Port 2005h, and Bit0-1 of Port 2000h). The Pointer is automatically incremented by the hardware during rendering, and points to the currently drawn tile row, the same pointer register is also used by software to access VRAM during VBlank or when the display is disabled. The Reload value defines the horizontal and vertical origin of upper-left pixel, the reload value is automatically loaded into the Pointer at the end of the vblank period (vertical reload bits), and at the begin of each scanline (horizontal reload bits). The relation between Pointer and Reload bits is:

VRAM-Pointer			Scroll-Reload	
A8	2006h/1st-Bit0	<-->	Y*64	2005h/2nd-Bit6
A9	2006h/1st-Bit1	<-->	Y*128	2005h/2nd-Bit7
A10	2006h/1st-Bit2	<-->	X*256	2000h-Bit0
A11	2006h/1st-Bit3	<-->	Y*240	2000h-Bit1
A12	2006h/1st-Bit4	<-->	Y*1	2005h/2nd-Bit0
A13	2006h/1st-Bit5	<-->	Y*2	2005h/2nd-Bit1
-	2006h/1st-Bit6	<-->	Y*4	2005h/2nd-Bit2
-	2006h/1st-Bit7	<-->	-	-
A0	2006h/2nd-Bit0	<-->	X*8	2005h/1st-Bit3
A1	2006h/2nd-Bit1	<-->	X*16	2005h/1st-Bit4
A2	2006h/2nd-Bit2	<-->	X*32	2005h/1st-Bit5
A3	2006h/2nd-Bit3	<-->	X*64	2005h/1st-Bit6
A4	2006h/2nd-Bit4	<-->	X*128	2005h/1st-Bit7
A5	2006h/2nd-Bit5	<-->	Y*8	2005h/2nd-Bit3
A6	2006h/2nd-Bit6	<-->	Y*16	2005h/2nd-Bit4
A7	2006h/2nd-Bit7	<-->	Y*32	2005h/2nd-Bit5
-	-	<-->	X*1	2005h/1st-Bit0
-	-	<-->	X*2	2005h/1st-Bit1
-	-	<-->	X*4	2005h/1st-Bit2

Port 2006h-1st Write (VRAM Pointer MSB)

As one might (not) have expected, this does NOT change the VRAM Pointer, instead, the written value is stored in the corresponding Reload bits (Port 2005h/2000h settings), the VRAM pointer is left unchanged for now.

Port 2006h-2nd Write (VRAM Pointer LSB)

The written value is stored in the VRAM Pointer LSB Bits (and maybe also in the corresponding Reload bits ?). And, the VRAM Pointer MSB is now loaded from the corresponding Reload bits (ie. the value from the previous Port 2006h-1st Write is applied now).

Port 2005h-1st Write (Horizontal Scroll Origin, X*1, 0-255)

Port 2005h-2nd Write (Vertical Scroll Origin, Y*1, 0-239)

Port 2000h-Bit0 (Horizontal Name Table Origin, X*256)

Port 2000h-Bit1 (Vertical Name Table Origin, Y*240)

Writing to these registers changes the Reload value bits only, the VRAM Pointer is left unchanged (except for indirect changes at times when the Reload value is loaded into the Pointer during rendering).

Full-screen and Mid-frame Scrolling

Simple full-screen scrolling can be implemented by initializing the Reload value via Ports 2005h and 2000h. Many games change the scroll settings mid-frame to split the screen into a scrolled and non-scrolled area: The Horizontal bits can be changed by re-writing the Reload value via Ports 2005h and 2000h, the vertical bits by re-writing the Pointer value via Port 2006h. Changing both horizontal and vertical bits is possible by mixed writes to Port 2005h and 2006h, for example:

$$[2006h.1st] = (X/256) * 4 + (Y/240) * 8$$

```
[2005h.2nd]=((Y MOD 240) AND C7h)
[2005h.1st]=(X AND 07h)
[2006h.2nd]=(X AND F8h)/8 + ((Y MOD 240) AND 38h)*4
```

Notes: In that example, most bits are updated twice, once via 2006h and once via 2005h, above shows only the relevant bits, the other bits would be don't care (eg. writing unmasked values to 2005h would be faster, and wouldn't change the functionality). The 1st/2nd-write-flipflop is toggled on each of the four writes, so that above does <first> change 2005h-2nd-write, and <then> 2005h-1st-write.

Pointer Increment/Reload during Rendering

During rendering, A4-A0 is incremented per tile, with carry-out to A10, at end of HBlank A4-A0 and A10 are reset to the Reload value. "A14-A12" are used as LSBs of Tile Data address, these bits are incremented per scanline, with carry-out to tile row A9-A5, the tile row wraps from 29 to 0 with carry-out to A11.

Note: Initializing the tile row to 30 or 31 will display garbage tiles (fetched from Attribute table area), in that case the row wraps from 31 to 0, but without carry-out to A11.

PPU Tile Memory

PPU 0000h-0FFFh - Pattern Table 0 (4K) (256 Tiles)

PPU 1000h-1FFFh - Pattern Table 1 (4K) (256 Tiles)

Pattern Table Format

Each pattern table contains 256 tiles. When using both pattern table 0 and 1, up to 512 tiles can be used for Background and Sprites.

Each tile consists of a 8x8 pixel bitmap with 2bit depth (4 colors). Each tile occupies 16 bytes, the first 8 bytes contain color bit 0 for each pixel, the next 8 bytes color bit 1. Each byte defines a row of 8 pixels (MSB left).

Pattern Table Memory

The console does NOT include built-in Pattern Table Memory. Instead, Pattern tables are located in the cartridge, usually in a separate ROM chip, or (less often) in a SRAM chip. Cartridges with more than 8K Pattern memory may contain whatever mapping mechanisms to map the memory into the PPU 8K Pattern Memory area.

PPU Background

PPU 2000h-23FFh - Name Table 0 and Attribute Table 0 (1K)

PPU 2400h-27FFh - Name Table 1 and Attribute Table 1 (1K)

PPU 2800h-2BFFh - Name Table 2 and Attribute Table 2 (1K)

PPU 2C00h-2FFFh - Name Table 3 and Attribute Table 3 (1K)

PPU 3000h-3EFFh - Mirror of 2000h-2EFFh

Name Table Format

Each Name Table occupies 3C0h bytes, containing 8bit tile numbers for 32x30 tiles (256x240 pixels). The tiles are fetched from Pattern Table 0 or 1 (depending on Bit 4 in PPU Control Register 1). Note that NTSC displays may be unable to display the whole 256x240 pixels, basically the relevant portion of screen output should be in the <middle> 32x28 tiles (256x224 pixels) see PPU Dimensions and Timings chapter for more info.

Attribute Table Format

Each Name Table is directly followed by an Attribute Table of 40h bytes, containing 2bit background

palette numbers for each 16x16 pixel field. Each byte in the Attribute table defines palette numbers for a 32x32 pixel area:

```
Bit0-1  Palette Number for upperleft 16x16 pixels of the 32x32 area
Bit2-3  Palette Number for upperright 16x16 pixels of the 32x32 area
Bit4-5  Palette Number for lowerleft 16x16 pixels of the 32x32 area
Bit6-7  Palette Number for lowerright 16x16 pixels of the 32x32 area
```

Note: Attributes for each 8x1 pixel row are fetched from cartridge bus. The MMC5 Mapper with EXRAM allows to use different palettes for each 8x8 pixel tile, instead of sharing one palette for above 16x16 areas.

Background Scrolling

Scrolling origin is defined by the Name Table selection in Bit0-1 of \$2000, and by offsets in \$2005, of which Horizontal offsets may range in 0-255, vertical offsets in 0-239; values above 239 are considered negative (eg. 248 is -8). The picture wraps to the next Name Table when drawing exceeds the boundaries of the current Name Table...

Multiple Name Tables

The NES has the capability of addressing up to four Name Tables (NT0-3), allowing to define backgrounds of up to 512x480 pixels, arranged as such:

```
Square      Horizontal Scroll  Vertical Scroll
NT0 NT1     NT0 left/right NT1  NT0 above/below NT2
NT2 NT3     NT2 left/right NT3  NT1 above/below NT3
```

However, the NES includes only 2K VRAM, so that not more than two Name Tables can be used (unless the cartridge includes external Name Table memory).

Name Table Mapping/Mirroring

The NES outputs the desired Name Table number (NT0-3) to the cartridge, which may then respond by selecting one of the two internal 1K RAM blocks (BLK0-1), or by presenting an external RAM/ROM block (eg. BLK2-3). Examples:

Name Table	NT0	NT1	NT2	NT3	Purpose
Horizontal Mirroring	BLK0	BLK0	BLK1	BLK1	Vertical Scrolling
Vertical Mirroring	BLK0	BLK1	BLK0	BLK1	Horizontal Scrolling
Four-screen	BLK0	BLK1	BLK2	BLK3	Four-Way Scrolling

When using only the internal blocks, the cartridge may use a simple hardwired connection between two pins to select horizontal or vertical mirroring. Also, the cartridge may contain whatever circuits to map Single-Screen or CHR-ROM to whatever addresses dependently or independently of the selected NT number.

Background Clipping

The PPU allows to mask the left 8 pixels of BG, allowing to use horizontal scrolling with only one Name Table, the 16pix-width palette attribute isn't fully clipped though. Also, BG could be vertically clipped by software, which would require accurate timing though. Aside from that, it'd be no problem to implement four-way scrolling by using only one name table.

PPU Sprites

SPR-RAM 00-FF - Sprite Attributes (256 bytes, for 64 sprites / 4 bytes each)

The PPU supports 64 sprites, which can be either 8x8 or 8x16 pixels in size, only 8 sprites can be displayed per scanline. The sprite Tile bitmaps are kept within the Pattern Table region of VRAM (which is also used for BG Tiles).

Sprite-RAM is built-in in the PPU-chip, and can be accessed via I/O ports only (it is not part of the PPU or CPU memory area). Each four bytes in SPR-RAM define attributes for one sprite, bytes 0-3 for sprite 0, up to bytes FCh-FFh for sprite 63.

SPR-RAM Byte 0 - Y Coordinate Minus 1

Vertical Position-1 (FFh,00h..Eh=Scanline 0..239, EFh..FEh=Not displayed)

The sprites can be moved bottom-offscreen, but cannot be moved top-offscreen.

SPR-RAM Byte 1 - Tile Number

In 8x8 pixel mode (PPU Control Register 1, Bit5=0):

Bit7-0 Specifies 8bit tile number
And, Pattern Table selected by Bit 3 in PPU Control Register 1

In 8x16 pixel mode (PPU Control Register 1, Bit5=1):

Bit7-1 Upper 7bit of tile number (N=0-127 uses Tiles N*2 and N*2+1)
Bit0 Pattern Table Address (0=VRAM 0000h, 1=VRAM 1000h)

SPR-RAM Byte 2 - Attributes

7 Vertical Flip (0=Normal, 1=Mirror)
6 Horizontal Flip (0=Normal, 1=Mirror)
5 Background Priority (0=Sprite In front of BG, 1=Sprite Behind BG)
4-2 Not used (Always zero when reading from SPR-RAM)
1-0 Sprite Palette (0-3=Sprite Palette 0-3)

SPR-RAM Byte 3 - X Coordinate

Horizontal Position (00h..FFh)

Sprites can be moved right-offscreen, and clipping via Port 2001h also allows to move sprites somewhat left-offscreen.

Sprite Priorities

If two or more non-transparent sprite-pixels overlap, then only the sprite with highest priority is processed. If the sprites background priority bit is set to "Behind BG", then it will be hidden behind any non-transparent background pixels.

Sprite 0 = highest priority
Sprite 63 = lowest priority

Mind that the PPU processes ONLY the sprite with highest priority, eg. if a non-transparent pixel of sprite 5 hides "Behind BG", then sprite 6-63 won't be displayed (even if they are "In Front of BG").

Sprite 0 Hit Flag (Collision Check between BG and Sprite 0)

The Hit Flag, Bit 6 of register 2002h, gets set when the cathode ray beam passes a non-transparent Sprite 0 pixel which is overlapping a non-transparent BG pixel (regardless the sprites BG priority).

The Hit Flag is automatically reset at the end of the VBlank period, it cannot be set or reset by software, that means one can detect only one Hit per frame.

Aside from a normal collision detection, the Hit Flag is also useful to detect when the cathode ray beam has reached a specific screen location, eg. to split the picture into a scrolled and non-scrolled section.

Color 1 or color 2 are (NOT) non-transparent (?)

PPU Palettes

PPU 3F00h-3F1Fh - Background and Sprite Palettes

Palette Memory (25 entries used)

3F00h Background Color (Color 0)
3F01h-3F03h Background Palette 0 (Color 1-3)
3F05h-3F07h Background Palette 1 (Color 1-3)
3F09h-3F0Bh Background Palette 2 (Color 1-3)
3F0Dh-3F0Fh Background Palette 3 (Color 1-3)
3F11h-3F13h Sprite Palette 0 (Color 1-3)
3F15h-3F17h Sprite Palette 1 (Color 1-3)
3F19h-3F1Bh Sprite Palette 2 (Color 1-3)
3F1Dh-3F1Fh Sprite Palette 3 (Color 1-3)

Palette Gaps and Mirrors

3F04h, 3F08h, 3F0Ch - Three general purpose 6bit data registers.
3F10h, 3F14h, 3F18h, 3F1Ch - Mirrors of 3F00h, 3F04h, 3F08h, 3F0Ch.
3F20h-3FFFh - Mirrors of 3F00h-3F1Fh.

Palette Entries

Bit7-6 Not used (contains garbage when reading palette memory)
Bit5-4 Luminance (Grayscale) (0-3)
Bit3-0 Chrominance (Color) (0-F)

The Color values are based on the NTSC color-wheel (even on PAL consoles):

	0		1	2	3	4	5	6	7	8	9	A	B	C		D		E	F	
	White		..Blue	..Magenta	..Red	Yellow	..Green	Blue		Gray		Black						

Color and Grayscale (0-3) can be combined as such:

Luminance	0Xh	1Xh	2Xh	3Xh
Color 0:	Med Gray,	Light Gray,	White,	White
Color 1-C:	(Dark),	(Normal),	(Brighter),	(Brightest/Pastelized)
Color D:	Reserved,	Black,	Dark Gray,	Lighter Gray
Color E-F:	Black,	Black,	Black,	Black

Some black/white colors are duplicated, one should normally use 0Eh or 0Fh as black. Of the two Light Grays, 3Dh is slightly brighter than 10h. The Reserved color is Blacker-than-black, producing a very low voltage, which some monitors may or may not treat to be a sync signal rather than a color.

Monochrome Television Set (nine different grayshades)

On mono TV sets, Colors 0 and D-F can be used for Black/Gray/White colors as usually, and Colors 1-C are all displayed as grayshades: 01h-0Ch=Black, 11h-1Ch=Med/Dark Gray, 21h-2Ch=Med/Light Gray, 31h-3Ch=Bright Gray. Intensity ramp example: 0Eh, 2Dh, 11h, 00h, 21h, 01h, 3Dh, 31h, 30h. Also, the Color Emphasis bits are somewhat affecting the luminance output, even on mono television sets.

Monochrome Bit (three different grayshades)

When Port 2001h/Bit0 is set: The lower 4bits of all palette entries are treated to be zero. That means that only 3 colors can be displayed: Gray, Light Gray, and White. All other colors cannot be used (even Black and Dark Gray are disabled). The Color Emphasis bits can be still used (eg. to change above 3 gray-shades into 3 pastelized green-shades).

Color Emphasis Bits

Port 2001h/Bit7-5 allow to adjust the palette, eg. with setting 001b the whole picture becomes more green.

000b Normal
001b Green
010b Brown
100b Blue

To play by the rules, one should reportedly not set more than one of the emphasis bits at once, setting two

or more bits may shortcut something inside the PPU, though it doesn't seem to damage the chip.

Border Color and Clipping Colors

The screen border is black, regardless of any palette settings.

The Color 0 setting is displayed when the BG is disabled, when the whole display is disabled, and when the left BG row is clipped.

VS Unisystem Palette

The VS Unisystem arcade machine uses a different color palette than NES/Famicom consoles. It seems to support 64 colors, further info is not available?

PPU Cartridge Bus Note

When software accesses palette RAM via Port 2006h/2007h, the palette address accessed actually does show up on the PPU address bus, but the PPU's /RD and /WR signals are not activated (palette memory is located inside of the PPU chip, not on external VROM/VROM chips).

PPU Dimensions & Timings

NTSC/PAL Timings

Item	NTSC	PAL
Video Clock	21.47727MHz	26.601712MHz
CPU Clock	1.7897725MHz	1.7734474MHz
Clock Divider	CPU=Video/12	CPU=Video/15
Cycles/Scanline	113.66; 1364/12	106.53; 1598/15
Total Scanlines	262 (240+22)	312 (240+72)
Frame Rate	60.098Hz	53.355Hz

Visible Screen Resolution

The logical screen resolution processed by the PPU is 256x240 pixels. However the visible screen resolution is somewhat smaller, due to improper blanking periods, and eventually due to exceeding the physical dimensions of (NTSC) displays.

On PAL hardware, the upper 1 scanline, the left 2 pixels, and the right 2 pixels are invisible (displayed as black border). On NTSC hardware, the upper 8 scanlines, and the lower 8 scanlines are often invisible (224 lines visible), eventually some NTSC screens are hiding only the upper 3 scanlines (237 lines visible).

To be compatible with all types of displays, output valid 256x240 pixels, but have the relevant information in the <middle> 240x224 pixels (30x28 tiles) only.

Synchronizing Software with the Cathode Ray Beam

The PPU sets the VBlank flag in PPU Status Register (and optionally produces an NMI) once per frame. It doesn't support scanline interrupts, or a current scanline number register, which is making it a bit difficult to access PPU registers at specific Cathode Ray Beam locations (for example, to split the the screen into two sections with different colors or scroll offsets). However, a couple of methods could be used for that purpose:

- 1) Delay Loops synchronized with NMI (badly wasting CPU time) or using meaningful code with fixed non-conditional execution time instead delays.
- 2) Producing a "Sprite 0 Hit", or a "More Than 8 Sprites Per Scanline" situation at specific screen location (which sets corresponding flag in PPU Status Register, one cannot reset the flag manually, so either works only once per frame)
- 3) Using PCM Sound IRQs as Timer (synchronized with NMI)
- 4) Using external Timers (contained in some Cartridge Mappers)

The APU also contains a so-called "Frame" counter - that counter is NOT physically synchronized with the PPU, and it doesn't even match the exact number of clock cycles per frame. There's a limited chance that one could program it to produce an IRQ at a specific screen location (and to resynchronize it for the next

frame).

More detailed Timing Info

[PPU 2C02 Timings](#)

PPU 2C02 Timings

PPU base timing - NTSC

the 21.48 MHz signal is divided by 4 to get 5.37 MHz, and is used as the smallest unit of timing in the PPU. All following references to PPU clock cycle (abbr. "cc") timing in this document will be in respect to this timing base, unless otherwise indicated.

- Pixels are rendered at the same rate as the base PPU clock.
In other words, 1 clock cycle= 1 pixel.
- One frame consists of 262 scanlines.
This equals 341*262 PPU cc's per frame (divide by 3 for # of CPU cc's).
- 341 PPU cc's make up the time of a typical scanline (or 341/3 CPU cc's).

All PPU memory access cycles are 2 clocks long, and can be made back-to-back (typically done during rendering). Here's how the access breaks down:

At the beginning of the access cycle, PPU address lines 8..13 are updated with the target address. This data remains here until the next time an access cycle occurs.

Miscellaneous PPU info

- Reading from \$2002 clears the vblank flag (bit 7), and resets the internal \$2005/6 flip-flop. Writes here have no effect.
- \$2002.5 and \$2002.6 after being set, stay that way for the first 20 scanlines of the new frame, relative to the VINT.
- Pin /VBL on the 2C02 is the logical NAND between 2002.7 and 2000.7.

Frame rendering details

The following describes the PPU's status during all 262 scanlines of a frame. Any scanlines where work is done (like image rendering), consists of the steps which will be described in the next section.

0..19: Starting at the instant the VINT flag is pulled down (when a NMI is generated), 20 scanlines make up the period of time on the PPU which I like to call the VINT period. During this time, the PPU makes no access to it's external memory (i.e. name / pattern tables, etc.).

20: After 20 scanlines worth of time go by (since the VINT flag was set), the PPU starts to render scanlines. This first scanline is a dummy one; although it will access it's external memory in the same sequence it would for drawing a valid scanline, no on-screen pixels are rendered during this time, making the fetched background data immaterial. Both horizontal *and* vertical scroll counters are updated (presumably) at cc offset 256 in this scanline. Other than that, the operation of this scanline is identical to any other. The primary reason this scanline exists is to start the object render pipeline, since it takes 256 cc's worth of time to determine which objects are in range or not for any particular scanline.

21..260: after rendering 1 dummy scanline, the PPU starts to render the actual data to be displayed on the screen. This is done for 240 scanlines, of course.

261: after the very last rendered scanline finishes, the PPU does nothing for 1 scanline (i.e. the programmer gets screwed out of perfectly good VINT time). When this scanline finishes, the VINT flag is set, and the process of drawing lines starts all over again.

Scanline rendering details

As explained before, external PPU memory can be accessed every 2 cc's. With 341 cc's per scanline, this gives the PPU enough time to make 170 memory accesses per scanline (and it uses all of them!). After the 170th fetch, the PPU does nothing for 1 clock cycle. Remember that a single pixel is rendered every clock cycle.

Note that the PPU fetches an attribute table byte for every 8 sequential horizontal pixels it draws. This essentially limits the PPU's color area (the area of pixels which are forced to use the same 3-color palette) to only 8 horizontally sequential pixels.

Memory fetch phase 1 thru 128 - BG Fetch

Fetches 4x32 bytes; one Name Table entry, one Attribute Table entry, and two Pattern Table bytes; for 3rd..34th tile in scanline (33th tile may be parts visible if BG scrolled, 34th is never visible). From the time of the Name Table fetch, it takes (16-n) clock cycles until the first pixel of the fetched tile is processed (drawn on screen, unless being covered by a Sprite-pixel, and eventually setting the Hit Flag) (n=0..7, horizontal scroll offset).

Simultaneously with above BG fetch, the PPU pre-processes SPR-RAM for the NEXT scanline by searching for Sprite Y-coordinates that are visible in that scanline, only the first eight matches will be recursed, if the search finds additional matching entries then bit5 of \$2002 will get set to indicate that one or more entries have been ignored.

Memory fetch phase 129 thru 160 - Sprite Fetch

Fetches 4x8 bytes; two dummy Name Table entries, and two Pattern Table bytes; for 1st..8th sprite in NEXT scanline (fetches dummy patterns if the scanline contains less than 8 sprites).

Memory fetch phase 161 thru 168 - BG Fetch

Fetches 4x2 bytes; one Name Table entry, one Attribute Table entry, and two Pattern Table bytes; for 1st..2nd tile in NEXT scanline.

Memory fetch phase 169 thru 170 (and a half) - Padding

Fetches 2 bytes; two dummy reads from the Name Table address of the 3rd tile in next scanline. After that fetches, the PPU rests for one "dead" cycle here (or the equivalent of 1/2 memory access cycle) before repeating the whole pixel/scanline rendering process. Scanline 20 is the only scanline that has variable length, on every odd frame, this scanline is only 340 cycles (the dead cycle at the end is removed). This is done to cause a shift in the NTSC colorburst phase.

Audio Processing Unit (APU)

[APU Channel 1-4 Register 0 \(Volume/Decay\)](#)

[APU Channel 1-4 Register 1 \(Sweep\)](#)

[APU Channel 1-4 Register 2 \(Frequency\)](#)

[APU Channel 1-4 Register 3 \(Length\)](#)

[APU Channel 5 - DMC Sound](#)

[APU Control and Status Registers](#)

[APU 4-bit DAC](#)

[APU Various](#)

[APU External Sound Channels](#)

[Controllers - Microphone](#)

Based on "2A03 technical reference by Brad Taylor" (1st release, April 2004).

APU Channel 1-4 Register 0 (Volume/Decay)

4000h - APU Volume/Decay Channel 1 (Rectangle)

4004h - APU Volume/Decay Channel 2 (Rectangle)

400Ch - APU Volume/Decay Channel 4 (Noise)

0-3 Volume / Envelope decay rate
When Bit4=1: Volume (0=Silent/None..F=Loud/Max)
When Bit4=0: Envelope decay rate, NTSC=240Hz/(N+1), PAL=192Hz/(N+1)

4 Envelope decay disable (0=Envelope/Decay, 1=Fixed Volume)

5 Length counter clock disable / Envelope decay looping enable
When Bit4=1: length counter clock disable
When Bit4=0: envelope decay looping enable
0: Disable Looping, stay at 0 on end of decay [_____]
1: Enable Looping, restart decay at F [\\\\\\\]
(Does this still affect Length counter clock disable ?)

6-7 Duty cycle type (unused on noise channel)

0	[--_____]	12.5%	Whereas,
1	[----_____]	25.0%	[_] = LOW (zero) (0)
2	[-----_____]	50.0%	[-] = HIGH (volume/decay) (0..F)
3	[-----_____]	75.0%	Noise randomly outputs LOW or HIGH

The Duty Cycle counter is reset when the length counter of the same channel is written to (via \$4003/\$4007).

Initial Decay Volume:

Only a write out to \$4003/\$4007/\$400F will reset the current envelope decay counter to a known state (to \$F, the maximum volume level) for the appropriate channel's envelope decay hardware. Otherwise, the envelope decay counter is always counting down (by 1) at the frequency currently contained in the volume / envelope decay rate bits (even when envelope decays are disabled by setting bit 4), except when the envelope decay counter contains a value of 0, and envelope decay looping (bit 5) is disabled (0).

4008h - APU Linear Counter Channel 3 (Triangle)

0-6 linear counter load register
7 length counter clock disable / linear counter start

Linear counter incremented-or-decremented? at NTSC=240Hz, PAL=192Hz, same purpose, but higher resolution than length counter, so Triangle channel is ALWAYS using either linear-counter or length-counter, and cannot be used with both counters disabled?

The Triangle channel does not have a variable volume, nor variable duty cycle. Instead, it produces the following fixed 32-step output level stream:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, F, E, D, C, B, A, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

On 2A03 reset, the stream starts at 0. The stream will be halted (at the current position) whenever the Triangle channel's length or linear counter contains a count of 0, and will continue at whatever old position when it is restarted.

APU Channel 1-4 Register 1 (Sweep)

4001h - APU Sweep Channel 1 (Rectangle)

4005h - APU Sweep Channel 2 (Rectangle)

0-2 Sweep right shift amount (S=0..7)
3 Sweep Direction (0=[+]Increase, 1=[-]Decrease)
4-6 Sweep update rate (N=0..7), NTSC=120Hz/(N+1), PAL=96Hz/(N+1)
7 Sweep enable (0=Disable, 1=Enable)

At specified Update Rate, the 11bit Wavelength will be modified as such:

Wavelength = Wavelength +/- (Wavelength SHR S)
(For Channel 1 Decrease only: minus an additional 1)
(I.e. in Decrease mode: Channel 1 uses NOT, Channel 2 uses NEG)

Wavelength register will be updated only if all 3 of these conditions are met:

Bit 7 is set (sweeping enabled)
The shift value (which is S in the formula) does not equal to 0
The channel's length counter contains a non-zero value

Sweep end: The channel gets silenced, and sweep clock is halted, when:

- 1) current 11bit wavelength value is less than 008h
- 2) new 11bit wavelength would become greater than 7FFh

Note that these conditions pertain regardless of any sweep refresh rate values, or if sweeping is enabled/disabled (via Bit7).

4009h - APU N/A Channel 3 (Triangle)

400Dh - APU N/A Channel 4 (Noise)

0-7 Unused (No Sweep support for these channels)

APU Channel 1-4 Register 2 (Frequency)

4002h - APU Frequency Channel 1 (Rectangle)

4006h - APU Frequency Channel 2 (Rectangle)

400Ah - APU Frequency Channel 3 (Triangle)

0-7 Lower 8 bits of wavelength (upper 3 bits in Register 3)

$F = 1.79\text{MHz}/(N+1)/16$ for Rectangle channels

$F = 1.79\text{MHz}/(N+1)/32$ for Triangle channel

400Eh - APU Frequency Channel 4 (Noise)

0-3 Noise frequency, $F=1.79\text{MHz}/2/(N+1)$
Value 0..F corresponds to following 11bit clock cycle value:
N=002,004,008,010,020,030,040,050,065,07F,0BE,0FE,17D,1FC,3F9,7F2
4-6 Unused
7 Random number type generation (0=32767 bits, 1=93 bits)

The random number generator consists of a 15bit shift register. The MSB (Bit14) is output/inverted (1=Low/Zero, 0=High/Decay/Volume). At the specified frequency, Bit14 is XORed with Bit13 (32767-bit mode) or with Bit8 (93-bit mode), the register is then shifted to the left, with the result of the XOR operation shifted-in to Bit0.

On 2A03 reset, this shift register is loaded with a value of 1.

Not sure if it is reset when switching from 32767-bit mode to 93-bit mode? If it isn't reset then 93-bit mode will act unstable: produce different 93-bit patterns, or even a 31-bit pattern, depending on old shift register content.

APU Channel 1-4 Register 3 (Length)

4003h - APU Length Channel 1 (Rectangle)

4007h - APU Length Channel 2 (Rectangle)

400Bh - APU Length Channel 3 (Triangle)

400Fh - APU Length Channel 4 (Noise)

Writing to the length registers restarts the length (obviously), and also restarts the duty cycle (channel 1,2

only), and restarts the decay volume (channel 1,2,4 only).

- 2-0 Upper 3 bits of wavelength (unused on noise channel)
- 7-3 Length counter load register (5bit value, see below)

The above 5bit value is translated to the actual 7bit counter value as such:

- Bit3=0 and Bit7=0 (Dividers matched for use with PAL/50Hz)
 - Bit6-4 (0..7 = 05h,0Ah,14h,28h,50h,1Eh,07h,0Dh)
- Bit3=0 and Bit7=1 (Dividers matched for use with NTSC/60Hz)
 - Bit6-4 (0..7 = 06h,0Ch,18h,30h,60h,24h,08h,10h)
- Bit3=1 (General Fixed Dividers)
 - Bit7-4 (0..F = 7Fh,01h..0Fh)

The 7bit counter value is decremented once per frame (PAL=48Hz, or NTSC=60Hz) the counter and sound output are stopped when reaching a value of zero. The counter can be paused (and restarted at current location) by Length Counter Clock Disabled bit in Register 0.

APU Channel 5 - DMC Sound

4010h - DMC Play mode and DMA frequency

- 7 IRQ Enable, when Length=0 AND Loop=Disabled (0=Disable, 1=Enable)
DMC IRQs can be acknowledged by writing 0 to Bit7 of 4010h, or by writing any value to 4015h
- 6 Loop when reaching Length=0 (0=Stop, 1=Loop)
In looped mode, the sample block is restarted by reloading the DMA Start Address and Length values, IRQs are not generated.
- 5-4 Appear to be unused
- 3-0 DMC frequency control. For values 0-F, number of cycles/samplebyte are:
D60, BE0, AA0, A00, 8F0, 7F0, 710, 6B0, 5F0, 500, 470, 400, 350, 2A8, 240, 1B0

Cycles/samplebit is 8 times less (faster) than cycles/samplebyte.

4011h - DMC Delta counter load register

- 7 Appears to be unused
- 6-1 MSBs of 7bit DAC (6bit "Delta Counter")
- 0 LSB of 7bit DAC

Used to initialize the Delta Counter (for DMC usage), or to output 7bit data directly (for PCM usage). Another use of this register has been to somewhat control the volume of the Triangle & Noise sound channel outputs. Please see NESSOUND.TXT for more information.

4012h - DMC address load register

Specifies the DMA Start Address. The Start Address is loaded to the actual DMA pointer, when the DMC is activated from an inactive state, or when restarting looped playback.

- 7-0 DMA Start Address for DMC (Address = C000h+N*40h)

The DMA pointer is 15 bits in size, and wraps from FFFFh to 8000h (not C000h).

4013h - DMC length register

- 7-0 DMA Length DMC (Length = N*10h+1 Bytes = N*80h+8 Bits)

When it arrives at 0, the DMC will take action(s) based on the 2 MSB of \$4010. This counter will be loaded with the current calculated address value of \$4013 when the DMC is activated from an inactive state.

Usage as Delta Modulation Channel (DMC) with Direct Memory Access (DMA)

This method uses 1-bit samples, processed at the specified sample-bit-rate,

Every eight sample-bits, the DMC will halt the CPU for 2 clock cycles to retrieve the next sample-byte per
0 = Decrement Delta counter by 1 (unless result would be less than 0)

DMA, each sample byte is processed bit-by-bit (LSB first).

Usage as Pulse Code Modulation (PCM) Channel

Alternately, 7bit samples can be written directly to the DAC.

Advantages are that all 7bit can be used (instead only the upper six Delta bits), and that the DAC can be directly changed from one value to any other value (which would take up to 64 increment/decrement steps in DMC mode).

Disadvantages are that it requires exact software timings and more CPU load than the DMA method.

No idea if it is required to "enable" the DMC channel (eg. by outputting a looped dummy 55h sample byte) in order to use PCM ?

On 2A03 reset, the DMC's IRQ flag is cleared (disabled), and the [DMC] channel is disabled. On 2A03 reset, all 7 used bits of \$4011 are reset to 0.

APU Control and Status Registers

4015h - DMC/IRQ/length counter status/channel enable register

0	Status/Enable rectangle wave channel 1
1	Status/Enable rectangle wave channel 2
2	Status/Enable triangle wave channel 3
3	Status/Enable noise channel 4
4	Status/Enable DMC channel 5
5	Not used (returns garbage on reading)
6	Frame IRQ status (active when set)
7	DMC's IRQ status (active when set)

Reading Bit4-0 returns 0 for a zero count status in the length counter (channel's sound is disabled), and 1 for a non-zero status. Reading Bit7-6 returns IRQ status flags.

Writing to Bit4-0: Writing 0 forces to disable the channel, it will get stopped and become silent (as if its length counter has reached 0), writing 1 de-activates the forced-stop (without changing the length counter, playback continues at whatever value have been in the length counter).

Writing to Bit7-5: Unknown.

DMC IRQs are acknowledged by WRITING any value to 4015h.

Frame IRQs are acknowledged by READING from 4015h.

Note that all 5 writable bits in 4015h will be set to 0 upon 2A03 reset.

4017h - APU Low frequency timer control (W)

Any write to \$4017 resets both the frame counter, and the clock divider.

Sometimes, games will write to this register in order to synchronize the sound hardware's internal timing, to the sound routine's timing (usually tied into the NMI code). The frame IRQ frequency is slightly smaller than the PPU's vertical retrace frequency, so you can see why games would desire this synchronization.

bit6: Frame IRQ Disable (0=Enable Frame IRQ, 1=Disable Frame IRQ)
bit7: Frame Rate Select (0=NTSC=60Hz=240Hz/4, 1=PAL=48Hz=240Hz/5)

On 2A03 reset, Bit6-7 will be cleared. That means that Frame IRQs are enabled by default (though usually not executed since the CPU's IRQ-Disable-Flag is set on reset).

Frame IRQs are acknowledged by reading from 4015h.

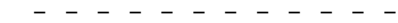
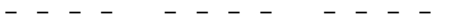


Frame Counter

Several audio timings are referred to as "Frames" and "PAL" and "NTSC",

! These timings are NOT physically related to actual PPU VBlank/NMI timings !

The Audio "Frame" counter can be switched into "PAL" or "NTSC" mode by software - regardless of whether the game does run on a PAL/NTSC console. Audio-frames may be (more or less) synchronized

with video-frames by choosing PAL/NTSC audio-mode matching to PAL/NTSC console type respectively. The frame counter is based on a 240Hz signal which is gained from dividing the 1.78Mhz PHI2 clock edges (2*1.78M edges/second) by 14915. In PAL Mode (4017h Bit7=1), the 240Hz signal is divided by 1.25 (by simply leaving out each fifth clock pulse), resulting in a somewhat dirty 192Hz signal. This PAL/NTSC adjustment mechanism counts through 4 or 5 steps, producing output as such:

0/NTSC: 4, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3		1/PAL: 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4
240Hz: 		192Hz: 
120Hz: 		96Hz: 
60Hz: (above somehow div by 2)		48Hz: (above somehow divided by 2)

Frame Counter is reset on writing to 4017h, and does then restart sequences as shown above (in NTSC mode starting with a skipped step, whilst directly starting with a non-skipped step in PAL mode). Linear counter (triangle) and envelope decay counters (rectangle/noise) are clocked by 240Hz/192Hz. Frequency sweep (rectangle) clocked by 120Hz/96Hz. Length counters (all channels) and Frame IRQ clocked by 60Hz/48Hz.

4014h - SPR-RAM DMA

Sprite RAM DMA Function contained in 2A03 chip. See PPU description.

4016h - Write

Three bit general purpose output latch contained in 2A03 chip. Used to strobe joysticks. See Controllers chapter for more info.

4016h/4017h - Read

The 2A03 chip does not actually contain read-able registers at these addresses, however, it does output read-request signals for these addresses, which are used to activate on-board joystick inputs. See Controllers chapter for more info.

Note: 4015h is the only R/W register in the 4000h-4017h area, all other registers in this area are write-only, and do not respond to read cycles (except for the external read-able 4016h/4017h registers).

APU 4-bit DAC

Channel 1-4 are (each) using a standard 4-bit DAC with 16 steps of output voltage resolution. On the 2A03, rectangle wave 1 & 2 are mixed together, and are available via pin 1. Triangle, noise, and DMC are available on pin 2.

Signals are then merged via 20KOhm (pin 1) and 12KOhm (pin2), respectively, rectangle channels have different output levels than equivalent volume settings on triangle/noise channels?

The output waveforms have some linear asymmetry (the desired output voltage would increase on a linear scale, the actual outputted voltage increases less and less each step).

The side effect of this is that the DMC's 7-bit DAC port (\$4011) is able to indirectly control the volume (somewhat) of both triangle & noise channels. When \$4011=0, triangle & noise volume outputs are at maximum. When \$4011=7F, triangle & noise channel outputs operate at only 57% total volume. A few games actually take advantage of this "volume" feature, and write values to \$4011 in order to regulate the amplitude of the triangle wave channel's output.

Forced Zero Volume:

When hardware in the channel wants to disable it's sound output (like the length counter, or sweep unit).

APU Various

After 2A03 reset, the sound channels are unavailable for playback during the first 2048 CPU clocks.

The rectangle channel(s) frequency in the range of 54.6 Hz to 12.4 KHz.

The triangle wave channel range of 27.3 Hz to 55.9 KHz.

The random wavelength channel range anywhere from 29.3 Hz to 447 KHz.

RP2A03E quirk

I have been informed that revisions of the 2A03 before "F" actually lacked support for the 93-bit looped noise playback mode. While the Famicom's 2A03 went through 4 revisions (E..H), I think that only one was ever used for the front loading NES: "G". Other differences between 2A03 revisions are unknown. Is that Quirk True and Confirmed ?

APU External Sound Channels

The Famicom 60-pin cartridge slot includes a SND_IN pin, allowing external sound controllers (as included in some Cartridge Mappers, and in Famicom Disk System) to produce additional sound channels which are merged with the normal APU channels:

[Mapper 5: MMC5 - BANKING, IRQ, SOUND, VIDEO, MULTIPLY, etc.](#)

[Mapper 19: Namcot 106 - PRG/8K, VROM/1K/VRAM, IRQ, SOUND](#)

[Mapper 20: Disk System - PRG RAM, BIOS, DISK, IRQ, SOUND](#)

[Mapper 24: Konami VRC6A - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND](#)

[Mapper 26: Konami VRC6B - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND](#)

[Mapper 85: Konami VRC7A/B - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND](#)

However, the NES 72-pin cartridge slot DOES NOT include a SND_IN pin, even though it does have more (more or less unused) pins than Famicom.

Controllers

[Controllers - I/O Ports](#)

[Controllers - Pin-Outs](#)

[Controllers - Joypads](#)

[Controllers - Zapper](#)

[Controllers - Paddles](#)

[Controllers - Keyboard](#)

[Controllers - Power Pad](#)

[Controllers - Microphone](#)

[Controllers - Reset Button](#)

[Controllers - Arcade Machines](#)

R.O.B (Robotic Operating Buddy), a plastic robot used by Gyromite and Stack-up.

Controllers - I/O Ports


```

4 In port1-D4 (zapper button) -----
5 In port1-D3 (zapper light)
6 In port1-D2
7 In port1-D1 (joystick 4 serial input) (paddle ADC serial input)
8 In port1-D0 (joystick 2 serial input)
9 Out port1-CLK (joystick 2+4 clock read)
10 Out OUT2
11 Out OUT1
12 Out OUT0 (joystick 1+2+3+4 start)
13 In port0-D1 (joystick 3 serial input) (paddle button input)
14 Out port0-CLK (joystick 1+3 clock read)
15 Out +5V

```

Used to connect a 3rd and 4th joystick, and various other expansion hardware.
Keep in mind that older Famicom Joypads cannot be disconnected, so the input at Pin 8 may be disturbed by joystick 2 signals.

Note: Joypads/PowerPads/etc are normally using standard 4021 parallel-in serial-out shift registers.

Controllers - Joypads

Joypads (or Joysticks)

Each joystick includes an 8bit shift register, set Port 4016h/Bit0=1 to reload the button states into the shift registers of both joypads, then reset Port 4016h/Bit0=0 to disable the shift reload (otherwise all further reads would be stuck to the 1st bit, ie. Button A). Joypad data can be then read from bit 0 of 4016h (joypad 1) and/or bit 0 of 4017h (joypad 2) as serial bitstream of 8bit length, ordered as follows:

```
A, B, SELECT, START, UP, DOWN, LEFT, RIGHT
```

The console automatically sends a clock pulse to the Joypad 1 shift register after each read from 4016h (and to joystick 2 after read from 4017h). There are no timing restrictions, joypads can be handled as fast, or as slow, as desired. Received bits are 1=LOW=Pressed/Moved, 0=HIGH=Released. 1-Bits (LOW) are returned after the 8th bit has been received. 0-Bits (HIGH) are returned if no controller connected to the console.

Note that older Famicom controllers include Select & Start buttons only on joystick 1 - joystick 2 probably returns unused dummy bits instead.

NES Four-player devices (Satellite and Four Score)

Used by Tengen's "Gauntlet II", and Nintendo's "RC Pro Am 2". The device is connected to both of the consoles two controller ports, and up to four controllers can be connected to the device.

The device is accessed much like normal joypads, except that the returned bitstream consist of 24 bits instead of normal 8 bits:

```

write "1-then-0" to (4016h) (that only once, for all 24 bits)
read 1st 8 bits: controller 1 (4016h) / controller 2 (4017h) (as normal)
read 2nd 8 bits: controller 3 (4016h) / controller 4 (4017h) (new ports)
read 3rd 8 bits: 0,0,0,1,0,0,0,0 (4016h) / 0,0,1,0,0,0,0,0 (4017h) (ID codes)

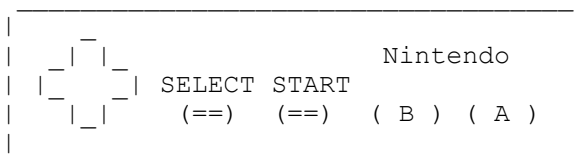
```

The ID codes can be used to detect if the 4-player adapter is connected (used by RC Pro Am 2, not used by Gauntlet II). Otherwise the ID field typically contains all ones (normal/single controller), or all zeros (no controller connected at all).

Famicom Four-player device (Two extra joypads at Expansion Port)

Older Famicom consoles have 2 "built-in" joypads, additional 2 joypads can be connected to the expansion port. Used by Nekketsu Kakutou Densetsu? The procedure for reading Famicom 4-player data is similar as for 2-player data: As normal, write "1-then-0" to 4016h, then read 8 times from 4016h, that simultaneously receives data for two pads, Bit 0 for joystick 1, and additionally Bit 1 for joystick 3. Respectively, Bit 0 and 1 of 4017h are for pad 2 and 4.

Joypad Layout



Jump and Run Games conventionally use A=Jump, B=Fire.

Controllers - Zapper

Zapper (Light Gun) Ports / Connection

Zapper state can be obtained by reading Bit4-5 of Port 4017h and/or 4016h.

```

Bit4  Trigger state of the gun (0=Released, 1=Pulled/Pressed)
Bit3  State of the gun sight   (0=None, 1=Light detected)

```

Famicom Zapper connected to Famicom Expansion Port (Inputs at 4017h).

NES Zappers connected to 1st and/or 2nd Joypad Port (Inputs at 4016h, 4017h).

Many NES games may default to use 2nd Joypad Port because Famicom uses 4017h.

Control Methods

The light detection flag gets set when sensing light emission from the display, ie. when the cathode ray beam outputs a bright color (preferably white) at the location where the gun is pointed to.

Most video controllers are latching the current cathode ray beam coordinates at the time when the light detection flag gets set - that's not supported by the NES/Famicom video controller - it could be eventually implemented by software, ie. by counting the number of clock cycles between vblank and light detection. Otherwise, the following trick can be used: Output a black picture, with a white field at the desired target location, wait for 1-2 frames, then check the light detection flag to see if the zapper was pointed to the target area or not. The downside is that the normal picture cannot be displayed during that time, so one should check the zapper position only when necessary, ie. typically only at the moment when the trigger gets pulled.

Unknown if/when/how the light detection flag gets reset?

Controllers - Paddles

The Paddle consists of a potentiometer (can be turned left/right by about 270 degrees), and a push button. The potentiometer is connected to an Analogue to Digital Converter, that ADC data is then sent to the console in serial via shift register. Taito's "Arkanoid" uses a paddle as it's primary controller.

The paddle position is read via D1 of \$4017; the read data is inverted (0=1, 1=0). The first value read is the MSB, and the 8th value read is (obviously) the LSB. Valid value ranges are 98 to 242, where 98 represents the paddle being turned completely counter-clockwise.

For example, if %01101011 is read, the value would be NOT'd, making %10010100 which is 146. The paddle also contains one button, which is read via D1 of \$4016. A value of 1 specifies that the button is being pressed.

Is that Famicom-Expansion-Port only ?

Controllers - Keyboard

Keyboard with 72 Keys, and tape read/write port, connected to 15-pin Famicom Expansion port. Used by Famicom BASIC. Also, the Study and Game 32-in-1 and Education 18-in-1 cartridges use an identical protocol, but with different keyboard matrix.

Keyboard Access Pseudo Code

```
[4016h]=05h:WAIT(16clks)           ;reset (force row 0)
FOR i=0 TO 8                         ;loop 9 rows
  [4016h]=04h:WAIT(56clks)         ;request LSB of NEXT row
  Row[i]=((([4017h] SHR 1) AND 0Fh) ;read LSB
  [4016h]=06h:WAIT(56clks)         ;request MSB of SAME row
  Row[i]=((([4017h] SHL 3) AND F0h)+Row[i] ;read MSB
NEXT                                  ;loop next
```

Column 0-7 are then in Bit0-7 of each row. Bits are 0=Pressed, 1=Released (unlike for most other NES/Famicom controllers, which are 1=Pressed).

When reading more than 9 rows, the 10th read (row 9) returns garbage data, and then starts over at row 0.

Famicom Keyboard Matrix

Row	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
0	F8	RETURN	[]	KANA	R-SHFT	\(Yen)	STOP
1	F7	@	:	;	/	-	-	^
2	F6	O	L	K	.	,	P	0
3	F5	I	U	J	M	N	9	8
4	F4	Y	G	H	B	V	7	6
5	F3	T	R	D	F	C	5	4
6	F2	W	S	A	X	Z	E	3
7	F1	ESC	Q	CTRL	L-SHFT	GRPH	1	2
8	CLR	UP	RIGHT	LEFT	DOWN	SPACE	DEL	INS

Famicom Keyboard Layout

	F1	F2	F3	F4	F5	F6	F7	F8	
	1	2	3	4	5	6	7	8	
	9	0	-	^	\	STOP			
	ESC	Q	W	E	R	T	Y	U	
	I	O	P	@	[ENTER	CLR	INS	
	DEL	CTRL	A	S	D	F	G	H	
	J	K	L	;	:]	KANA	UP	
	SHIFT	Z	X	C	V	B	N	M	
	,	.	/	_	SHIFT	LEFT	RIGHT		
	GRPH	SPACE				DOWN			

32-in-1 Study and Game / Education Keyboard Matrix

Row	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
0	4	G	F	C	F2	E	5	V
1	2	D	S	END	F1	W	3	X
2	INS	BS	PGDN	RIGHT	F8	PGUP	ESC	HOME
3	9	I	L	,	F5	O	0	.
4]	ENTER	UP	LEFT	F7	[\	DOWN
5	Q	CAPS	Z	Pa	ESC	A	1	CTRL
6	7	Y	K	M	F4	U	8	J
7	-	;	'	/	F6	P	=	SHIFT
8	T	H	N	SPACE	F3	R	6	B

The 32-in-1 menu also checks Bit4 in Row 9, if that bit is zero then it does additionally read row 0Ah..0Ch. Aside from the menu, most or all games in the 32-in-1 cartridge don't seem to use that extra rows though.

32-in-1 Study and Game Keyboard Layout (as shown in Typing School I)

	ESC	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	Pa.	Br	Nu	Re.	
	~	1	2	3	4	5	6	7	8	9	0	-	+	BS	HOME			

```
| TAB Q W E R T Y U I O P [ ] \ END |
| CAPS A S D F G H J K L ; ' ENTER PGUP |
| SHIFT Z X C V B N M , . / SHIFT UP PGDN |
| _###_CTRL_ALT_##_[____SPACE____]_ALT_INS_DEL_LT_DN_RIGH_|
```

Keyboard I/O Signals

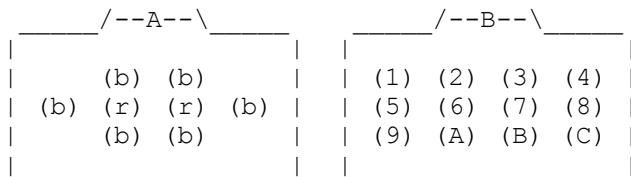
```
OUT.0      Keyboard Strobe/Reset (0=Normal, 1=Initialize)
OUT.1      Keyboard Clock (0=LSB, 1=MSB) (1-to-0=Next Row)
OUT.2      Tape Output? (Should be 1 when accessing Keyboard)
PORT0-1    Tape Input
PORT1-4..1 Keyboard Input Bit3..0 (either MSB or LSB of current row)
```

Controllers - Power Pad

The Power Pad or Family Trainer is a device made by Bandai/Nintendo (1987/1988) which serves as an "exercising fun center" for the whole family. That is, a large (1x1 meters) vinyl mat with 12 touch-sensitive areas, or actually step-sensitive since it's intended to be put on the floor. The mat has two sides, with different patterns drawn on each side.

Used by Athletic World, World Class Track Meet (originally called Stadium Events), Super Team Games, Street Cop, Dance Aerobics, Short Order/Eggsplode.

Power Pad Layout (Side A and Side B)



Side A has 2 red fields, 6 blue fields, and 4 hidden fields.

Side B has 12 fields, numbered 1..12 (referred to as 1-C hex, in this document), the left fields (1,2,5,6,9,A) are blue, the other are red.

Power Pad Connection / Access

Famicom version is connected to Expansion Port (inputs at 4017h), NES version can be connected to Joypad Port 1 or 2 (inputs at 4016h or 4017h, or both with two pads). Many NES games may default to use Joypad Port 2, for Famicom compatibility, also Port 1 is often used for menu selection via joypad. To read the button states:

```
Output 1-then-0 to Bit0 of Port 4016h
Read eight times from Port 4017h (and/or 4016h on NES)
```

Each read receives two button states in Bit 3 and 4, in following order:

```
Bit4  4,3,C,8,u,u,u,u (0=Released, 1=Pressed) (u=Unused always 1)
Bit3  2,1,5,9,6,A,B,7 (0=Released, 1=Pressed)
```

Whereas, 1-C (hex) are button numbers 1-12 on Side B, or equivalent buttons on Side A (horizontally mirrored, of course).

Controllers - Microphone

Microphone

On older Famicoms, the second control pad (that without Start and Select buttons) has a microphone with

volume control built-in. The signal goes to Bit 2 of Port 4016h (simple 1bit input, not an analogue ADC-converted input).

The signal is also merged with the PPU's sound output signals, as such, allowing to use the television set/speaker as amplifier/megaphone.

Controllers - Reset Button

Reset Button

As an additional "control" the console is equipped with a reset button, which is grounding the 2A03s /RST pin (resets CPU and APU).

On NES consoles (not on Famicom consoles), the reset signal is also connected to PPU /SYNC input, causing the picture to be disabled during reset (the PPU registers are left unchanged though). For curiosity, the NES reset signal is also connected to the power LED, the LED goes off when pressing Reset (or when the lockout chip generates a reset).

Anyways, RAM and VRAM is left unaffected, so that the program may recover from reset by invoking a warmboot rather than complete coldboot. As simple example, it may, if desired, preserve high score values, etc.

The cartridge bus doesn't include a reset signal, so mapper registers would be usually left unaffected as well - unless any mappers figure out any ridiculous ways to detect resets, for example by examining address signals.

The Reset button is also important for some games with battery backed SRAM: The consoles cartridge bus becomes unstable during power-off, so that SRAM content may get overwritten randomly. As workaround, some games prompt the user to hold down the reset button during power-off (eg. Maniac Mansion, MMC1). Other games include mappers that can enable/disable SRAM, and don't need that trick (eg. Kirby's Adventure, MMC3).

Controllers - Arcade Machines

VS Unisystem

Arcade Machine with additional coin-detection and DIP-switch inputs.

```
Port 4016h/Write:
  Bit2      Select 8K VROM bank at PPU 0000h-1FFFh (Mapper 99 games only)
Port 4016h/Read:
  Bit2      Credit Service Button      (0=Released, 1=Service Credit)
  Bit3-4    DIP Switch 1-2             (0=Off, 1=On)
  Bit5-6    Credit Left/Right Coin Slot (0=None, 1=Coin) (Acknowledge via 4020h)
Port 4017h/Read:
  Bit2-7    DIP Switch 3-8             (0=Off, 1=On)
Port 4020h/Write:
  Bit0      Acknowledge Coin Slot Signal (0=Normal, 1=Acknowledge Coin)
```

The controls are working like normal joypads, with different bits assignments, Start/Select are renamed to Button 1-4, and controls for Player 1 and 2 are exchanged:

Read	NES/4016h	VS/4016h	NES/4017h	VS/4017h
1st	Button A (1)	Button A (2)	Button B (2)	Button A (1)
2nd	Button B (1)	Button B (2)	Button B (2)	Button B (1)
3rd	Select (1)	Button 1	Select (2)	Button 2
4th	Start (1)	Button 3	Start (2)	Button 4
5th	Up (1)	Up (2)	Up (2)	Up (1)
6th	Down (1)	Down (2)	Down (2)	Down (1)
7th	Left (1)	Left (2)	Left (2)	Left (1)
8th	Right (1)	Right (2)	Right (2)	Right (1)

Also note that the VS Unisystem uses different palettes than NES/Famicom.

Play Choice 10

Arcade Machine with additional coin-detection, DIP-switch, and game-select inputs. The inputs are controlled by a separate Z80 CPU. More info:

[Nintendo Playchoice 10](#)

Cartridges and Mappers

General Cartridge Info

[Cartridge Info](#)

Mappers (Numbers as used in .NES fileformat)

[Mapper 0: NROM - \(No Mapper\)](#)

[Mapper 1: MMC1 - PRG/32K/16K, VROM/8K/4K, NT](#)

[Mapper 2: UNROM - PRG/16K](#)

[Mapper 3: CNROM - VROM/8K](#)

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

[Mapper 5: MMC5 - BANKING, IRQ, SOUND, VIDEO, MULTIPLY, etc.](#)

[Mapper 6: FFE F4xxx - PRG/16K, VROM/8K, NT, IRQ](#)

[Mapper 7: AOROM - PRG/32K, Name Table Select](#)

[Mapper 8: FFE F3xxx - PRG/32K, VROM/8K, NT, IRQ](#)

[Mapper 9: MMC2 - PRG/24K/8K, VROM/4K, NT, LATCH](#)

[Mapper 10: MMC4 - PRG/16K, VROM/4K, NT, LATCH](#)

[Mapper 11: Color Dreams - PRG/32K, VROM/8K](#)

[Mapper 12: FFE F6xxx - Not specified, NT, IRQ](#)

[Mapper 13: CPROM - 16K VRAM](#)

[Mapper 15: X-in-1 - PRG/32K/16K, NT](#)

[Mapper 16: Bandai - PRG/16K, VROM/1K, IRQ, EPROM](#)

[Mapper 17: FFE F8xxx - PRG/8K, VROM/1K, NT, IRQ](#)

[Mapper 18: Jaleco SS8806 - PRG/8K, VROM/1K, NT, IRQ, EXT](#)

[Mapper 19: Namcot 106 - PRG/8K, VROM/1K/VRAM, IRQ, SOUND](#)

[Mapper 20: Disk System - PRG RAM, BIOS, DISK, IRQ, SOUND](#)

[Mapper 21: Konami VRC4A/VRC4C - PRG/8K, VROM/1K, NT, IRQ](#)

[Mapper 22: Konami VRC2A - PRG/8K, VROM/1K, NT](#)

[Mapper 23: Konami VRC2B/VRC4E - PRG/8K, VROM/1K, NT, \(IRQ\)](#)

[Mapper 24: Konami VRC6A - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND](#)

[Mapper 25: Konami VRC4B/VRC4D - PRG/8K, VROM/1K, NT, IRQ](#)

[Mapper 26: Konami VRC6B - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND](#)

[Mapper 32: Irem G-101 - PRG/8K, VROM/1K, NT](#)

[Mapper 33: Taito TC0190/TC0350 - PRG/8K, VROM/1K/2K, NT, IRQ](#)

[Mapper 34: Nina-1 - PRG/32K, VROM/4K](#)

[Mapper 40: FDS-Port - Lost Levels](#)

[Mapper 41: Caltron 6-in-1](#)

[Mapper 42: FDS-Port - Mario Baby](#)

[Mapper 43: X-in-1](#)

[Mapper 44: 7-in-1 MMC3 Port A001h](#)

[Mapper 45: X-in-1 MMC3 Port 6000hx4](#)

[Mapper 46: 15-in-1 Color Dreams](#)

[Mapper 47: 2-in-1 MMC3 Port 6000h](#)

[Mapper 48: Taito TC190V](#)

[Mapper 49: 4-in-1 MMC3 Port 6xxxh](#)

[Mapper 50: FDS-Port - Alt. Levels](#)

[Mapper 51: 11-in-1](#)

[Mapper 52: 7-in-1 MMC3 Port 6800h with SRAM](#)
[Mapper 56: Pirate SMB3](#)
[Mapper 57: 6-in-1](#)
[Mapper 58: X-in-1](#)
[Mapper 61: 20-in-1](#)
[Mapper 62: X-in-1](#)
[Mapper 64: Tengen RAMBO-1 - PRG/8K, VROM/2K/1K, NT](#)
[Mapper 65: Irem H-3001 - PRG/8K, VROM/1K, NT, IRQ](#)
[Mapper 66: GNROM - PRG/32K, VROM/8K](#)
[Mapper 67: Sunsoft3 - PRG/16K, VROM/2K, IRQ](#)
[Mapper 68: Sunsoft4 - PRG/16K, VROM/2K, NT-VROM](#)
[Mapper 69: Sunsoft5 FME-7 - PRG/8K, VROM/1K, NT ctrl, SRAM, IRQ](#)
[Mapper 70: Bandai - PRG/16K, VROM/8K, NT](#)
[Mapper 71: Camerica - PRG/16K](#)
[Mapper 72: Jaleco Early Mapper 0 - PRG-LO, VROM/8K](#)
[Mapper 73: Konami VRC3 - PRG/16K, IRQ](#)
[Mapper 74: Whatever MMC3-style](#)
[Mapper 75: Jaleco SS8805/Konami VRC1 - PRG/8K, VROM/4K](#)
[Mapper 76: Namco 109 - PRG/8K, VROM/2K](#)
[Mapper 77: Irem - PRG/32K, VROM/2K, VRAM 6K+2K](#)
[Mapper 78: Irem 74HC161/32 - PRG/16K, VROM/8K](#)
[Mapper 79: AVE Nina-3 - VROM/8K](#)
[Mapper 80: Taito X-005 - PRG/8K, VROM/2K/1K, NT](#)
[Mapper 81: AVE Nina-6](#)
[Mapper 82: Taito X1-17 - PRG/8K, VROM/2K/1K](#)
[Mapper 83: Cony](#)
[Mapper 84: Whatever](#)
[Mapper 85: Konami VRC7A/B - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND](#)
[Mapper 86: Jaleco Early Mapper 2 - PRG/32K, VROM/8K](#)
[Mapper 87: Jaleco/Konami 16K VROM - VROM/8K](#)
[Mapper 88: Namco 118](#)
[Mapper 89: Sunsoft Early - PRG/16K, VROM/8K](#)
[Mapper 90: Pirate MMC5-style](#)
[Mapper 91: HK-SF3 - PRG/8K, VROM/2K, IRQ](#)
[Mapper 92: Jaleco Early Mapper 1 - PRG-HI, VROM/8K](#)
[Mapper 93: 74161/32 - PRG/16K](#)
[Mapper 94: 74161/32 - PRG/16K](#)
[Mapper 95: Namcot MMC3-Style](#)
[Mapper 96: 74161/32](#)
[Mapper 97: Irem - PRG HI](#)
[Mapper 99: VS Unisystem Port 4016h - VROM/8K](#)
[Mapper 100: Whatever](#)
[Mapper 105: X-in-1 MMC1](#)
[Mapper 112: Asder - PRG/8K, VROM/2K/1K](#)
[Mapper 113: Sachen/Hacker/Nina](#)
[Mapper 114: Super Games](#)
[Mapper 115: MMC3 Cart Saint](#)
[Mapper 116: Whatever](#)
[Mapper 117: Future](#)
[Mapper 118: MMC3 with different Name Tables](#)
[Mapper 119: MMC3 TQROM with VROM+VRAM Pattern Tables](#)
[Mapper 122: Whatever](#)
[Mapper 133: Sachen](#)
[Mapper 151: VS Unisystem - PRG/8K, VROM/4K](#)
[Mapper 152: Whatever](#)

[Mapper 160: Same as Mapper 90](#)
[Mapper 161: Same as Mapper 1](#)
[Mapper 180: Nihon Bussan - PRG HI](#)
[Mapper 182: Same as Mapper 114](#)
[Mapper 184: Sunsoft - VROM/4K](#)
[Mapper 185: VROM-disable](#)
Mapper 187: No Info
[Mapper 188: UNROM-reversed](#)
[Mapper 189: MMC3 Variant](#)
[Mapper 222: Dragon Ninja](#)
[Mapper 225: X-in-1](#)
[Mapper 226: X-in-1](#)
[Mapper 227: X-in-1](#)
[Mapper 228: X-in-1 Homebrewn](#)
[Mapper 229: 31-in-1](#)
[Mapper 230: X-in-1 plus Contra](#)
[Mapper 231: 20-in-1](#)
[Mapper 232: 4-in-1 Quattro Camerica](#)
[Mapper 233: X-in-1 plus Reset](#)
[Mapper 234: Maxi-15](#)
[Mapper 240: C&E/Supertone - PRG/32K, VROM/8K](#)
[Mapper 241: X-in-1 Education](#)
[Mapper 242: Waixing - PRG/32K, NT](#)
[Mapper 243: Sachen Poker - PRG/32K, VROM/8K](#)
[Mapper 244: C&E - PRG/32K, VROM/8K](#)
Mapper 245: No Info (seems to be some sort of MMC3 variant)
[Mapper 246: C&E - PRG/8K, VROM/2K, SRAM](#)
Mapper 248: No Info
Mapper 249: No Info
Mapper 250: No Info
Mapper 251: No Info
Mapper 252: No Info
Mapper 254: No Info
[Mapper 255: X-in-1 - \(Same as Mapper 225\)](#)

Cartridge Info

General Info

[Cartridge Overview](#)
[Cartridge ROM-Image File Formats](#)
[Cartridge IRQ Counters](#)
[Cartridge Bus Conflicts](#)
[Cartridge Cicurity Chip \(CIC\) \(Lockout Chip\)](#)
[Cartridge Game Genie](#)
[Cartridge Pin-Outs](#)

Cartridge Overview

Standard Mappers

There are more than hundred different mappers, though most are unimportant, the standard types are Mapper 0,1,2,3,4 for ROM-cartridges. And Mapper 20 for Floppy disks.

Type	Games	Percent
Mapper 0 (NROM)	446	12.5%
Mapper 1 (MMC1)	723	20.3%
Mapper 2 (UNROM)	397	11.2%
Mapper 3 (CNROM)	273	7.7%
Mapper 4 (MMC3)	784	22.1%
Mapper 20 (FDS)	?	x.x%
Other Mappers	932	26.2%
Total	3555	100.0%

Other mapper types are used by less than 2% per type, though together they make up 26.2%.

Non-standard Mappers

Some mappers like MMC5 have been used only in a few newer cartridges. Several third-party companies (Konami, Irem, Jaleco, Bandai, Sunsoft, etc.) have developed their own mappers which are used only for their own games. That mappers may be important to play specific games, though they are often used only by 1-2 titles.

Also, there have been various pirate / multi-game cartridges manufactured, containing modified ROM-images with custom mapper circuits, these mappers are completely unimportant since the original games used standard mappers.

Maximum manufactured ROM Size

The largest single NES game that I know of is Dragonquest 4 / Dragon Warrior 4. It has 1 megabyte of program ROM. Also, the Japanese game Metal Slader Glory has 512K of PRG and 512K of CHR ROM, making it also a full megabyte. Several pirate/unlicensed Famicom games are also pretty large.

Minimum manufactured ROM Size

Although the .NES fileformat deems 16K PRG ROM games as the minimum, there have been some 8K games manufactured, such as Galaxian. Later on, skilled programmers have learned to squeeze better code into even less memory, but nowadays most are probably dead.

Banking Granularity

PRG ROM is usually split into banks of 8K, 16K, or 32K, a few mappers like MMC5 also have smaller 1K SRAM banks. VROM is usually split into banks of 1K, 2K, 4K, or 8K.

Note on Mapper Descriptions

In this document Bank Selections are sometimes described as 4K, or as 4x1K. The linear address for a 4K bank is $(N*4096)$.

The linear address for a 4x1K bank is $((N \text{ AND } (\text{NOT } 3))*1024)$.

Ie. in the latter case, the bank value is specified in 1K-steps, with lower bits ignored, and rounded down to a 4K boundary.

Mapper Reset

In general, mapper registers are uninitialized on reset/power-up, and should be initialized by software; if memory at FFFCh is mappable, then valid reset vectors (and reset handlers) should be contained in all banks.

There is no reset signal available on cartridge bus, possible ways to detect reset are to sense inactivity on A0 or PHI2 lines, to sense reads from the reset vector at FFFCh, or to use power-up capacitors for coldboot detection (though that not for warmboot).

Caution: Several mappers in this document are described to have initial settings on reset or power-up. Most of that info has been taken from other documents, in most cases that is unconfirmed, and probably incorrect. A few mappers seem to be actually containing reset circuits though.

iNES Format (.NES)

This fileformat and mapper-numbers have been designed/assigned by Marat Fayzullin (author of iNES emulator), please contact him if you want to make any changes to the format or numbers. The file header is 16 bytes:

```
00h File ID ('NES',1Ah)
04h Number of 16K PRG-ROM pages
05h Number of 8K CHR-ROM pages (00h=None / VRAM)
06h Cartridge Type LSB
  Bit7-4 Mapper Number (lower 4bits)
  Bit3    1=Four-screen VRAM layout
  Bit2    1=512-byte trainer/patch at 7000h-71FFh
  Bit1    1=Battery-backed SRAM at 6000h-7FFFh, set only if battery-backed
  Bit0    0=Horizontal mirroring, 1=Vertical mirroring
07h Cartridge Type MSB (ignore this and further bytes if Byte 0Fh nonzero)
  Bit7-4 Mapper Number (upper 4bits)
  Bit3-2 Reserved (zero)
  Bit1    1=PC10 game (arcade machine with additional 8K Z80-ROM) (*)
  Bit0    1=VS Unisystem game (arcade machine with different palette)
08h Number of 8K RAM (SRAM?) pages (usually 00h=None-or-not-specified)
09h Reserved (zero)
0Ah Reserved (zero) (sometimes 03h,10h,13h,30h,33h purpose unknown) (*)
0Bh Reserved (zero)
0Ch Reserved (zero)
0Dh Reserved (zero)
0Eh Reserved (zero)
0Fh Nonzero if [07h..0Fh]=GARBAGE, if so, assume [07h..0Fh]=ALL ZERO (*)
```

Followed by 512 byte trainer (if any, see Byte 6, Bit 2, mainly FFE games).

Followed by N*16K PRG-ROM pages (see Byte 4).

Followed by N*8K CHR-ROM pages (if any, see Byte 5).

Followed by 8K Play Choice 10 Z80-ROM (if any, see Byte 7, Bit 1) (*).

Followed by 128 (or 127) bytes title at end of file (ASCII, zero-padded) (*).

Items marked as (*) are regularly used, but not official part of the format.

Many PC10 files declare Z80-ROM as additional VROM bank (instead Byte7/Bit1).

.UNF - Universal NES Image File Format (UNIF) by Tennessee Carmel-Veilleux

A "newer" fileformat dated back to 2000, the relation between iNES mapper numbers and UNIF MAPR names is still undocumented, and of course nobody uses files with .UNF extension. Still, it's having one or two useful features, and may become more popular if somebody dares to fix the MAPR problem, and to rename it from .UNF to .NES extension.

File Header (32 bytes)

```
00h-03h: "UNIF" tag identifier
04h-07h: Revision number ("currently 4, for REV 7b, Revision 6 of UNIF" Huh!)
08h-1Fh: Reserved for future usage
```

The header is followed by whatever chunks, all chunks are optional, and may or may not be included in the file, only the PRG0 one is obviously required. Software may skip any chunks which are uninteresting or unrecognized, each chunk formatted as such:

```
00h-03h: Chunk ID string (4-letter ASCII, described below)
04h-07h: Length of Data Block in bytes (excluding above ID and length entry)
08h... : Data
```

MAPR - Board Name (aka Mapper) (ASCIZ, suggested max: 32 chars)

This uses ASCIZ strings to describe the board names (instead of iNES mapper numbers), it's meant to be more specific than mapper numbers, for example, it's using different names for different MMC1-boards.
<http://www.parodius.com/~veilleux/boardtable.txt>
<http://www.parodius.com/~veilleux/boardnames>

PRG0..PRGF - Binary data of the PRG ROM

CHR0..CHRF - Binary data of the CHR ROM (aka VROM in general)

Normally using only PRG0 (and CHR0, if VROM used).

In rare cases, if the cart contains more than 1 PRG (or CHR) ROM chip, then PRG1-F and CHR1-F may be used for the additional chips.

TVCI - Television Standards Compatability Information (One Byte)

00h 60Hz/NTSC (USA, Japan, etc.)
01h 50Hz/PAL (Germany, etc.)
02h Compatible with both 50Hz and 60Hz refresh rates

CTRL - Controllers used by the cartridge (currently only 1 Byte / 8bit)

Bit0 Regular Joypad
Bit1 Zapper
Bit2 R.O.B
Bit3 Arkanoid Controller (presumably Paddle)
Bit4 Power Pad
Bit5 Four-Score adapter (NES 4-player adapter) (Not Famicom adapter!)
Bit6-7 Reserved

MIRR - Name Table Mirroring (1 Byte)

00h Two-Screen Horizontal Mirroring (Hard Wired)
01h Two-Screen Vertical Mirroring (Hard Wired)
02h Single-Screen BLK0 (Hard Wired)
03h Single-Screen BLK1 (Hard Wired)
04h Four-Screens of VRAM (Hard Wired)
05h Mirroring Controlled By Mapper Hardware

BATR - Battery installed on Board (1 dummy byte)

Presence of this chunk means yes, absence means no.

NAME - Game Title, ASCIZ String

Game Title

READ - Readme/Comments/Notes/Credits

Probably some sort of ASCII text of unspecified formatting

VROR - Allow homebrewn games to over-write VROM (1 dummy byte)

Presence of this chunk means yes, absence means no.

PCK0..PCKF - 32-bit CRCs for PRG0..PRGF blocks (4 bytes, each)

CCK0..CCKF - 32-bit CRCs for CHR0..CHRF blocks (4 bytes, each)

Intended "to make sth sure on EPROMs" ;-) Checksum alorgythm not specified.

DINF - Dumper information block (204 Bytes)

100 bytes ASCIZ name of the person who dumped the cart
4 bytes day, month, year-lsb, year-msb when cartridge was dumped
100 bytes ASCIZ agent "name of the ROM-dumping means used"

Note: All words and dwords in header/chunks stored LSB first.

Cartridge IRQ Counters

The MMC3's scanline counter

The MMC3 bases it's scanline counter on PPU address line A13 (which is why IRQ's can be fired off manually by toggling A13 a bunch of times via \$2006).

A13 cycles (0 -> 1) exactly 42 times per scanline, whereas the CPU count of cycles per scanline is not an exact integer (113.67).

Konami IRQ counters

Running at 113.75 cycles, including during VBlank.

Famicom Disk System IRQ counters

Allows to count clock cycles, rather than scanlines.

Cartridge Bus Conflicts

/PRG Pin - Indicates CPU Memory Access to 8000h-FFFFh (LOW=Read or Write)

R/W Pin - Indicates CPU Direction (LOW=Write, HIGH=Read)

The /PRG Pin indicates read-or-write access to the PRG ROM memory area at 8000h-FFFFh, the read/write direction could be determined by R/W Pin. Most cartridges are ignoring the R/W signal, and are assuming all memory accesses to be read-requests (which makes sense since ROM is read-only).

However, many cartridges have write-only mapper ports at 8000h-FFFFh, activated when /PRG=LOW and R/W=LOW. Many of these carts (especially simple TTL circuits like UNROM, CNROM, etc.) still activate ROM on any /PRG signal without checking R/W, so that ROM outputs data simultaneously with the CPU writing data to the mapper port.

Common workaround is to write to a ROM address that contains a value equal to the written value. Also one could probably write to an address that contains FFh (low signals are stronger than high signals, so the values would be logically, or 'forcefully' ANDed. Don't know about any games using that method though). Another workaround is to interpret the lower address bits instead of the data bits (eg. Mapper 225), that's of course still producing a bus-conflict (shortcut), but without disturbing the program flow.

Cartridge Cicurity Chip (CIC) (Lockout Chip)

Lockout chips are contained in most NES consoles, and in all NES cartridges. Both chips are generating an identical serial data stream, and, everything works fine if the streams match. Otherwise the chip in the console issues a 1Hz reset signal - the cartridge will be permanently restarted, screen will be flashing, and the power LED blinks.

That mechanism is intended both to prevent software piracy, and to prevent third party developers from distributing (unlicensed) games. Also, an US cartridge won't work on a UK console and vice versa, because of different lockout chip versions used in different countries:

3193A	NES, USA
3195A	NES, European
3196A	NES, Hong Kong
3197A	NES, UK
N/A	NES, newer top-loading version (1993-1995)
N/A	Famicom, Japan (1983-1995)

The chips have been invented when releasing the NES in 1985, the original Famicom didn't have lockout chips, nor do newer Famicoms, for backwards compatibility reasons. There have been also some lockout chip revisions to make newer NES consoles incompatible with some "faked" lockout chips from other manufacturers.

The lockout chips are 4bit microprocessors in DIL16 package with a built-in program called 10NES, and are connected to S0,S1,S2,4MHz pins of the 72pin NES cartridge slot. Pin 4 of the chip is used to configure the chip:

HIGH (+5V)	Lock, used in console
LOW (GND)	Key, used in cartridge

To disable the chip in the console, wire that pin to GND instead of 5V, the NES will then work with any cartridges with or without lockout chip, and with cartridges from other countries - though some NTSC (60Hz) games may be incompatible with PAL (50Hz) refresh rates, and vice versa.

Cartridge Game Genie

The Game Genie is an adapter to be connected between the console and game cartridge, it includes a BIOS ROM which prompts the user to enter 6-letter or 8-letter cheat-codes, and then starts the actual game. The adapter compares the CPU address bus (PRG ROM area 8000h-FFFFh), and optionally also the CPU data bus (reduces the risk to mess-up values in other banks in cartridges with Memory Mappers), if the comparison matches, then the value on data bus will be replaced.

Code Format

The letters are translated into 4bit Hex-digits:

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Letter	A	P	Z	L	G	I	T	Y	E	O	X	U	K	S	V	N

Address/Data/Compare bits A14-A0, D7-D0, C7-C0 are scrambled as such:

Char	Bit3	Bit2	Bit1	Bit0	Char	Bit3	Bit2	Bit1	Bit0
1st	D7	D2	D1	D0	2nd	A7	D6	D5	D4
3rd	LEN	A6	A5	A4	4th	A3	A14	A13	A12
5th	A11	A2	A1	A0	6th	CD3	A10	A9	A8
7th	C7	C2	C1	C0	8th	D3	C6	C5	C4

6-Letter code: LEN=0, CD3 used as D3, acts as "[A]=D"

8-letter code: LEN=1, CD3 used as C3, acts as "If [A]=C then [A]=D"

Example: Code "SXIOPO" changes [91D9h]=ADh (Infinite lives in smb1).

Cartridge Pin-Outs

60-Pin Famicom Cartridge connector

1	GND	19-25	PPU A6-A0	45	EXP SND_IN
2-13	CPU A11-A0	26-29	PPU D0-D3	45, 46	EXP SND_OUT
14	CPU R/W	30-31	+5VDC	47	PPU /WR
15	CPU /IRQ	32	PHI2 CLK	48	PPU NT /CS
16	GND	33, 35	CPU A12-A14	49	PPU NT /A13
17	PPU /RD	36-43	CPU D7-D0	49-56	PPU A7-A12, A13
18	PPU NT A10	44	CPU /PRG	57-60	PPU D7-D4

72-Pin NES Cart connector

Pin	Dir	Use	Expl.
1	Out	VEE	GND
2-13	Out	CPU	A11-A0
14	Out	CPU	R/W
15	I/O	CPU	/IRQ
16-20	I/O	EXP	Expansion Port Pins 42-38 (not used by the console itself)
21	Out	PPU	/R
22	In	PPU	VA10 (A10 of internal 2K VRAM, ie. select BLK0 or BLK1)
23-29	Out	PPU	A6-A0
30-33	I/O	PPU	D0-D3
34-35	I/O	CIC	S0-S1 (cicurity/lockup chip protocol signals)
36	Out	VCC	+5VDC
37	Out	CPU	21.47727MHz (NTSC), 26.601712MHz (PAL)
38	Out	CPU	PHI2
39-41	Out	CPU	A12-A14
42-49	I/O	CPU	D7-D0
50	Out	CPU	/PRG (PRG-ROM access, logical NAND of PHI2 and CPU A15)
51-55	I/O	EXP	Pins 06-10
56	Out	PPU	/W
57	In	PPU	/VCS (internal 2K VRAM Chip-Select)

```

58   Out PPU /A13 (inverted A13, wired to /VCS when used as name table)
59-65 Out PPU A7-A9,A11,A10,A12-A13
66-69 I/O PPU D7-D4
70   I/O CIC S2 (cicurity/lockup chip protocol signals)
71   Out CIC 4Mhz (cicurity/lockout chip clock line)
72   Out VEE GND

```

Mapper 0: NROM - (No Mapper)

No mapper used in games with 32K ROM + 8K VROM (or less).
Name Table can be hardwired either to Horizontal or Vertical Mirroring.

Mapper 1: MMC1 - PRG/32K/16K, VROM/8K/4K, NT

This mapper is used on numerous U.S. and Japanese games, including Legend of Zelda, Metroid, Rad Racer, Mega Man 2, and many others.

```

8000h-FFFFh
  Bit 0  Serial data loaded to 5bit shift register (LSB=1st write)
  Bit 7  Clear 5bit shift register (1=Reset, next write will be "1st write")

```

On fifth write, data in shift register is copied to Register 0..3 (depending on upper address bits), and the shift register is automatically cleared.

```

8000h-9FFFh  Register 0 - Configuration Register
  Bit0-1 Name Table Mirroring
    0,1  Single-Screen (BLK0 only)
    2    Two-Screen Vertical Mirroring
    3    Two-Screen Horizontal Mirroring
  Bit2-3 PRG-Switching Mode (usually 3)
    0,1  Switchable 32K Area at 8000h-FFFFh (via Register 3)
    2    Switchable 16K Area at C000h-FFFFh (via Register 3)
         And Fixed 16K Area at 8000h-BFFFh (always 1st 16K)
    3    Switchable 16K Area at 8000h-BFFFh (via Register 3)
         And Fixed 16K Area at C000h-FFFFh (always last 16K)
  Bit4    VROM Switching Size (for carts with VROM)
    0     Swap 8K of VROM at PPU 0000h
    1     Swap 4K of VROM at PPU 0000h and 1000h
A000h-BFFFh  Register 1
  Bit4-0 Select 4K or 8K VROM bank at 0000h (4K and 8K Mode, see Reg0/Bit4)
C000h-DFFFh  Register 2
  Bit4-0 Select 4K VROM bank at 1000h (used in 4K Mode only, see Reg0/Bit4)
E000h-FFFFh  Register 3
  Bit3-0 Select 16K or 2x16K ROM bank (see Reg0/Bit3-2)
  Bit4    Unused ?

```

Initially 1st and last 16K are mapped to 8000h and C000h.
In 32K PRG and 8K VROM mode, bank numbers specified in steps of two.

Register 3 is restricted to sixteen 16K banks, cartridges with more than 256K PRG ROM use Bit4 of Register 0-2 to expand the available memory area:

```

Register 0, Bit 4
<1024K carts>
  0 = Ignore 256K selection register 1
  1 = Acknowledge 256K selection register 1
Register 1, Bit4 - 256K ROM Selection Register 0
<512K carts>
  0 = Swap banks from first 256K of PRG
  1 = Swap banks from second 256K of PRG
<1024K carts with bit 4 of register 0 off>

```

```

0 = Swap banks from first 256K of PRG
1 = Swap banks from third 256K of PRG
<1024K carts with bit 4 of register 0 on>
Low bit of 256K PRG bank selection
Register 2, Bit4 - 256K ROM Selection Register 1
<1024K carts with bit 4 of register 0 off>
Store but ignore this bit (base 256K selection on 256K selection Reg 0)
<1024K carts with bit 4 of register 0 on>
High bit of 256K PRG bank selection

```

Reportedly some MMC1 carts have 16K SRAM, of which only 8K are battery backed, no idea how/where the additionally 8K are accessed, and no idea which 8K are battery backed and which are not (?).

Mapper 2: UNROM - PRG/16K

This mapper is used on many older U.S. and Japanese games, such as Castlevania, Mega Man, Ghosts & Goblins, and Amagon.

```

8000h-FFFFh Select 16K ROM bank at 8000h-BFFFh (initially 1st bank)
N/A         Fixed 16K ROM at C000h-FFFFh (always last bank)

```

All carts using it have 8K of VRAM at PPU 0000h. Most carts with this mapper are 128K. A few, mostly Japanese carts, such as Final Fantasy 2 and Dragon Quest 3, are 256K.

Bus-conflicts. Uses a 74LS161 chip (connection: /PRG-CLK, R/W-/LOAD), and its outputs are each ORed with A14 by a 74LS32 chip.

Board NES-UN-ROM-05 and Konami 531320 (both using only 3bits / 8banks)

Mapper 3: CNROM - VROM/8K

This mapper is used on many older U.S. and Japanese games, such as Solomon's Key, Gadius, Cybernoid, and Hudson's Adventure Island.

```

8000h-FFFFh
Bit 0-1 Select 8K VROM bank at PPU 0000h (initially 1st bank)
Bit 4-5 Name Table Mirroring Select?

```

Bus-conflicts. Uses a 74LS161 chip (connection: /PRG-CLK, R/W-/LOAD), circuit for name table outputs is unknown?

Cybernoid supports both above Port 8000h-FFFFh, and alternately Port 6000h.

Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ

A great majority of newer NES games (early 90's) use this mapper, both U.S. and Japanese. Among the better-known MMC3 titles are Super Mario Bros. 2 and 3, Mega Man 3, 4, 5, and 6, and Crystals.

```

8000h Index/Control (5bit)
Bit7   CHR Address Select (0=Normal, 1=Address Areas XOR 1000h)
Bit6   PRG Register 6 Area (0=8000h-9FFFh, 1=C000h-DFFFh)
Bit2-0 Command Number
0 - Select 2x1K VROM at PPU 0000h-07FFh (or 1000h-17FFh, if Bit7=1)
1 - Select 2x1K VROM at PPU 0800h-0FFFh (or 1800h-1FFFh, if Bit7=1)
2 - Select 1K VROM at PPU 1000h-13FFh (or 0000h-03FFh, if Bit7=1)
3 - Select 1K VROM at PPU 1400h-17FFh (or 0400h-07FFh, if Bit7=1)
4 - Select 1K VROM at PPU 1800h-1BFFh (or 0800h-0BFFh, if Bit7=1)
5 - Select 1K VROM at PPU 1C00h-1FFFh (or 0C00h-0FFFh, if Bit7=1)
6 - Select 8K ROM at 8000h-9FFFh (or C000h-DFFFh, if Bit6=1)

```

7 - Select 8K ROM at A000h-BFFFh
 N/A - Fixed 8K ROM at C000h-DFFFh (or 8000h-9FFFh, if Bit6=1)
 N/A - Fixed 8K ROM at E000h-FFFFh (always last 8K bank)
 8001h Data Register (Indexed via Port 8000h)
 A000h Mirroring Select (Bit0: 0=Vertical, 1=Horizontal Mirroring)
 A001h SaveRAM Toggle (Bit7: 0=Disable 6000h-7FFFh, 1=Enable 6000h-7FFFh)
 C000h IRQ Counter Register - The IRQ countdown value is stored here.
 C001h IRQ Latch Register - A temporary value is stored here.
 E000h IRQ Control Register 0
 Any value written here will disable IRQ's and copy the
 latch register to the actual IRQ counter register.
 E001h IRQ Control Register 1 - Any value written here will enable IRQ's.

The fixed PRG banks are always the LAST two 8K banks in the cart.
 On carts with VROM, the first 8K of VROM is swapped into PPU \$0000 on reset.
 On carts without VROM, as always, there is 8K of VRAM at PPU \$0000.

The IRQ counter is decremented each scanline, based on PPU address line A13 which toggles between Pattern Tables (LOW) and Name Tables (HIGH) 42 times per scanline. The counter is paused during VBlank, which allows to use the same settings for PAL and NTSC timings. Note that the counter gets clocked, even during VBlank, when toggling A13 a bunch of times via Port 2006h.

Multicarts with MMC3 and additional Game-Select Ports

[Mapper 44: 7-in-1 MMC3 Port A001h](#)
[Mapper 45: X-in-1 MMC3 Port 6000hx4](#)
[Mapper 47: 2-in-1 MMC3 Port 6000h](#)
[Mapper 49: 4-in-1 MMC3 Port 6xxxh](#)
[Mapper 52: 7-in-1 MMC3 Port 6800h with SRAM](#)

Mapper 5: MMC5 - BANKING, IRQ, SOUND, VIDEO, MULTIPLY, etc.

Used by Gun Sight (Laser Invasion), Uchuu Keibitai SDF, Bandit Kings (Suikoden), Castlevania 3, Nobunaga Sengoku (Nobunaga's Ambition 2), Nobunaga Bushou, Shin 4 Nin Uchi Mahjong, Ishin no Arashi, L'Empereur, Ganbare Goemon Gaiden (bugged hack?), Romance of the Three Kingdoms 2 (Sangokushi 2), Gemfire (Royal Blood), Uncharted Waters (Daikoukai Jidai), Aoki Ookami, Just Breed, Metal Slader Glory.

[Mapper 5: MMC5 - I/O Map](#)
[Mapper 5: MMC5 - CPU Memory Control](#)
[Mapper 5: MMC5 - Video Name Table](#)
[Mapper 5: MMC5 - Video Pattern Table](#)
[Mapper 5: MMC5 - Video Split and IRQ](#)
[Mapper 5: MMC5 - Video EXRAM](#)
[Mapper 5: MMC5 - Sound Control](#)
[Mapper 5: MMC5 - Other Registers](#)

Mapper 5: MMC5 - I/O Map

Summary of all MMC5 Registers

5000h	Sound Channel 1 Pulse Control
5002h	Sound Channel 1 Frequency LSB
5003h	Sound Channel 1 Frequency MSB
5004h	Sound Channel 2 Pulse Control

5006h	Sound Channel 2 Frequency LSB
5007h	Sound Channel 2 Frequency MSB
5010h	Sound Channel 3 Enable
5011h	Sound Channel 4 Synthetic Voice business channel 2
5015h	Sound Channel 1 and 2 Enable
5100h	PRG Bank Size (Mode for Port 5114h-5117h)
5101h	CHR Bank Size
5102h	RAM Write Protect Key 1
5103h	RAM Write Protect Key 2
5104h	EXRAM Mode Setting
5105h	Name Table Select
5106h	Name Table Fill-Mode Tile Number
5107h	Name Table Fill-Mode Palette Number
5113h-5117h	PRG Bank Selection Registers
5120h-5127h	CHR Bank Selection for Sprites and for CPU Access
5128h-512Bh	CHR Bank Selection for Background
5130h	Unknown
5200h	Horizontal Split Control
5201h	Horizontal Split Scroll Position
5202h	Horizontal Split CHR Bank Selection
5203h	Vertical IRQ Counter
5204h	Vertical IRQ Control/Status (R/W)
5205h	Multiply unit input/output
5206h	Multiply unit input/output
5800h	Unknown
5C00h-5FFFh	EXRAM (1K)

Ports 5102h-5103h, 5105h-5107h, 5200h-5203h are Write Only.

Ports 5204h-5206h are Read/Write. Other Ports unknown.

Mapper 5: MMC5 - CPU Memory Control

5100h - PRG Bank Size Control (Mode for Port 5114h-5117h)

Bit7-2 Not used
 Bit1-0 PRG Bank Size (0=32K, 1=16K, 2=Mixed, 3=8K)

5102h-5103h - RAM Write Protect Keys

5102h RAM Write Protect Key 0 (Lower 2bit must be 02h for write-enable)
 5103h RAM Write Protect Key 1 (Lower 2bit must be 01h for write-enable)

RAM is always read-able, but is write-able only with above settings.

5113h-5117h - PRG Bank Selection Registers

Port	Type	Mode3/8K	Mode2/Mixed	Mode1/16K	Mode0/32K
5113h	RAM	8K at 6000h	8K at 6000h	8K at 6000h	8K at 6000h
5114h	ROM/RAM	8K at 8000h, N/A	, N/A	, N/A	, N/A
5115h	ROM/RAM	8K at A000h, 2x8K at 8000h,	2x8K at 8000h,	2x8K at 8000h,	N/A
5116h	ROM/RAM	8K at C000h, 8K at C000h,	N/A	, N/A	
5117h	ROM	8K at E000h, 8K at E000h,	2x8K at C000h,	4x8K at 8000h	

Lower one or two bits of 2x8K or 4x8K bank numbers are ignored.

RAM bank selection via Port 5113h-5116h (not 5117h):

Bit7 ROM/RAM Mode (0=RAM, 1=ROM) (Port 5114h-5116h only, not 5113h,5117h)
 Bit6-3 Not used
 Bit2 RAM Chip Select (0=1st chip, 1=2nd chip, or open bus if single chip)
 Bit1-0 Select 8K RAM Bank in currently selected RAM chip (32K chips only)

Existing RAM configurations are: 8K (single 8K chip), 16K (two 8K chips), and 32K (single 32K chip).

On reset, 8K mode is activated, and all ROM banks are set to the LAST 8K bank in the cartridge.

Mapper 5: MMC5 - Video Name Table

5105h - Name Table Select

Bit1-0 Select NT0 VRAM at 2000h-23FFh (0=BLK0, 1=BLK1, 2=EXRAM, 3=FILLMODE)
Bit3-2 Select NT1 VRAM at 2400h-27FFh (0=BLK0, 1=BLK1, 2=EXRAM, 3=FILLMODE)
Bit5-4 Select NT2 VRAM at 2800h-2BFFh (0=BLK0, 1=BLK1, 2=EXRAM, 3=FILLMODE)
Bit7-6 Select NT3 VRAM at 2C00h-2FFFh (0=BLK0, 1=BLK1, 2=EXRAM, 3=FILLMODE)

If it isn't used for other purpose, then EXRAM can be used as 3rd Name-table.

5106h - Name Table Fill-Mode Tile Number (Bit7-0)

5107h - Name Table Fill-Mode Palette Number (only Bit1-0 used)

In FILLMODE, the entire Name-table is filled by Port 5106h/5107h settings.

Mapper 5: MMC5 - Video Pattern Table

5101h - CHR Page Size

Bit7-6 Not used
Bit1-0 CHR Bank Size (0=8K, 1=4K, 2=2K, 3=1K)

Bank selection registers below are 8bit, so 1K mode can address only 256K VROM, 8K mode could address up to 2MB VROM.

5120h-5127h - CHR Bank Selection for Sprites and for CPU Access

Port	Mode3/1K	Mode2/2K	Mode1/4K	Mode1/8K
5120h	1K at 0000h	N/A	N/A	N/A
5121h	1K at 0400h	2K at 0000h	N/A	N/A
5122h	1K at 0800h	N/A	N/A	N/A
5123h	1K at 0C00h	2K at 0800h	4K at 0000h	N/A
5124h	1K at 1000h	N/A	N/A	N/A
5125h	1K at 1400h	2K at 1000h	N/A	N/A
5126h	1K at 1800h	N/A	N/A	N/A
5127h	1K at 1C00h	2K at 1800h	4K at 1000h	8K at 0000h

Used for Sprite Tiles (not for Background Tiles), and also used for CPU VRAM Access via Port 2006h/2007h.

5128h-512Bh - CHR Bank Selection for Background

5128h	1K at X000h	N/A	N/A	N/A
5129h	1K at X400h	2K at X000h	N/A	N/A
512Ah	1K at X800h	N/A	N/A	N/A
512Bh	1K at XC00h	2K at X800h	4K at X000h	8K at 0000h

Used for Background Tiles, the "XN00h" addresses in 1K,2K,4K are shared for both Pattern Tables at 0N00h and 1N00h, ie. BG Pattern Table selection in Port 2000h/Bit4 doesn't matter (except in 8K mode).

Other Background CHR Bank Selection Modes (which do not use 5128h-512Bh)

In Horizontal Split Mode, left or right BG Tiles use 4K CHR bank in Port 5202h.

In ExGrafix Mode, 4K CHR banks are specified for each single BG Tile in EXRAM.

Mapper 5: MMC5 - Video Split and IRQ

MMC5 allows to split the screen horizontally and vertically.
Horizontal Split is handled by hardware (automatically mid-scanline).
Vertical Split is to be handled by software (upon IRQ during HBlank).

5200h - Horizontal Split Control

Bit7 For the E function (0=Don't use, 1=Use)
Bit6 Boundary's side is for using Split Mode extension of graphics
(0=Left side, 1=Right side)
Bit5 Not used
Bit4-0 Left boundary is designated with the char. # to count places

Used by Uchuu Keibitai SDF, most or all other games don't use H-Split.

Examples for 5200h Settings:

00h (not?) used yet
82h Used for SplitMode GFX extension from left 1-2 character
C2h Used for SplitMode GFX extension from the right side 3 chars.
C0h Used for SplitMode GFX extension on the whole screen
D0h Used for SplitMode GFX extension on the right side of the screen
90h Used for SplitMode GFX extension on the left side of the screen

5201h - Horizontal Split Scroll Position

"\$2005 determines the vertical movement; it can also delay ext. gfx's vert. movement if necessary. It's written 2 times in bulk in the same way as it would slip off a grade in \$2005."

5202h - Horizontal Split CHR Bank Selection

Bit7-6 Not used
Bit5-0 Select 4K VROM at both 0000h-0FFFh and 1000h-1FFFh

Presumably used for BG Tiles in the Horizontal Split area, instead of the normal BG-CHR Bank Selection via 5128h-512Bh.

5203h - Vertical IRQ Counter

MMC3-style, decremented each scanline, paused during VBlank.
A setting of 00h seems to disable the counter (or, maybe sets it to 256 lines).

5204h - Vertical IRQ Control/Status (R/W)

Bit7/Write IRQ Enable (0=Disable, 1=Enable)
Bit6/Read Unknown

Reading from 5204h automatically acknowledges IRQs, and probably also returns the current status of the IRQ flag in one bit.

Bit6 contains whatever flag, it seems to be NOT directly IRQ related - many games do never write to 5203h/5204h, but still expect Bit6 to toggle on/off.

Mapper 5: MMC5 - Video EXRAM

5C00h-5FFFh - EXRAM

Built-in 1K RAM, can be used in different modes, as VRAM or as WRAM.

5104h - EXRAM Mode Setting

Bit7-6 Not used
Bit1-0 Select EXRAM Mode
0 VRAM Extra Name Table (via Port 5105h)
1 VRAM ExGrafix Color Expansion (see below)
2 General purpose WRAM (read/write)

3 General purpose WRAM (write protected)

In VRAM modes, EXRAM can be probably accessed during VBlank only (just as normal VRAM). In WRAM modes, EXRAM can be probably accessed at any time, for use as general purpose Work RAM, instead of (or additionally to) normal SRAM.

ExGrafix Mode (used by most MMC5 titles, except Castlevania 3)

5C00h-5FBFh - Tile Number banks and Palettes for 32x30 Tiles

Bit7-6 Palette Number for each Tile

Bit5-0 4K Bank Number for each Tile

The 6bit Bank Numbers expand each of the 8bit Tile Numbers in Name Table entries 000h-3BFh to 14bit Tile Numbers (max 256K VROM addressable). The Palette Numbers allow to specify different palettes for each single Tile, instead of the normal Name Table palettes which share one palette entry for each 4 Tiles. Name Table entries 3C0h-3FFh and EXRAM 5FC0h-5FFFh are not used in this mode. Also BG-CHR Bank Selection Ports 5128h-512Bh are not used.

Mapper 5: MMC5 - Sound Control

MMC5 Sound, Japanese 60pin Famicom carts only, not NES 72pin carts.

5000h Sound Channel 1 Pulse Control

5004h Sound Channel 2 Pulse Control

Bit7-6 Duty Cycle (0..3 = 87.5%, 75.0%, 50.0%, 25.0%)

Bit5 Waveform Hold (e.g. Looping) (0=Off, 1=On)

Bit4 Envelope Select (0=Varied, 1=Fixed)

Bit3-0 When Bit4=0: Playback Rate (0..0Fh = Fast..Slow)

Bit3-0 When Bit4=1: Output Volume (0..0Fh)

5002h Sound Channel 1 Frequency LSB

5006h Sound Channel 2 Frequency LSB

Bit7-0 Lower 8bit of 11bit Frequency

5003h Sound Channel 1 Frequency MSB

5007h Sound Channel 2 Frequency MSB

Bit7-3 Sound Occurrence Time

Bit2-0 Upper 3bit of 11bit Frequency

5010h Sound Channel 3 Enable

Bit7-1 Not used

Bit0 Wave output (0=Off, 1=On)

5011h ch4 Synthetic Voice business channel 2

Bit7-0 Wave Size

5015h Sound Channel 1 and 2 Enable

Bit7-2 Not used

Bit1 Channel 2 (0=Disable, 1=Enable)

Bit0 Channel 1 (0=Disable, 1=Enable)

Mapper 5: MMC5 - Other Registers

5205h - WR multiply unit input/output

5206h - WR multiply unit input/output

$(\$5205_{in}) * (\$5206_{in}) = \$5205, \5206_{out}

Result seems to be 16bit, 5205h=LSB, 5206h=MSB (?)

5130h - Unknown

Just Breed, Gun Sight, and Uchuu Keibitai SDF write 00h to this address.

5800h - Unknown

Just Breed writes 0xh to this address.

Mapper 6,8,12,17: Front Far East (FFE) Configuration, IRQs, Patches

Front Far East (FFE) disk drive "backup unit" connects to the cartridge slot, allows to load copies of games from floppy/hdd/cdrom into RAM (max 512K) and VRAM (max 256K).

FFE Mapper Modes

[Mapper 2: UNROM - PRG/16K](#)

[Mapper 6: FFE F4xxx - PRG/16K, VROM/8K, NT, IRQ](#)

[Mapper 7: FFE F4xxx - PRG/16K, VROM/8K, NT, IRQ](#)

[Mapper 8: FFE F3xxx - PRG/32K, VROM/8K, NT, IRQ](#)

[Mapper 12: FFE F6xxx - Not specified, NT, IRQ](#)

[Mapper 17: FFE F8xxx - PRG/8K, VROM/1K, NT, IRQ](#)

FFE IRQ Registers

```
4501h  IRQ Disable/Acknowledge (write any value, usually 00h)
4502h  IRQ set lower 8bit of 16bit counter
4503h  IRQ set upper 8bit of 16bit counter and Start/Enable IRQs
```

IRQ counter is incremented each clock cycle, and produces IRQ on overflow.

FFE Trainers/Patches

All FFE games are patched to work with the "FFE" mappers. In case that the patches don't fit into normal ROM area, additional 512-byte patches are often located in SRAM area at 7000h-71FFh (or, in a few cases, reportedly at 5D00h), that patch-area may be used to handle FFE memory mapping, or for cheats/trainers. Some MMC games also contain similar trainers (maybe working on FFE device, if it supports MMC mappers?, or otherwise working on emulators only).

FFE Configuration Registers

Configuration Registers are initialized before the game is started, so most games don't need to access these registers, except for changing Name Table / Mirroring bits. A few games might also change the mode bits.

```
42FCh-42FFh  Configuration Register 1
A0          Name Table Mode (0=One-Screen, 1=Two-Screen) (with D4 below)
A1          Unknown (0=WE, 1=SW) (usually 1)
D7-D5      Memory Mode (0-7) "*MODE"
            1          Mapper 6 F4xxx
            2          Mapper 2 UNROM
            3          Mapper ? F4xxx
            4          Mapper 8 F3xxx/GNROM
            0,5-7      unknown (Great Tank uses settings 1 and 6)
            ?          unknown how to select Mapper 12 and Mapper 17
```

```

    0      Mapper 17 (Kaiketsu, Saiyuuki)
    7      Mapper 17 (Wing of Madoola)
D4      When A0=0: Select VRAM Page (?=BLK0, ?=BLK1)
        When A0=1: ?Mirroring (0=Vertical, 1=Horizontal Mirroring)
D3-D0   Unknown (usually zero)
43FEh   Memory Control (apparently independendly of current Mode) (?)
D7-D2   Select ?K ROM at 8000h-?
D1-D0   Select 8K VROM at PPU 0000h-1FFFh
43FFh   Memory Control (as for current mode, ie. mirror of 8000h-FFFFh) (?)
4500h   Configuration Register 2
D7-D6   FDS Mode (0=Disk/Load, 1=Reserved, 2=Cartridge, 3=Disk/Execute)
D5-D4   SRAM 6000h-7FFFh BANK "Present or Not" (0-3=?)
D3      SW Pin (maybe something related with above WE/SW selection)
D2-D0   PPU Mode Select (1or2?="*MODE" (32K), 5=256K, VRAM EXT, 7=256K)

```

Mapper 6: FFE F4xxx - PRG/16K, VROM/8K, NT, IRQ

Several hacked Japanese titles use this mapper, such as the hacked version of Wai Wai World. The unhacked versions of these games seem to use a Konami VRC mapper, and it's better to use them if possible.

```

8000h-FFFFh Memory Control (6bit)
Bit1-0   Select 8K VROM (read/write-able) at PPU 0000h-1FFFh
Bit5-2   Select 16K ROM at 8000h-BFFFh (bank 0-0Fh)
N/A     Fixed 16K ROM at C000h-FFFFh (always bank 7) (!)

```

Additional FFE registers:

[Mapper 6,8,12,17: Front Far East \(FFE\) Configuration, IRQs, Patches](#)

Mapper ?: FFE F4xxx - PRG/16K, VROM/8K, NT, IRQ

10% of games declared as "Mapper 6" are this (and not Mapper 6).

```

8000h-FFFFh Memory Control (6bit)
Bit3-0   Select 16K ROM at 8000h-BFFFh
Bit5-4   Select 8K VROM at PPU 0000h-1FFFh

```

Lower bits are ROM bank, upper bits VROM bank, ie. vice-versa as Mapper 6.

Additional FFE registers:

[Mapper 6,8,12,17: Front Far East \(FFE\) Configuration, IRQs, Patches](#)

Mapper 7: AOROM - PRG/32K, Name Table Select

Numerous games released by Rare Ltd. use this mapper, such as Battletoads, Wizards & Warriors, and Solar Jetman.

```

8000h-FFFFh Memory Control
Bit2-0   Select 32K ROM bank at 8000h-FFFFh (initially 1st bank)
Bit4     One-Screen Name Table Select (0=BLK0, 1=BLK1)
Bit3,5-7 Not used

```

Uses a single 74LS161 chip (connection: /PRG-CLK, R/W-/LOAD). ANROM additionally uses a 74LS02 to enable ROM only when R/W=HIGH and /PRG=LOW.

Board NES-AOROM-03: 256K PRG ROM (8 banks) (32pin ROM) (with bus-conflicts)

Board NES-BNROM-01: 128K PRG ROM (4 banks) (28pin ROM) (with bus-conflicts)

Board NES-ANROM-03: 128K PRG ROM (4 banks) (28pin ROM) (without bus-conflicts)

All carts using it have 8K of VRAM at PPU 0000h.

Deadly Towers seems to be BNROM, using horizontal mirroring, instead Bit4?

Mapper 8: FFE F3xxx - PRG/32K, VROM/8K, NT, IRQ

Several hacked Japanese titles use this mapper, such as the hacked version of Doraemon.

8000h-FFFFh Memory Control (same as GNROM, Mapper 66)
Bit1-0 Select 8K VROM (usually read-only) at PPU 0000h-1FFFh
Bit5-4 Select 32K ROM at 8000h-FFFFh (initially 1st bank)

Additional FFE registers:

[Mapper 6,8,12,17: Front Far East \(FFE\) Configuration, IRQs, Patches](#)

Mapper 9: MMC2 - PRG/24K/8K, VROM/4K, NT, LATCH

Used only on by Punch-Out, and Mike Tyson's Punch-Out.

A000h-AFFFh Select 8K ROM at 8000h-9FFFh (initially 1st bank)
N/A Fixed 24K ROM at A000h-FFFFh (always last three 8K banks)
B000h-CFFFh Select 4K VROM at PPU 0000h-0FFFh
D000h-DFFFh Select 4K VROM at PPU 1000h-1FFFh (used when latch=FDh)
E000h-EFFFh Select 4K VROM at PPU 1000h-1FFFh (used when latch=FEh)
F000h-FFFFh Mirroring Select (Bit0: 0=Vertical, 1=Horizontal mirroring)
PPU 1FD0h-1FDFh Access to Pattern Table 0, Tile FDh --> sets latch=FDh
PPU 1FE0h-1FEFh Access to Pattern Table 0, Tile FEh --> sets latch=FEh

The latch contains FEh on reset. The latch is automatically written to on any access to PPU 1FD0h-1FEFh, which does usually happen when the PPU fetches bitmap data for Tile FDh or FEh from Pattern Table 1. The latches might also get changed on access to PPU 0FD0h-0FEFh (?)

Mapper 10: MMC4 - PRG/16K, VROM/4K, NT, LATCH

Used only by Fire Emblem, Fire Emblem Gaiden, and Family War.

A000h-AFFFh Select 16K ROM bank at 8000h-BFFFh (initially 1st bank)
N/A Fixed 16K ROM bank at C000h-FFFFh (always last bank)
B000h-BFFFh Select 4K VROM bank at PPU 0000h-0FFFh (used when latch0=FDh)
C000h-CFFFh Select 4K VROM bank at PPU 0000h-0FFFh (used when latch0=FEh)
D000h-DFFFh Select 4K VROM bank at PPU 1000h-1FFFh (used when latch1=FDh)
E000h-EFFFh Select 4K VROM bank at PPU 1000h-1FFFh (used when latch1=FEh)
F000h-FFFFh Mirroring Select (Bit0: 0=Vertical, 1=Horizontal mirroring)
PPU 0FD0h-0FDFh Access to Pattern Table 0, Tile FDh --> sets latch0=FDh
PPU 0FE0h-0FEFh Access to Pattern Table 0, Tile FEh --> sets latch0=FEh
PPU 1FD0h-1FDFh Access to Pattern Table 1, Tile FDh --> sets latch1=FDh
PPU 1FE0h-1FEFh Access to Pattern Table 1, Tile FEh --> sets latch1=FEh

The latches contain FEh on reset. Latches are automatically written to on any access to PPU 0FD0h-0FEFh or 1FD0h-1FEFh, which does usually happen when the PPU fetches bitmap data for Tile FDh or FEh. The new latch setting is then used for all <further> tiles (tiles FDh/FEh are still fetched from the <old> latch setting).

Mapper 11: Color Dreams - PRG/32K, VROM/8K

This mapper is used on several unlicensed Color Dreams titles, including Crystal Mines and Pestertinator.

Not sure if their religious ("Wisdom Tree") games use the same mapper or not.

```
8000h-FFFFh Memory Control
  Bit3-0 Select 32K ROM bank at 8000h-FFFFh (initially 1st bank)
  Bit7-4 Select 8K VROM bank at PPU 0000h-1FFFh (initially 1st bank)
```

Many games using this mapper are somewhat glitchy. Bus-conflicts.
Uses a single 74LS377 (8bit D flip-flop with clock enable).

Mapper 12: FFE F6xxx - Not specified, NT, IRQ

No info. Don't have a ROM-image.

Maybe this meant to be "Mapper ?",

[Mapper ?: FFE F4xxx - PRG/16K, VROM/8K, NT, IRQ](#)

Additional FFE registers:

[Mapper 6,8,12,17: Front Far East \(FFE\) Configuration, IRQs, Patches](#)

Mapper 13: CPROM - 16K VRAM

Used by Videomation (a bitmap drawing program).

```
N/A          Fixed 4K VRAM at PPU 0000h-0FFFh (always Bank 0)
8000h-FFFFh Select 4K VRAM at PPU 1000h-1FFFh (Bank 0-3)
```

16K VRAM for 32x30 different BG tiles (plus 64 sprites). Bus-conflicts.

Mapper 15: X-in-1 - PRG/32K/16K, NT

Used by Contra 100-in-1 (fake multicart with less than 100 different games), and hacked versions of Crazy Climber, Dragon Ball, and Mobile Suit (single game carts).

```
8000h-FFFFh Memory Control (Decoded by address AND data lines)
  D5-D0 Select 16K ROM Bank (X)
  D6      Mirroring Control (0=Vertical, 1=Horizontal Mirroring)
  D7      Select 8K ROM Bank (Y) (should be zero in non-8K-modes)
  A1-A0 ROM Bank Mode (0=32K, 1=128K, 2=8K, 3=16K)
```

Mapping in different modes is:

```
8K Mode - Bank (X*2+Y) at each 8000h, A000h, C000h, E000h
16K Mode - Bank (X) at 8000h-BFFFh and (X) at C000h-FFFFh
32K Mode - Bank (X) at 8000h-BFFFh and (X OR 1) at C000h-FFFFh
128K Mode - Bank (X) at 8000h-BFFFh and LAST bank at C000h-FFFFh
```

Initially first 32K ROM selected. The cartridge contains 8K VRAM.

Mapper 16: Bandai - PRG/16K, VROM/1K, IRQ, EPROM

This mapper is used on several Japanese titles by Bandai, such as the DragonBall Z series and the SD Gundam Knight series.

```
6000h,7FF0h,8000h Select 1K VROM at PPU 0000h-03FFh
6001h,7FF1h,8001h Select 1K VROM at PPU 0400h-07FFh
6002h,7FF2h,8002h Select 1K VROM at PPU 0800h-0BFFh
6003h,7FF3h,8003h Select 1K VROM at PPU 0C00h-0FFFh
6004h,7FF4h,8004h Select 1K VROM at PPU 1000h-13FFh
6005h,7FF5h,8005h Select 1K VROM at PPU 1400h-17FFh
6006h,7FF6h,8006h Select 1K VROM at PPU 1800h-1BFFh
```

```

6007h,7FF7h,8007h Select 1K VROM at PPU 1C00h-1FFFh
6008h,7FF8h,8008h Select 16K ROM at 8000h-BFFFh (initially 1st bank)
N/A Fixed 16K ROM at C000h-FFFFh (always last bank)
6009h,7FF9h,8009h Mirroring/Page Select (Bit1-0)
  0 Two-Screen Vertical mirroring
  1 Two-Screen Horizontal mirroring
  2 Single-Screen BLK0
  3 Single-Screen BLK1
600Ah,7FFAh,800Ah IRQ Control Register (Bit 0)
  0 Disable/Acknowledge IRQ
  1 Enable IRQ
600Bh,7FFBh,800Bh Low byte of IRQ counter
600Ch,7FFCh,800Ch High byte of IRQ counter
600Dh,7FFDh,800Dh EPROM I/O Port - I am not sure how this works.

```

The IRQ counter is decremented each clock cycle if active, and set off when it reaches zero. An IRQ interrupt is executed at that point.

Mapper 17: FFE F8xxx - PRG/8K, VROM/1K, NT, IRQ

Several hacked Japanese titles use this mapper, such as the hacked versions of Parodius and DragonBall Z 3.

```

4504h Select 8K ROM at 8000h-9FFFh (initially 1st half of 1st 16K)
4505h Select 8K ROM at A000h-BFFFh (initially 2nd half of 1st 16K)
4506h Select 8K ROM at C000h-DFFFh (initially 1st half of last 16K)
4507h Select 8K ROM at E000h-FFFFh (initially 2nd half of last 16K)
4510h Select 1K VROM at PPU 0000h-03FFh
4511h Select 1K VROM at PPU 0400h-07FFh
4512h Select 1K VROM at PPU 0800h-0BFFh
4513h Select 1K VROM at PPU 0C00h-0FFFh
4514h Select 1K VROM at PPU 1000h-13FFh
4515h Select 1K VROM at PPU 1400h-17FFh
4516h Select 1K VROM at PPU 1800h-1BFFh
4517h Select 1K VROM at PPU 1C00h-1FFFh

```

Additional FFE registers:

[Mapper 6,8,12,17: Front Far East \(FFE\) Configuration, IRQs, Patches](#)

Mapper 18: Jaleco SS8806 - PRG/8K, VROM/1K, NT, IRQ, EXT

This mapper is used on several Japanese titles by Jaleco, such as Baseball 3, Lord of Kings, etc.

```

8000h/8001h Select 8K ROM at 8000h-9FFFh (Lower/Upper 4bits)
8002h/8003h Select 8K ROM at A000h-BFFFh (Lower/Upper 4bits)
9000h/9001h Select 8K ROM at C000h-DFFFh (Lower/Upper 4bits)
N/A Fixed 8K ROM at E000h-FFFFh (always last bank)
9002h Battery Back SRAM (Bit0: 0=Enable, 1=Disable)
      (unused by Lord of Kings)
9003h Unknown
      (used by Lord of Kings)
A000h/A001h Select 1K VROM at PPU 0000h-03FFh (Lower/Upper 4bits)
A002h/A003h Select 1K VROM at PPU 0400h-07FFh (Lower/Upper 4bits)
B000h/A001h Select 1K VROM at PPU 0800h-0BFFh (Lower/Upper 4bits)
B002h/A003h Select 1K VROM at PPU 0C00h-0FFFh (Lower/Upper 4bits)
C000h/C001h Select 1K VROM at PPU 1000h-13FFh (Lower/Upper 4bits)
C002h/C003h Select 1K VROM at PPU 1400h-17FFh (Lower/Upper 4bits)
D000h/D001h Select 1K VROM at PPU 1800h-1BFFh (Lower/Upper 4bits)
D002h/D003h Select 1K VROM at PPU 1C00h-1FFFh (Lower/Upper 4bits)
E000h/E001h Lower 8bit of decrementing 16bit IRQ counter (Lower/Upper 4bits)
E002h/E003h Upper 8bit of decrementing 16bit IRQ counter (Lower/Upper 4bits)
F000h IRQ Control Register 0
      Bit0 Maybe 1=Load Counter?

```

F001h IRQ Control Register 1
 Bit0 IRQ Enable (0=Disabled, 1=Enable)
 Bit1-3 IRQ Counter Width (0=16bit, 1=12bit, 2-3=8bit, 4-7=4bit)
 With widths less than 16bit, underflows recurse only lower counter bits.
 F002h Name Table Select (2bit)
 0 Two-Screen, Horizontal Mirroring
 1 Two-Screen, Vertical Mirroring
 2-3 Single-Screen BLK0
 F003h Unused (or an External I/O Port which is unused?)

Mapper 19: Namcot 106 - PRG/8K, VROM/1K/VRAM, IRQ, SOUND

This mapper is used on several Japanese titles by Namcot, such as Splatterhouse and Family Stadium '90.

Pattern Table Control

8000h-87FFh Select 1K VROM at PPU 0000h-03FFh (with E800h/Bit6)
 8800h-8FFFh Select 1K VROM at PPU 0400h-07FFh (""")
 9000h-97FFh Select 1K VROM at PPU 0800h-0BFFh (""")
 9800h-9FFFh Select 1K VROM at PPU 0C00h-0FFFh (""")
 A000h-A7FFh Select 1K VROM at PPU 1000h-13FFh (with E800h/Bit7)
 A800h-AFFFh Select 1K VROM at PPU 1400h-17FFh (""")
 B000h-B7FFh Select 1K VROM at PPU 1800h-1BFFh (""")
 B800h-BFFFh Select 1K VROM at PPU 1C00h-1FFFh (""")

The upper two bits Port E800h-EFFFh are used to select VROM/VRAM mode (mind that the lower six bits of that port Select 8K ROM at A000h-BFFFh).

E800h, Bit6 VROM/VRAM Mode for PPU 0000h-0FFFh (0=VROM+VRAM, 1=VROM-Only)
 E800h, Bit7 VROM/VRAM Mode for PPU 1000h-1FFFh (0=VROM+VRAM, 1=VROM-Only)

In VROM-Only mode, VROM banks 0-FFh can be selected. In VROM+VRAM mode only VROM banks 0-DFh can be selected, and values E0h-FFh select VRAM.

Name Table Control

C000h-C7FFh Select 1K VROM/VRAM at PPU 2000h-23FFh (E0h and up = VRAM)
 C800h-CFFFh Select 1K VROM/VRAM at PPU 2400h-27FFh (E0h and up = VRAM)
 D000h-D7FFh Select 1K VROM/VRAM at PPU 2800h-2BFFh (E0h and up = VRAM)
 D800h-DFFFh Select 1K VROM/VRAM at PPU 2C00h-2FFFh (E0h and up = VRAM)

Only VROM banks 0-DFh can be selected, and values E0h-FFh activate internal VRAM, Bit0 of the bank number is then used to select BLK0 or BLK1.

CPU Memory Control

E000h-E7FFh Select 8K ROM at 8000h-9FFFh (initially 1st half of 1st 16K)
 Bit5-0 Page_number
 E800h-EFFFh Select 8K ROM at A000h-BFFFh (initially 2nd half of 1st 16K)
 Bit5-0 Page_number
 Bit6 Select at CHR_address \$0000-\$0FFF 0:ROM&RAM 1:ROM
 Bit7 Select at CHR_address \$1000-\$1FFF 0:ROM&RAM 1:ROM
 F000h-F7FFh Select 8K ROM at C000h-DFFFh (initially 1st half of last 16K)
 Bit5-0 Page_number
 N/A Fixed 8K ROM at E000h-FFFFh (always 2nd half of last 16K)

The lower 6bit of these registers specify ROM bank numbers. Caution: The upper 2bit of E800h-EFFFh are used to select Pattern Table VROM/VRAM Mode.

IRQ Control

5000h-57FFh Bit7-0: Lower 8bit of 15bit IRQ counter (R/W) (!)

5800h-5FFFh Bit6-0: Upper 7bit of 15bit IRQ counter (R/W) (!)
Bit7: 0=Disable IRQs, 1=Enable IRQs

The IRQ counter is incremented each clock cycle, an IRQ is generated when it overflows (at 7FFFh, since it's a 15bit value). Sangokushi 2 uses IRQs, but many other Namcot games don't use IRQs.

Sound Control

4800h Expand I/O Data Register
F800h Expand I/O Address Register
Bit7 Auto Increment (0=Disable, 1=Enable)
Bit6-0 Address (00h-7Fh)

Index Addresses: (Dots "." = Japanese text, not translated)

00h-3Fh See NAMCO.TXT, Japanese (.....)
40h, 48h, 50h, 58h, 60h, 68h, 70h, 78h Channel 1-8, Frequency Lower 8bit
41h, 49h, 51h, 59h, 61h, 69h, 71h, 79h See NAMCO.TXT, Japanese (.....)
42h, 4Ah, 52h, 5Ah, 62h, 6Ah, 72h, 7Ah Channel 1-8, Frequency Middle 8bit
43h, 4Bh, 53h, 5Bh, 63h, 6Bh, 73h, 7Bh See NAMCO.TXT, Japanese (.....)
44h, 4Ch, 54h, 5Ch, 64h, 6Ch, 74h, 7Ch Channel 1-8, Frequency Upper 2bit & Option
Bit7-5 Not used
Bit4-2 VVV: 8-(.....) (... 2byte) ...: VVV=000... 16byte, VVV=100..8byte....
Bit1-0 Frequency Upper 2bit
45h, 4Dh, 55h, 5Dh, 65h, 6Dh, 75h, 7Dh See NAMCO.TXT, Japanese (.....)
46h, 4Eh, 56h, 5Eh, 66h, 6Eh, 76h, 7Eh Channel 1-8, Offset Address (00h-3Fh)
Bit7-1 AAAAAAA [6bit address stored in a 7bit value?]
Bit0 Not used
47h, 4Fh, 57h, 5Fh, 67h, 6Fh, 77h, 7Fh Channel 1-8
Bit7-4 ????: 7... (kingofkings), 3...
Bit3-0 VVVV:

Frequency: 0=Lowest, 3FFFFh=Highest.

According to Gorohs frequency table, Tone "C" of Octave "1-8" is: 1:47Eh, 2:8FBh, 3:11F6h, 4:23ECh, 5:47DAh, 6:8FB3h, 7:11F66h, 8:23ECCh.

According to my CPC frequency table, middle "C" of octave "0" should be 261.626Hz. No idea how to match that into a formula.

Mapper 20: Disk System - PRG RAM, BIOS, DISK, IRQ, SOUND

Used by Famicom Disk System only.

[Famicom Disk System \(FDS\)](#)

Mapper 21: Konami VRC4A/VRC4C - PRG/8K, VROM/1K, NT, IRQ

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

Mapper 22: Konami VRC2A - PRG/8K, VROM/1K, NT

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

Mapper 23: Konami VRC2B/VRC4E - PRG/8K, VROM/1K, NT,

(IRQ)

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

Mapper 24: Konami VRC6A - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

Mapper 25: Konami VRC4B/VRC4D - PRG/8K, VROM/1K, NT, IRQ

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

Mapper 26: Konami VRC6B - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

VRC6 Sound Registers

Two Rectangle channels with 4bit volume levels each, one Saw channel with 5bit volume level. These are added into a 6bit output level, and then merged with normal Famicom sound signal.

9000h/A000h - Channel 1/2 - Square Volume/Duty

Bit7-4 Duty Cycle bits:

0000 - 1/16	"-_____"	(6.25%)
0001 - 2/16	"--_____"	(12.50%)
0010 - 3/16	"---____"	(18.75%)
0011 - 4/16	"----___"	(25.00%)
0100 - 5/16	"-----"	(31.25%)
0101 - 6/16	"-----"	(37.50%)
0110 - 7/16	"-----"	(43.75%)
0111 - 8/16	"-----"	(50.00%)
1xxx - 16/16	"-----"	(100.00%)

Bit3-0 Linear Volume (0=Silence, 0Fh=Loudest)

100% Duty can be used as digitized mode, the channel permanently outputs high level, ie. the current volume setting, and isn't affected by the frequency registers.

B000h - Channel 3 - Saw Volume Step

Bit7-6 Not used

Bit5-0 Volume Step (V) (0..2Ah=Silent..Loudest) (2Bh..3Fh=Wraps/Garbage)

The overall output looks like "//////" whereas each "/" is split into 7 steps, the output level on step 1-7 is calculated as such:

```
FOR I=1 to 7                ;step 1-7
  IF I=1 THEN X=0           ;reset to 0 in 1st step
  ELSE X=(X+V) AND FFh     ;add accumulator
  Output=(X/8)              ;output upper 5bit of X
NEXT
```


Note: X is an 8bit value, and wraps on overflow, ie. if $V > 2Ah$.

9001h/A001h/B001h - Channel 1/2/3 - Frequency LSB

Bit7-0 Lower 8 bits of frequency data

9002h/A002h/B002h - Channel 1/2/3 - Frequency MSB

Bit7 Channel disable (0=Disable, 1=Enable)

Bit6-4 Not used

Bit3-0 Upper 4 bits of frequency data

To calculate output frequency:

Channel 1/2: $F = 1.79\text{MHz} / 16 / (N+1)$;16-step duty cycles

Channel 3: $F = 1.79\text{MHz} / 14 / (N+1)$;7-step phases

Mapper 21-26,73,75,85: Konami VRC Mappers

This chapter describes all known VRC variants. The different Port addresses are specified as X.Y.Z which may look a bit abstract at the first glance, at second glance it should be easier to understand as than using separate chapters for each of the 13 variants.

VRC Chip Versions

Type	PRG Bank	VROM Banks	NT	IRQ	Sound
VRC1	PRG/8K	VROM/4K	NT	-	-
VRC2	PRG/8K	VROM/1K	NT	-	-
VRC3	PRG/16K	VRAM		IRQ	-
VRC4	PRG/8K	VROM/1K	NT	IRQ	-
VRC6	PRG/16K/8K	VROM/1K	NT	IRQ	SOUND
VRC7	PRG/16K/8K	VROM/1K	NT	IRQ	SOUND

For most chips, there are different connection variants, VRC2a, VRC2b, etc.

VRC Data Bus

VRC1, VRC2, and VRC3 use a 4bit data bus, connected to D3-D0, any 8bit registers are thus split into two 4bit ports. VRC4 seems to have an additional D4 pin, which is used only for 5bit PRG ROM banks. VRC6 and VRC7 have a full 8bit data bus.

VRC Address Bus

Most VRCs have a 6bit address bus, described here as X.Y.Z - the upper four address bits are always A15-A12 (X), however, the connection of the lower two address bits (Y and Z) varies. VRC7 normally uses only X.Y bits (only the Sound registers use X.Y.Z). VRC1/VRC3 uses only the X address bits. A15 serves as chip select, and must be HIGH for all VRC registers.

VRC Connection Variants of Lower two bits of X.Y.Z addresses

Mapper	Y	Z	Used by
75 VRC1	-	-	Ganbare Goemon 1, Junior Basket - Two on Two, King Kong 2, Exciting Boxing, Jajamaru Ninpou Chou, Tetsuwan Atom
22 VRC2a	A0	A1	Twin Bee 3, Ganbare Pennant Race
23 VRC2b	A1	A0	Wai Wai World 1, Getsufuu Maden, Kaiketsu Yanchamaru 2, Dragon Scroll, Gryzor/Contra, Jarinko Chie, Ganbare Goemon
73 VRC3	-	-	Salamander
21 VRC4a	A2	A1	Wai Wai World 2
21 VRC4c	A7	A6	Ganbare Goemon Gaiden 2
25 VRC4b	A0	A1	Bio Miracle Bokutte Upa, Ganbare Goemon Gaiden, Gradius 2, Racer Mini Yonku
25 VRC4d	A2	A3	Teenage Mutant Hero Turtles 1+2, Goal!!
23 VRC4e	A3	A2	Parodius da!, Akumajou Special, Crisis Force, Tiny Toon Adventures 1, Moe Pro!

24	VRC6a	A1	A0	Akumajou Densetsu (Castlevania 3)
26	VRC6b	A0	A1	Esper Dream 2, Mouryou Senki Madara
85	VRC7	A4	(A5)	Lagrange Point (Z=A5 used for Sound only)
85	VRC7b	A3	(?)	Tiny Toon Adventures 2 (no Sound - maybe not a VRC7 ?)

Note that most mapper numbers are shared for two different connection variants, mapper 23 is even shared for different chip versions, the unused address bits are usually zero, so that software and hardware could, for example, reproduce $Z=(A0 \text{ OR } A2)$ for mapper 23.

VRC2a variant uses VROM bank outputs Bit1-7, all other VRCs use Bit0 and up.

PRG ROM Bank Registers (decoded by X or X.Y parts of the X.Y.Z address)

VRC1	VRC2	VRC3	VRC4	VRC6	VRC7	Expl.
-	-	F	-	8	-	Select 16K ROM at 8000h-BFFFh
8	8	-	8	-	8.0	Select 8K ROM at 8000h-9FFFh
A	A	-	A	-	8.1	Select 8K ROM at A000h-BFFFh
C	-	-	-	C	9.0	Select 8K ROM at C000h-DFFFh
-	FIX	FIX	FIX	-	-	Fixed 8K ROM at C000h-DFFFh (last-1 8K)
FIX	FIX	FIX	FIX	FIX	FIX	Fixed 8K ROM at E000h-FFFFh (last-0 8K)

The 16K banks of VRC3/VRC6 are Linear Addresses divided by 16K (not by 8K).

VRC4 can swap 8000h-9FFFh and C000h-DFFFh, see VRC4 Memory Control below.

VROM Bank Registers (VRC2,VRC4=2x4bit LSB/MSB, VRC6,VRC7=8bit)

VRC2,4	VRC6	VRC7	Expl.
B.0.0/1	D.0.0	A.0	Select 1K VROM bank at PPU 0000h-03FFh
B.1.0/1	D.0.1	A.1	Select 1K VROM bank at PPU 0400h-07FFh
C.0.0/1	D.1.0	B.0	Select 1K VROM bank at PPU 0800h-0BFFh
C.1.0/1	D.1.1	B.1	Select 1K VROM bank at PPU 0C00h-0FFFh
D.0.0/1	E.0.0	C.0	Select 1K VROM bank at PPU 1000h-13FFh
D.1.0/1	E.0.1	C.1	Select 1K VROM bank at PPU 1400h-17FFh
E.0.0/1	E.1.0	D.0	Select 1K VROM bank at PPU 1800h-1BFFh
E.1.0/1	E.1.1	D.1	Select 1K VROM bank at PPU 1C00h-1FFFh

Note that VRC2A uses Bit7-1 of the 2x4bit register, VRC2B uses Bit6-0, VRC4 and up use Bit6-0 (or all bits, Bit7-0, for large VROMs).

Lagrange Point contains VRAM instead VROM, the VRAM <is> map-able.

Salamander (VRC3) contains VRAM, which appears to be <not> map-able.

VRC1 VROM Bank Registers (5bit values, split into 4+1 bits)

9	Bit0: Mirroring, Bit1-2: MSBs of VROM banks, Bit3: Unused/zero
E	Lower 4bit of 4K VROM bank at PPU 0000h-0FFFh (MSB in Bit1 of Register 9)
F	Lower 4bit of 4K VROM bank at PPU 1000h-1FFFh (MSB in Bit2 of Register 9)

IRQ Registers (VRC1,VRC2=N/A, VRC3,VRC4=2x4bit, VRC6,VRC7=8bit)

VRC4	VRC6	VRC7	VRC3	Expl.
F.0.0/1	F.0.0	E.1	A/B	IRQ Reload value
F.1.0	F.0.1	F.0	C	IRQ Control (Bit0: 0=Disable, Bit1: 0=One-Shot)
F.1.1	F.1.0	F.1	D	IRQ Acknowledge (write any value to this address)

IRQ Reload is loaded to actual counter on write to IRQ Control Register, and on Counter overflow. If IRQ Control Register Bit1 is set, then the counter is automatically Restarted and Reloaded on Overflow.

The IRQ counter is incremented each 113.75 cycles (or each 114 cycles?), which is almost exactly once per NTSC-scanline, including for "hidden" scanlines during VBlank (only exception is VRC3, which is incremented every 256 cycles).

Mind that PAL/NTSC have different VBlank/Hblank times, and so, need different counter values. The VRC4 games Goal! and Moe Pro! appear to be bugged pirate ports from original Jaleco mapper to Konami-style mapper, these games do incorrectly acknowledge IRQs by writing zero to the IRQ Control register rather than by writing any value to the IRQ Acknowledge register, not sure if that works on real VRC4 hardware.

VRC4 Memory Control (VRC4 only - not VRC2,6,7)

9.0.1 (or 9.1.0?) Memory Control (2bit)

Bit1: PRG ROM Swap (0=Normal, 1=Swap) When swapped: Port 8.0.0 controls ROM at C000h-DFFFh, and ROM at 8000h-9FFFh becomes fixed, containing the 1st half of <last> 16K.

Bit0: Enable SRAM at 6000h-7FFFh (0=Disable, 1=Enable), VRC6 is having an equivalent function Bit7 of Name Table Control register.

Name Table Control

VRC1	VRC2,4	VRC6	VRC7	Expl.
9	9.0.0	B.1.1	E.0	Mirroring/Page Select

Mirroring selection VRC2,VRC4,VRC7: Bit1-0, VRC6: Bit3-2, VRC1: Bit0:

0	Two-Screen Vertical mirroring	(VRC1: Register 9, Bit0=0)
1	Two-Screen Horizontal mirroring	(VRC1: Register 9, Bit0=1)
2	Single-Screen BLK1	(VRC1: N/A)
3	Single-Screen BLK0	(VRC1: N/A)

On VRC6, bit5 additionally inverts the BLK-outputs, BLK0 then becomes BLK1, and vice versa, that also applies in Two-Screen modes, ie. the upper-left name table may be either BLK1 or BLK0.

And, on VRC6, Bit7 controls SRAM (0=Disable, 1=Enable).

VRC6 Sound Registers

[Mapper 26: Konami VRC6B - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND](#)

VRC7 OPL2 Sound Registers

9.1.0 Index Register

9.1.1 Data Register

[Mapper 85: Konami VRC7A/B - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND](#)

Unknown Registers

VRC3 - Salamander writes 00h to Port 8000h and 9000h.

Mapper 32: Irem G-101 - PRG/8K, VROM/1K, NT

This mapper is used on several Japanese titles by Irem, such as ImageFight 2.

9FFFh	Control Register (Bit1,0)
	Bit0 - Name Table ?Mirroring (0=Horizontal, 1=Vertical Mirroring)
	Bit1 - Port 8FFFh Switching Mode (see above)
8FFFh	When 9FFFh/Bit1=0:
	Select 8K ROM bank at 8000h-9FFFh (initially 1st 8K bank)
	Fixed 8K ROM bank at C000h-DFFFh (always 1st half of last 16K)
	When 9FFFh/Bit1=1:
	Fixed 8K ROM bank at 8000h-9FFFh (always 1st 8K bank)
	Select 8K ROM bank at C000h-DFFFh (initially probably 9FFFh/Bit1=0)
AFFFh	Select 8K ROM bank at A000h-BFFFh (initially 2nd 8K bank)
N/A	Fixed 8K ROM bank at E000h-FFFFh (always last 8K bank)
BFF0h	Select 1K VROM bank at PPU 0000h-03FFh
BFF1h	Select 1K VROM bank at PPU 0400h-07FFh
BFF2h	Select 1K VROM bank at PPU 0800h-0BFFh
BFF3h	Select 1K VROM bank at PPU 0C00h-0FFFh
BFF4h	Select 1K VROM bank at PPU 1000h-13FFh
BFF5h	Select 1K VROM bank at PPU 1400h-17FFh
BFF6h	Select 1K VROM bank at PPU 1800h-1BFFh
BFF7h	Select 1K VROM bank at PPU 1C00h-1FFFh

Mapper 33: Taito TC0190/TC0350 - PRG/8K, VROM/1K/2K, NT, IRQ

Used by Don Doko Don I, II, Flintstones - Rescue of Dino & Hoppy, Bakushou Jinsei Gekijou I, II, III, Insector X, Operation Wolf, Power Blazer, Golf Ko Open, Akira, Takeshi no Sengoku Fuuunji, Jetsons - Cogswell's Caper, Bubble Bobble 2, Captain Saver.

TC0190 and TC0350 are slightly different, one has IRQs, one doesn't. No idea which is which, so they'll be referenced as Type I and II.

Type I and II - Memory Banking Registers

8000h Select 8K ROM bank at 8000h-9FFFh (Type I: Bit6=Mirroring, see below)
8001h Select 8K ROM bank at A000h-BFFFh
N/A Fixed 16K ROM bank at C000h-FFFFh (always last 16K)
8002h Select 2K VROM bank at PPU 0000h-07FFh
8003h Select 2K VROM bank at PPU 0800h-0FFFh
A000h Select 1K VROM bank at PPU 1000h-13FFh
A001h Select 1K VROM bank at PPU 1400h-17FFh
A002h Select 1K VROM bank at PPU 1800h-1BFFh
A003h Select 1K VROM bank at PPU 1C00h-1FFFh

Type I - Mirroring

8000h Bit4-0:See above, Bit6:Mirroring (0=Vertical, 1=Horizontal Mirroring)

Ignore this bit if Type II registers are used.

Type II - Mirroring and IRQ

C000h IRQ Counter (incremented every scanline, paused during VBlank)
C001h IRQ Related (write same value as to C000h)
C002h IRQ Start/Enable (write any value)
C003h IRQ Acknowledge/Stop (write any value)
E000h Mirroring (Bit6) (0=Vertical, 1=Horizontal Mirroring)
E001h,E002h,E003h Unknown

Mapper 34: Nina-1 - PRG/32K, VROM/4K

Used by Impossible Mission II.

7FFEh Select 4K VROM bank at PPU 0000h-0FFFh (4bit)
7FFFh Select 4K VROM bank at PPU 1000h-1FFFh (4bit)
7FFDh Select 32K ROM bank at 8000h-FFFFh (1bit) (initially 1st bank)

Contains 8K WRAM at 6000h-7FFFh (not sure if last three bytes can be used).

Uses six TTL chips, 2x74LS173, 74LS139, 74LS133, 74LS74, and 74LS00, and a faux-lockout chip labelled 'NINA'.

Mapper 40: FDS-Port - Lost Levels

Used by Super Mario Bros 2 - Lost Levels.

8000h-9FFFh Disable/Reset IRQ counter (by writing any value)
A000h-BFFFh Enable/Start IRQ counter (by writing any value)
C000h-DFFFh Not Used

N/A	Fixed 8K ROM at 6000h-7FFFh (always bank 6)
N/A	Fixed 8K ROM at 8000h-9FFFh (always bank 4)
N/A	Fixed 8K ROM at A000h-BFFFh (always bank 5)
E000h-FFFFh	Select 8K ROM at C000h-DFFFh
N/A	Fixed 8K ROM at E000h-FFFFh (always bank 7, ie. last bank)

When enabled, IRQ generated after 4096 clock cycles (about 36 scanlines).

Uses different ports, but the features are about same as:

[Mapper 50: FDS-Port - Alt. Levels](#)

Mapper 41: Caltron 6-in-1

Used by Caltron 6-in-1 cartridge only.

6000h-67FFh	Main Control Register (decoded by ADDRESS lines A0-A5)
A2-A0	Select 32K ROM at 8000h-FFFFh
A2	MSB of above bank number - also enables second register
A4-A3	Upper two bits of 8K VROM bank at 0000h-1FFFh
A5	Name Table (0=Vertical, 1=Horizontal Mirroring)
8000h-FFFFh	Auxiliary CHR control (decoded by DATA lines D0-D1)
	This register is write-protected when above A2=0 (!)
D1-D0	Lower two bits of 8K VROM bank at 0000h-1FFFh

When the NES is switched on, or the reset button is pressed, both registers are cleared to 00h, done by a cool little diode / RC circuit on the PHI2 line.

Mapper 42: FDS-Port - Mario Baby

Used only by one game: A pirate copy of Bio Miracle Bouquette Upa, renamed to Mario Baby, and modified to work as cartridge (instead FDS floppy disk).

E000h-FFFCh	Select 8K ROM at 6000h-7FFFh
N/A	Fixed 32K ROM at 8000h-FFFFh (always last 32K)
E001h-FFFDh	Select mirroring (Bit3: 0=Vertical, 1=Horizontal Mirroring)
E002h-FFFEh	IRQ Control (Bit1: 0=Disable/Reset, 1=Enable/Start)
E003h-FFFFh	Not used

When enabled, IRQ generated after 24576 clock cycles (about 216 scanlines).

These ports are mirrored from E000h-FFFFh, every 4 bytes.

Consists of a whopping 11 chips - 9 TTL/CMOS, 8K RAM, and 128K ROM.

Mapper number 42 is also assigned to Ai Senshi Nicol. In short, doing this:

8000h	Select 8K VROM at PPU 0000h-1FFFh
F000h	Select 8K ROM at 6000h-7FFFh
N/A	Fixed 32K ROM at 8000h-FFFFh (always last 32K)

However, it is doing some odd initialization, writing to Port E000h (index, 1-5 used), and Port F000h (data for index 1-5, looks like 8K ROM banks at 8000h, A000h, C000h, 6000h, E000h). Note that: A) the last 32K seem to be always mapped to 8000h-FFFFh. B) each final write to F000h seems to be done with index 4. If that behaviour doesn't change later on in the game, then Port F000h (or even E000h) could be interpreted exactly as for Mario Baby.

Mapper 43: X-in-1

Used by 150-in-1 (a fake containing 56 games, plus some cheat modes, chip 0 is 1024K, chip 1 is 512K,

chip 2-3 are not installed).

8000h-FFFFh	Memory Control	(Write any data, port decoded by address lines)
A7-A0	Select 32K ROM Bank	(From currently selected Chip)
A9-A8	Select ROM Chip	(Empty bus if selected chip not installed)
A10	Not used	(Always zero)
A11	Bank Mode	(0=32K, 1=16K; Lower/Upper half via A12)
A12	Select 16K ROM Bank	(0=Lower, 1=Upper) (Should be zero in 32K mode)
A13	Mirroring	(0=Vertical, 1=Horizontal Mirroring)
A14	Not used	(Always zero)

Initially 1st 32K selected. The cartridge includes 8K VRAM.

Mapper 44: 7-in-1 MMC3 Port A001h

Used by Super Big 7-in-1.

A001h - Select 128K ROM/VROM Block (0..5) or last 256K ROM/VROM Block (6)

Block 0 seems to be required on power-up.

The rest of both mappers is same as MMC3 (for the selected block of memory),

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

Mapper 45: X-in-1 MMC3 Port 6000hx4

Used in Super 3-in-1, Super 4-in-1, Super 8-in-1, Hero 8-in-1, Super 13-in-1, and 1000000-in-1 (a fake with 1000000 duplicated/nonsense titles).

Configuration value initialized by each FOUR writes to Port 6000h.

6000h 1st write	- Configuration Bits 0-7
6000h 2nd write	- Configuration Bits 8-15
6000h 3rd write	- Configuration Bits 16-23
6000h 4th write	- Configuration Bits 24-31

The meaning of the 32 configuration bits is:

Bit7-0	VRAM base in 1K steps
Bit15-8	ROM base in 8K steps
Bit19-16	VRAM mask in 1K steps, Mask=(2 SHL (X AND 7))+(X AND 8)/8
Bit23-20	VRAM base in 256K steps
Bit29-24	ROM mask in 8K steps, Mask=(3Fh AND (NOT X))
Bit30	LOCK (set when menu selection completed, probably locks Port 6000h)
Bit31	???

Memory selections are Bank=((Mmc3Bank AND ConfigMask) OR ConfigBase). For MMC3 Registers 0 and 1 (map 2x1K banks), above formula applies to both banks, ie. VROM Mask=0 maps same 1K bank twice (instead two continuous 1K banks).

128K Block 3 seems to be required on power-up, ie. the entry point is at the end of 512K ROMs, or in the middle of 1024K ROMs. The MMC3 Port A001h apparently can write-protect Port 6000h (just like it disables SRAM at 6000h-7FFFh). At least some carts have at least 2K SRAM (Mario 3 in Super 8-in-1 uses RAM at 7800h-7FFFh).

The rest of the mapper is same as MMC3 (for the selected block of memory),

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

Mapper 46: 15-in-1 Color Dreams

Used by Rumble Station 15-in-1, which contains 15 Color Dreams games.

```
6000h-7FFFh Multicart Memory Control
  Bit0-3 Select 64K ROM Block (initially 1st bank) (always same as below)
  Bit4-7 Select 64K VROM Block (initially 1st bank) (always same as above)
8000h-FFFFh Memory Control (selection within current 64K block)
  Bit1 Select 32K ROM bank at 8000h-FFFFh (initially 1st bank)
  Bit6-4 Select 8K VROM bank at PPU 0000h-1FFFh (initially 1st bank)
```

Port 8000h-FFFFh is same as Mapper 11, except that it is limited to 64K, and except that it doesn't seem to have bus-conflicts (the menu always uses Port 8888h, regardless of underlying ROM content). In some games ROM/VROM is less than 64K, so some memory areas are unused.

[Mapper 11: Color Dreams - PRG/32K, VROM/8K](#)

Mapper 47: 2-in-1 MMC3 Port 6000h

Used by 2-in-1 cart "Super Spike V'Ball + Nintendo World Cup".

```
6000h Select 1st or 2nd half of ROM/VROM (0 or 1)
```

The rest of the mapper is same as MMC3 (for the selected block of memory),

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

The cartridge doesn't have SRAM, but the MMC3 Port A001h apparently can write-protect Port 6000h (just like it could disable SRAM at 6000h-7FFFh).

Mapper 48: Taito TC190V

Reportedly "Tatio TC190V" used by "FlintStone".

Sounds like the Type II (or Type I?) variant of Mapper 33:

[Mapper 33: Taito TC0190/TC0350 - PRG/8K, VROM/1K/2K, NT, IRQ](#)

Mapper 49: 4-in-1 MMC3 Port 6xxxh

Used by Super HIK 4-in-1.

ROM/VROM are split into 128K blocks each, done as such:

```
[6000h]=01h ;init
[6800h]=00h ;game 0 + [6808h]=08h crashes ?
[6841h]=41h ;game 1
[6881h]=81h ;game 2
[68C1h]=C1h ;game 3
```

The rest of the mapper is same as MMC3 (for the selected block of memory),

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

The cartridge doesn't have SRAM, but the MMC3 Port A001h apparently can write-protect Ports at 6000h-7FFFh (just like it could disable SRAM at 6000h-7FFFh).

Mapper 50: FDS-Port - Alt. Levels

Used by Super Mario Bros 2 - Alt. Levels.

```
4022h Select 8K ROM at C000h-DFFFh
  Bit0 and/or Bit3 ZERO Bank 0
```

	Bit0 and/or Bit3 SET	Bank 0Ch (or maybe INCREMENT bank number?)
	Other Bits	Unknown
4122h	IRQ Control	
	Bit0 and/or Bit1 ZERO	Disable/Acknowledge
	Bit0 and/or Bit1 SET	Enable/Start
N/A	Fixed 8K ROM at 6000h-7FFFh	(always bank 0Fh, ie. last bank)
N/A	Fixed 8K ROM at 8000h-9FFFh	(always bank 08h)
N/A	Fixed 8K ROM at A000h-BFFFh	(always bank 09h)
N/A	Fixed 8K ROM at E000h-FFFFh	(always bank 0Bh)

When enabled, IRQ generated after 4096 clock cycles (about 36 scanlines).

Uses different ports, but the features are about same as:

[Mapper 40: FDS-Port - Lost Levels](#)

Mapper 51: 11-in-1

Used by 11-in-1, containing ball games, like "soccer ball", and "golf ball".

6000h	Mode Register	
	Bit1 ROM Block Size (0=128K Mode, 1=32K Mode)	
	Bit4 Unknown	
8000h	Base Address in 32K Steps (X) (0-0Fh)	
	32K Mode: Select 32K Bank (X) at 8000h-FFFFh	(initially 1st 32K bank)
	128K Mode: Select 16K Bank (X*2 OR 07h) at C000h-FFFFh	
	And: Select 8K Bank (X*4 OR 23h) at 6000h-7FFFh	(for FDS ports)
C000h	Lower 16K Select (Y) (0-1Fh)	(128K Mode only, UNROM-style)
	128K Mode: Select 16K Bank (Y*2 OR Y/10h) at 8000h-BFFFh	

The cartridge does have 8K VRAM, even though there's a ROM-image around in which somebody has incorrectly included a copy of above 8K VRAM as "8K VROM".

Mapper 52: 7-in-1 MMC3 Port 6800h with SRAM

Used by Mario Party 7-in-1 (or short, Mari7in1).

"It's MMC3 and an extra bank control register. There is 1Mbyte of PRG ROM and 1Mbyte of CHR ROM on this cart. Interestingly, all the games appear to be NTSC, except SMB2. For some reason, this is the PAL version! It consists of 1 6264 8K RAM chip (for WRAM), and 3 glop-tops. 2 are 1Mbyte ROMs while the remaining chip is the mapper."

6800h	Bank Control Byte	
	Bit7 Not used	
	Bit6 VROM Bank Size (0=256K, 1=128K)	
	Bit5,2,4 VROM 128K Bank (Bit4 not used in 256K CHR mode)	
	Bit3 PRG ROM Bank Size (0=256K, 1=128K)	
	Bit2,1,0 PRG ROM 128K Bank (Bit0 not used in 256K PRG mode)	

Note: Bit2 is both MOST significant PRG bit, and MIDDLE significant CHR bit.

After a reset, this register is 00h. It can only be written once.

To reset it and allow another write, you must reset the console.

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

Mapper 56: Pirate SMB3

Used only by a pirate copy of Super Mario Bros. 3.

8000h Unknown (always 08h) (maybe counter LSBs, if any)
 9000h Bit7-4 of 16bit IRQ counter
 A000h Bit11-8 of 16bit IRQ counter
 B000h Bit15-12 of 16bit IRQ counter
 C000h IRQ Anable (FFh=Enable, 00h=Disable)
 D000h IRQ Acknowledge (Always write FFh, or EFh)
 E000h Ignore - MMC3-index (Port 8000h) relicts redirected to E000h
 F000h Select 8K ROM at 8000h-9FFFh
 F001h Select 8K ROM at A000h-BFFFh
 F002h Select 8K ROM at C000h-DFFFh
 N/A Fixed 8K ROM at E000h-FFFFh (always last bank)
 F003h Unknown (always 10h)
 F400h Select 1K VROM at PPU 0000h-03FFh
 F401h Select 1K VROM at PPU 0400h-07FFh
 F402h Select 1K VROM at PPU 0800h-0BFFh
 F403h Select 1K VROM at PPU 0C00h-0FFFh
 F404h Select 1K VROM at PPU 1000h-13FFh
 F405h Select 1K VROM at PPU 1400h-17FFh
 F406h Select 1K VROM at PPU 1800h-1BFFh
 F407h Select 1K VROM at PPU 1C00h-1FFFh

The cartridge often uses silly mirrors like C5A7h, mask 8000h-EFFFh by F000h, and F000h-FFFFh by F407h. The IRQ counter is incremented every clock cycle.

Mapper 57: 6-in-1

Used by Game Star 6-in-1 GK-L01A, 6-in-1 GK-L02A, and 54-in-1 GK-54.

8000h Extra Port for CNROM Games in 2nd 64K of VROM
 Bit2-0 Select 8K VROM at PPU 0000h-1FFFh (ORed with value in Port 8800h)
 Bit5-3 Not used (zero)
 Bit6 Must be set for Second 64K Block of VROM
 Bit7 Must be set for First 64K Block of VROM
 8800h Main Port
 Bit2-0 Select 8K VROM at PPU 0000h-1FFFh (ORed with value in Port 8000h)
 Bit3 Mirroring (0=Vertical, 1=Horizontal Mirroring)
 Bit4 ROM Size (0=16K; Bank X twice, 1=32K; Bank X and X+1)
 Bit7-5 Select 16K ROM at 8000h-BFFFh and C000h-FFFFh (X)

All carts have 128K ROM and 128K VROM and contain 6 games (even "54-in-1").

Mapper 58: X-in-1

Used by 68-in-1 (a fake containing only 8 games).

C000h-FFFFh Memory Control (Write any data, port decoded by address lines)
 A2-A0 Select 16K ROM Bank at 8000h-BFFFh and C000h-FFFFh (X)
 A5-A3 Select 8K VROM Bank at PPU 0000h-1FFFh
 A6 ROM Size (0=32K; Bank X and X+1, 1=16K; Bank X twice)
 A7 Mirroring (0=Vertical, 1=Horizontal Mirroring)
 A12-A8 Unknown (Usually 0,/A6,0,/A6,A5,A3, except in yie-ar-kung-fu)

Note: Study and Game 32-in-1 declared as "Mapper 58" should be Mapper 241, [Mapper 241: X-in-1 Education](#)

Mapper 61: 20-in-1

Used by 20-in-1.

8000h-FFFFh Memory Control (Write any data, port decoded by address lines)
 A3-0 Select 32K ROM Bank at 8000h-FFFFh
 A4 Bank Size (0=32K, 1=16K; only lower/upper half via Bit5)
 A5 Select lower/upper half of selected 32K bank (in 16K mode)
 A7 Mirroring (0=Vertical, 1=Horizontal Mirroring)
 A6,A8-A14 Not used (always 0)

There's also a version of the same cartridge with slightly different mapper:

[Mapper 231: 20-in-1](#)

Mapper 62: X-in-1

Used by 700-in-1 (a fake containing only somewhat 50-100 games).

8000h-BFFFh Memory Control (Decoded by address AND data lines)
 A4-A0,D1-D0 Select 8K VROM at PPU 0000h-1FFFh
 A5 ROM Size (0=32K; Bank X-1 and X, 1=16K; Bank X twice)
 A7 Mirroring (0=Vertical, 1=Horizontal Mirroring)
 A6,A13-A8 Select 16K ROM at 8000h-BFFFh and C000h-FFFFh (X)
 A14 Always 0 ?

ROM Bank lower bit (A8) should be always SET in 32K Mode. Initially 1st 32K.

Mapper 64: Tengen RAMBO-1 - PRG/8K, VROM/2K/1K, NT

This mapper is used on several U.S. unlicensed titles by Tengen. They include Shinobi, Klax, and Skull & Crossbones.

8000h Index/Control (6bit)
 Bit7 CHR Address Select (0=Normal, 1=Address Areas XOR 1000h)
 Bit6 PRG Address Select (0=Normal, 1=Address Areas plus 2000h)
 Bit3-0 Command Number (Note: Index 0-7 same as for MMC3)
 0 - Select 2x1K VROM at PPU 0000h-07FFh (or 1000h-17FFh, if Bit7=1)
 1 - Select 2x1K VROM at PPU 0800h-0FFFh (or 1800h-1FFFh, if Bit7=1)
 2 - Select 1K VROM at PPU 1000h-13FFh (or 0000h-03FFh, if Bit7=1)
 3 - Select 1K VROM at PPU 1400h-17FFh (or 0400h-07FFh, if Bit7=1)
 4 - Select 1K VROM at PPU 1800h-1BFFh (or 0800h-0BFFh, if Bit7=1)
 5 - Select 1K VROM at PPU 1C00h-1FFFh (or 0C00h-0FFFh, if Bit7=1)
 6 - Select 8K ROM at 8000h-9FFFh (or A000h-BFFFh, if Bit6=1)
 7 - Select 8K ROM at A000h-BFFFh (or C000h-DFFFh, if Bit6=1)
 F - Select 8K ROM at C000h-DFFFh (or 8000h-9FFFh, if Bit6=1)
 N/A - Fixed 8K ROM at E000h-FFFFh (always last 8K bank)
 8 - Select 1K VROM page at PPU 0400h
 9 - Select 1K VROM page at PPU 0C00h
 8001h Data Register (indexed via Port 8000h)
 A000h ?Mirroring Select (Bit0: 0=Horizontal, 1=Vertical Mirroring)
 No confidence in the accuracy of this information.

At reset, all four 8K banks are set to the last 8K bank in the cart.

Carts with VROM initially map the first 8K of VROM to PPU 0000h-1FFFh on reset. Carts without VROM should always have 8K of VRAM at PPU 0000h-1FFFh.

Mapper 65: Irem H-3001 - PRG/8K, VROM/1K, NT, IRQ

Used by Daiku no Gensan 2, Kaiketsu Yanchamaru 3, and Spartan X 2.

Note: Ai Sensei no Oshiete declared as "Mapper 65" is maybe a Konami mapper?

9000h Unknown
 9001h Unknown

9003h,9004h IRQ Control (not sure about difference between 9003h/9004h)
 (00h=Disable IRQ, C0h=Enable IRQ, other values unknown)
 9005h IRQ Counter MSB of decrementing 16bit counter
 9006h IRQ Counter LSB of decrementing 16bit counter
 B000h Select 1K VROM bank at PPU 0000h-03FFh
 B001h Select 1K VROM bank at PPU 0400h-07FFh
 B002h Select 1K VROM bank at PPU 0800h-0BFFh
 B003h Select 1K VROM bank at PPU 0C00h-0FFFh
 B004h Select 1K VROM bank at PPU 1000h-13FFh
 B005h Select 1K VROM bank at PPU 1400h-17FFh
 B006h Select 1K VROM bank at PPU 1800h-1BFFh
 B007h Select 1K VROM bank at PPU 1C00h-1FFFh
 8000h Select 8K ROM bank at 8000h-9FFFh (initially 1st half of 1st 16K)
 A000h Select 8K ROM bank at A000h-BFFFh (initially 2nd half of 1st 16K)
 C000h Select 8K ROM bank at C000h-DFFFh (initially 1st half of last 16K)
 N/A Fixed 8K ROM bank at E000h-FFFFh (always 2nd half of last 16K)

Mapper 66: GNROM - PRG/32K, VROM/8K

This mapper is used on several Japanese titles, such as Dragon Ball, and on U.S. titles such as Gumshoe and Dragon Power.

8000h-FFFFh Memory Control (2x2bits)
 Bit1-0 Select 8K VROM bank at PPU 0000h-1FFFh (initially 1st bank)
 Bit5-4 Select 32K ROM bank at 8000h-FFFFh (initially 1st bank)

This mapper is used on the DragonBall (NOT DragonBallZ) NES game.

Mapper 67: Sunsoft3 - PRG/16K, VROM/2K, IRQ

Used by Fantasy Zone 2.

8000h IRQ Acknowledge (write any data to this address)
 8800h-8FFFh Select 2K VROM bank at PPU 0000h-07FFh
 9800h-9FFFh Select 2K VROM bank at PPU 0800h-0FFFh
 A800h-AFFFh Select 2K VROM bank at PPU 1000h-17FFh
 B800h-BFFFh Select 2K VROM bank at PPU 1800h-1FFFh
 C800h-CFFFh IRQ Counter (two writes: 1st=MSB, 2nd=LSB)
 (16bit decrementing clock cycle counter)
 D800h-DFFFh IRQ Control (Bit4: 0=Disable, 1=Enable)
 E800h-EFFFh No info - maybe Mirroring control ?
 F800h-FFFFh Select 16K ROM bank at 8000h-BFFFh (initially 1st bank)
 N/A Fixed 16K ROM bank at C000h-FFFFh (always last bank)

Mapper 68: Sunsoft4 - PRG/16K, VROM/2K, NT-VROM

This mapper is used by After Burner I and II, and by Maharaja.

8000h Select 2K VROM bank at PPU 0000h-07FFh
 9000h Select 2K VROM bank at PPU 0800h-0FFFh
 A000h Select 2K VROM bank at PPU 1000h-17FFh

Mapper 69: Sunsoft5 FME-7 - PRG/8K, VROM/1K, NT ctrl, SRAM, IRQ

E000h Select 1K VROM bank at PPU 1800h-1FFFh
 C000h Select 1K VROM bank as BLK0 (in VROM Mode) (from LAST 128 banks)
 D000h Select 1K VROM bank as BLK1 (in VROM Mode) (from LAST 128 banks)
 E000h Name Table Control
 Bit4 Name Table VROM Mode (0=VRAM, 1=VROM via Port C000h/D000h)
 Bit0 Name Table Mirroring (0=Horizontal, 1=Vertical Mirroring)

This mapper is used by Batman, Hebereke, Pyokotan no Da Meiro, Barcode World, Gremm 2, Honoo no Doukyuuji 1 and 2 and Gimmick.
 F000h Select 16K ROM bank at 8000h-BFFFh (initially 1st bank)
 N/A Fixed 16K ROM bank at C000h-FFFFh (always last bank)

8000h Index Register (4bit)

- 0 - Select 1K VROM at PPU 0000h-03FFh
- 1 - Select 1K VROM at PPU 0400h-07FFh
- 2 - Select 1K VROM at PPU 0800h-0BFFh
- 3 - Select 1K VROM at PPU 0C00h-0FFFh
- 4 - Select 1K VROM at PPU 1000h-13FFh
- 5 - Select 1K VROM at PPU 1400h-17FFh
- 6 - Select 1K VROM at PPU 1800h-1BFFh
- 7 - Select 1K VROM at PPU 1C00h-1FFFh
- 8 - Select 8K ROM/RAM at 6000h-7FFFh
 - Bit6=0 --> Select 8K ROM (Page number in bit5-0)
 - Bit6=1, Bit7=1 --> Select 8K SRAM
 - Bit6=1, Bit7=0 --> Select 8K "pseudo-random numbers?"
- 9 - Select 8K ROM at 8000h-9FFFh
- A - Select 8K ROM at A000h-BFFFh
- B - Select 8K ROM at C000h-DFFFh
- C - Select Mirroring
 - 0 Two-Screen, Vertical Mirroring
 - 1 Two-Screen, Horizontal Mirroring
 - 2 One-Screen, BLK0
 - 3 One-Screen, BLK1
- D - IRQ control (00h=Disable, 81h=Enable, other values?)
- E - IRQ LSB of decrementing clock cycle counter
- F - IRQ MSB of decrementing clock cycle counter
- N/A - Fixed 8K ROM at E000h-FFFFh (always last 8K bank)

A000h Data Register (indexed via Port 8000h)

Mapper 70: Bandai - PRG/16K, VROM/8K, NT

Used by Taito's Arkanoid 2, and various Bandai games: Space Shadow, Kamen Rider Club, Saint Seiya, Pocket Zaurus, Gegege no Kitarou 2, and two Family Trainer games.

C000h-C0FFh Memory Control

- Bit7 Name Table Select (0/1 = BLK0/BLK1) (One-Screen Mode only)
- Bit6-4 Select 16K ROM at 8000h-BFFFh
- Bit3-0 Select 8K VROM at PPU 0000h-1FFFh

Bus-Conflicts, memory at C000h-C0FFh should be filled by 00h-FFh.

Mapper 71: Camerica - PRG/16K

This mapper is used on Camerica's unlicensed NES carts, including Firehawk and Linus Spacehead.

8000h-BFFFh Unknown

- C000h-FFFFh Select 16K ROM bank at 8000h-BFFFh (initially 1st bank)
- N/A Fixed 16K ROM bank at C000h-FFFFh (always last bank)

All carts using it have 8K of VRAM at PPU \$0000.

Many ROMs from these games are incorrectly defined as mapper 2.

Mapper 72: Jaleco Early Mapper 0 - PRG-LO, VROM/8K

Used by Pinball Quest, Pro Tennis, Pro Judo.

8000h-FFFFh Memory Control

- Bit7-6 Function Select
 - 0 Confirm Selection
 - 1 Select 8K VROM bank at PPU 0000h-1FFFh
 - 2 Select 16K ROM bank at 8000h-BFFFh (lower half of PRG memory)

3 Reserved (would probably select both PRG+VROM)
Bit5-4 Not used
Bit0-3 ROM or VROM Bank Number for above Selection

Bus-conflicts. Example: To select PRG Bank 7, first write 87h, then 07h.
Same as Mapper 92, except that this one maps the LOWER half of PRG memory.

Not sure if that makes sense, but it appears to consist of three latches, latch1 directly accessed from CPU, the other latches loaded from the latch1 on high-to-low transitions in bit6/7.

Mapper 73: Konami VRC3 - PRG/16K, IRQ

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

Mapper 74: Whatever MMC3-style

Reportedly "Taiwan MMC3 -Variant Mapper#0" used by "KidNiKi3J(hacked)". Personally, I have a ROM-image named "Ji Jia Zhan Shi", which may or may not be same as "KidNiKi3J(hacked)".

Anyways, that ROM-image seems to be basically MMC3-compatible,

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

except that it gets wrong anytime when selecting ROM bank number 14h via Register 7, don't know what is/should be happening there (?)

Mapper 75: Jaleco SS8805/Konami VRC1 - PRG/8K, VROM/4K

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

Mapper 76: Namco 109 - PRG/8K, VROM/2K

Used by Digital Devil / Megami Tensei.

8000h Index/Control (3bit)
Bit2-0 Command Number
0 - Not used
1 - Not used
2 - Select 2K VROM at PPU 0000h-07FFh
3 - Select 2K VROM at PPU 0800h-0FFFh
4 - Select 2K VROM at PPU 1000h-17FFh
5 - Select 2K VROM at PPU 1800h-1FFFh
6 - Select 8K ROM at 8000h-9FFFh
7 - Select 8K ROM at A000h-BFFFh
N/A - Fixed 16K ROM at C000h-FFFFh (always last bank)
8001h Data Register (Indexed via Port 8000h)

Mapper 77: Irem - PRG/32K, VROM/2K, VRAM 6K+2K

Used by Napoleon Senki.

8000h-FFFFh Memory Control

Bit0-1 Select 32K ROM bank at 8000h-FFFFh
Bit2-3 Not used
Bit4-7 Select 2K VROM bank at PPU 0000h-07FFh
6K VRAM at PPU 0800h-1FFFh (ie. upper 6K of Pattern Tables are VRAM)
2K VRAM at PPU 2800h-2FFFh (ie. uses Four-Screen Name Tables)

Bus-conflicts.

Mapper 78: Irem 74HC161/32 - PRG/16K, VROM/8K

Used by Holy Diver, and Cosmo Carrier (Uchuusen). The two games seem to be using different/incompatible mapper circuits for name table control?

8000h-FFFFh Memory Control
Bit2-0 Select 16K ROM bank at 8000h-BFFFh (initially 1st bank)
N/A Fixed 16K ROM bank at C000h-FFFFh (always last bank)
Bit3 Name Table Control
Jaleco/Cosmo Carrier: One-Screen (0=BLK0, 1=BLK1)
Irem/Holy Diver: Two-Screen (0=Horizontal, 1=Vertical Mirroring)
Bit7-4 Select 8K VROM bank at PPU 0000h-1FFFh

Mapper 79: AVE Nina-3 - VROM/8K

[See also]

[Mapper 113: Sachen/Hacker/Nina](#)

Made by American Video Entertainment (AVE), used by Krazy Kreatures, Double Strike, etc.

4100h Bit1-0 Select 8K VROM bank at PPU 0000h-1FFFh

Port decoded as A14=A8=HIGH, A15=A13=LOW, ie. with mirrors at 4100h-41FFh, 4300h-43FFh ... 5F00h-5FFFh. Contains 74LS175, 74LS138, and "Nina"-lockout chip.

Mapper 80: Taito X-005 - PRG/8K, VROM/2K/1K, NT

Used by Fudou Myouou Den (Demon Sword), Kyonshiizu 2, Mirai Shinwa Jarvas, Taito Grand Prix - Eikou heno License, Yamamura Misa Suspense - Kyouto Ryou no Tera Satsujin, Minelvaton Saga.

7EF0h Select 2x1K VROM at PPU 0000h-07FFh (Bit7: Name Table, see below)
7EF1h Select 2x1K VROM at PPU 0800h-0FFFh (Bit7: Name Table, see below)
7EF2h Select 1K VROM at PPU 1000h-13FFh
7EF3h Select 1K VROM at PPU 1400h-17FFh
7EF4h Select 1K VROM at PPU 1800h-1BFFh
7EF5h Select 1K VROM at PPU 1C00h-1FFFh
7EF6h Unknown (usually FFh, 01h, or 00h)
7EF8h SRAM Enable (A3h=Enable, FFh=Disable)
7EFAh Select 8K ROM 8000h-9FFFh
7EFCh Select 8K ROM A000h-BFFFh
7EFEh Select 8K ROM C000h-DFFFh
N/A Fixed 8K ROM E000h-FFFFh (always last bank)
7EF7h, 7EF9h Not used
7EFBh, 7EFDh, 7EFFh Dupes of 7EFAh, 7EFCh, 7EFEh used by Kyonshiizu 2 only
7F00h-7FFFh SRAM Area (seems to be only 256 bytes or less used)

Bit7 of 7EF0h and Bit7 of 7EF1h are somehow related to Name Tables: Most carts use Vertical Mirroring, these carts always have both bits cleared. Demon Sword uses One-Screen mode, either both bits cleared (BLK0), or both bits set (BLK1).

Mapper 81: AVE Nina-6

Made by American Video Entertainment (AVE), used by Deathbots, Mermaids of Atlantis, etc.

No info. For Deathbots, see:

[Mapper 113: Sachen/Hacker/Nina](#)

Also, presumably mis-numbered "Mapper 81 - Taito C075"

Reportedly "Tatio C075" used by "(many Japanese title from tatio)".

Don't have a ROM-image that is assigned as Mapper 81. Don't know which titles are using it. Maybe meant to be Taito's Arkanoid 2, ie. this mapper:

[Mapper 70: Bandai - PRG/16K, VROM/8K, NT](#)

Mapper 82: Taito X1-17 - PRG/8K, VROM/2K/1K

Used by SD Keiji Blader, and Kyuukyoku Harikiri 1, 2, 3.

```
7EF0h Select 2x1K VROM at PPU 0000h-07FFh (or 1000h-17FFh if swapped)
7EF1h Select 2x1K VROM at PPU 0800h-0FFFh (or 1800h-1FFFh if swapped)
7EF2h Select 1K VROM at PPU 1000h-13FFh (or 0000h-03FFh if swapped)
7EF3h Select 1K VROM at PPU 1400h-17FFh (or 0400h-07FFh if swapped)
7EF4h Select 1K VROM at PPU 1800h-1BFFh (or 0800h-0BFFh if swapped)
7EF5h Select 1K VROM at PPU 1C00h-1FFFh (or 0C00h-0FFFh if swapped)
7EF6h Swap PPU 0000h-0FFFh / 1000h-1FFFh (Bit1: 0=Normal, 1=Swap)
7EF7h SRAM .... CAh,00h,01h,40h
7EF8h SRAM .... 69h,00h,40h
7EF9h SRAM .... 84h,00h,40h
7EFAh Select 8K ROM 8000h-9FFFh (Bit7-2)
7EFBh Select 8K ROM A000h-BFFFh (Bit7-2)
7EFC h Select 8K ROM C000h-DFFFh (Bit7-2)
N/A Fixed 8K ROM E000h-FFFFh (unknown?)
7EFDh SRAM .... FFh
7EFEh SRAM .... FFh,07h
7EFFh SRAM .... FFh
6000h-7FFFh SRAM Area (probably 8K size, at least 6000h-73xxh used)
```

Mapper 83: Cony

There are different Cony variants for cartridges of different size:

```
Cony (A) 128K+256K Fatal Fury 2
Cony (B) 256K+512K World Heroes 2
Cony (C) 4x256K+4x256K Dragon Ball Z 4-in-1
Also used by Garou Densetsu Special?
```

Cony (A) and (C) only - 8bit bank numbers (256x1K=256K):

```
8310h Select 1K VROM at PPU 0000h-03FFh (in current 256K block)
8311h Select 1K VROM at PPU 0400h-07FFh (in current 256K block)
8312h Select 1K VROM at PPU 0800h-0BFFh (in current 256K block)
8313h Select 1K VROM at PPU 0C00h-0FFFh (in current 256K block)
8314h Select 1K VROM at PPU 1000h-13FFh (in current 256K block)
8315h Select 1K VROM at PPU 1400h-17FFh (in current 256K block)
8316h Select 1K VROM at PPU 1800h-1BFFh (in current 256K block)
8317h Select 1K VROM at PPU 1C00h-1FFFh (in current 256K block)
```

Cony (B) only - 8bit bank numbers (256x2K=512K):

```
8310h Select 2K VROM at PPU 0000h-07FFh
8311h Select 2K VROM at PPU 0800h-0FFFh
```

8316h Select 2K VROM at PPU 1000h-17FFh
8317h Select 2K VROM at PPU 1800h-1FFFh

Cony (A) only - 4bit bank numbers (16x8K=128K):

8300h Select 8K ROM at 8000h-9FFFh
8301h Select 8K ROM at A000h-BFFFh
8302h Select 8K ROM at C000h-DFFFh
N/A Fixed 8K ROM at E000h-FFFFh (always last bank)

Cony (B) and (C) only - 4bit bank numbers (16x16K=256K):

8000h Select 16K ROM at 8000h-BFFFh (in current 256K block)
N/A Fixed 16K ROM at C000h-FFFFh (last bank in current 256K block)

Cony (A) and (B) and (C):

8200h IRQ Counter LSB, writing to this address acknowledges IRQs
8201h IRQ Counter MSB, writing to this address starts counting

Cony (A) and (B) only:

8100h IRQ Control (Bit7=Enable IRQs) (other bits unknown) Bit7, unlike C
5000h Unknown, program reads from this address

Cony (C) only:

8100h IRQ Control (Bit1=Enable IRQs) (other bits unknown) Bit1, unlike A/B
B000h Select 256K ROM/VROM Windows (upper two address bits)
Bit0-3 Unknown
Bit4,6 Bit0 of 256K Block Number
Bit5,7 Bit1 of 256K Block Number
Used values are 00h,50h,A0h,F0h. Other values could probably select separate 256K banks for ROM/VROM. The ROM selection also affects the "fixed" 16K at C000h-FFFFh (last bank in current 256K block).
B0FFh Probably same as B000h
B1FFh Probably same as B000h
510Xh Unknown, program reads/writes to/from this address
430Xh Unknown, program reads from this address

Mapper 84: Whatever

No info. Reportedly "PC-SMB2J" used by "SMBJ2".

Don't have a ROM-image, unless it is meant to be same as Mapper 40 or 50:

[Mapper 40: FDS-Port - Lost Levels](#)

[Mapper 50: FDS-Port - Alt. Levels](#)

Mapper 85: Konami VRC7A/B - PRG/16K/8K, VROM/1K, NT, IRQ, SOUND

VRC7 General Memory and IRQ Registers

[Mapper 21-26,73,75,85: Konami VRC Mappers](#)

VRC7 Sound Registers

The sound generation is done using FM synthesis, so the music sounds like "Adlib" OPL2 music. All sound registers are accessed through only two physical registers.

9010h (aka 9.1.0) Index register
9030h (aka 9.1.1) Data Register

There are 6 channels, each containing three registers, and 8 custom instrument control registers.

Index 10h-15h - Channel 0-5, Frequency LSB

Bit7-0 Lower 8bit of 9bit Frequency (f; 0-1FFh)

Index 20h-25h - Channel 0-5, Frequency MSB, Octave, Trigger

Bit7-5 Unknown
Bit4 Channel trigger
Bit3-1 Octave Select (o; 0-7)
Bit0 Upper 1bit of 9bit Frequency (f; 0-1FFh)

Frequency is calculated as: $F = 49722\text{Hz} * f / 2^{(19-o)}$

Index 30h-35h - Channel 0-5, Instrument and Volume

Bit7-4 Instrument number (0=Custom, 1-0Fh=Fixed Instruments)
Bit3-0 Volume

Index 00h-07h - Custom Instrument (Instrument 0)

Note: I will not provide too extensive documentation of the instrument registers since their functions are identical to those of the OPL2 chip, commonly found on Adlib/Soundblaster/compatible cards, and there is a lot of information out on how to program these. I will use terminology similar to that found in said documents. My VRC7 "emulator" test program I wrote simply re-arranged and tweaked the register writes to correspond with the OPL2 registers.

Here's a link to a good document about this chip:
<http://www.ccms.net/~aomit/oplx/>

The tremolo depth is set to 4.3db and the vibrato depth is set to 14 cent (in regards to OPL2 settings; to achieve this you would write 0C0h to OPL register 0BDh). All operator connections are fixed in FM mode. (Where Modulator modulates the Carrier).

Index 00h (Modulator)

Index 01h (Carrier)

Bit7 Tremolo Enable
Bit6 Vibrato Enable
Bit5 Sustain Enable
Bit4 KSR
Bit3-0 Multiplier

Index 02h

Bit7-6 Key Scale Level
Bit5-0 Output Level

Index 03h

Bit7-5 Not used (Write 0's)
Bit4 Carrier Waveform
Bit3 Modulator Waveform
There are only two waveforms available. Sine and rectified sine (only the positive cycle of the sine; negative cycle "chopped off".)
Bit2-0 Feedback Control

Index 04h (Modulator)

Index 05h (Carrier)

Bit7-4 Attack

Bit3-0 Decay

Index 06h (Modulator)

Index 07h (Carrier)

Bit7-4 Sustain

Bit3-0 Release

Register Settings for the 15 fixed instruments

These instruments are not 100% correct! There is no way to extract the register settings from the chip short of an electron microscope.

I have "tuned" these instruments best I could, though I know a couple are not exactly right.

Table shows Register 0-7 settings for Instrument 1-0Fh

1	-	05	03	10	06	74	A1	13	F4
2	-	05	01	16	00	F9	A2	15	F5
3	-	01	41	11	00	A0	A0	83	95
4	-	01	41	17	00	60	F0	83	95
5	-	24	41	1F	00	50	B0	94	94
6	-	05	01	0B	04	65	A0	54	95
7	-	11	41	0E	04	70	C7	13	10
8	-	02	44	16	06	E0	E0	31	35
9	-	48	22	22	07	50	A1	A5	F4
A	-	05	A1	18	00	A2	A2	F5	F5
B	-	07	81	2B	05	A5	A5	03	03
C	-	01	41	08	08	A0	A0	83	95
D	-	21	61	12	00	93	92	74	75
E	-	21	62	21	00	84	85	34	15
F	-	21	62	0E	00	A1	A0	34	15

Mapper 86: Jaleco Early Mapper 2 - PRG/32K, VROM/8K

Used only by Moero Pro Baseball (Red/Black).

6000h Memory Control
Bit6,1,0 Select 8K VROM bank at PPU 0000h-1FFFh
Bit5,4 Select 32K ROM bank at 8000h-FFFFh
Bit7,3,2 Not used (always zero)
7000h Unknown

Also used by Lum no Wedding Bell though that does use only VROM banking.
Functional same as Mapper 87, though that does as well use only VROM banking.

Mapper 87: Jaleco/Konami 16K VROM - VROM/8K

Used only by Hyper Olympic, Goonies, Choplifter, Argus, Ninja Jajamaru Kun, City Connection.

6000h Select 8K VROM bank at PPU 0000h-1FFFh (Bit 1 used only)

Mapper 88: Namco 118

Used by Devil Man, Dragon Spirit, Namcot Mahjong 3, Quinty.

8000h Index/Control (3bit)

Bit2-0 Command Number
 0 - Select 2x1K VROM at PPU 0000h-07FFh (Banks 0-63)
 1 - Select 2x1K VROM at PPU 0800h-0FFFh (Banks 0-63)
 2 - Select 1K VROM at PPU 1000h-13FFh (Banks 64-127)
 3 - Select 1K VROM at PPU 1400h-17FFh (Banks 64-127)
 4 - Select 1K VROM at PPU 1800h-1BFFh (Banks 64-127)
 5 - Select 1K VROM at PPU 1C00h-1FFFh (Banks 64-127)
 6 - Select 8K ROM at 8000h-9FFFh
 7 - Select 8K ROM at A000h-BFFFh
 N/A - Fixed 16K ROM at C000h-FFFFh (always last bank)
 8001h Data Register (Indexed via Port 8000h)

The carts have 128K VROM, of which the lower 64K can be mapped only to Pattern Table 0, the upper 64K only to Pattern Table 1.

Devil Man additionally writes 00h to Port C000h, purpose unknown.

Devil Man ROM-image is declared as 4-screen-vertical-mirror, that is nonsense.

Mapper 89: Sunsoft Early - PRG/16K, VROM/8K

Used by only by Mito Koumon.

8000h-FFFFh Memory Control
 Bit7 Unknown - maybe Name Table related, maybe not.
 Bit6-4 Select 16K ROM bank at 8000h-BFFFh
 N/A Fixed 16K ROM bank at C000h-FFFFh (always last bank)
 Bit3-0 Select 8K VROM bank at PPU 0000h-1FFFh

The program seems to attempt to (unsuccessfully) resolve bus-conflicts.

Mapper 90: Pirate MMC5-style

Used by some pirate titles (Taiwan) such as Super Mario World, Tekken2 and Mortal Kombat. Features similar functions as MMC5, though the Port addresses are completely different.

5000h(W) Maths Coprocessor Parameter A
 5001h(W) Maths Coprocessor Parameter B
 5000h(R) Maths Coprocessor 8bit Result of A*B
 8000h PRG 8K at 8000h-9FFFh
 8001h PRG 8K at A000h-BFFFh or 16k PRG bank at \$A000-?
 8002h PRG 8K at C000h-DFFFh
 8003h PRG 8K at E000h-FFFFh
 9000h/A000h LSB/MSB of VROM bank at PPU 0000h (1K, 2K, 4K, 8K)
 9001h/A001h LSB/MSB of VROM bank at PPU 0400h (1K)
 9002h/A002h LSB/MSB of VROM bank at PPU 0800h (1K, 2K)
 9003h/A003h LSB/MSB of VROM bank at PPU 0C00h (1K)
 9004h/A004h LSB/MSB of VROM bank at PPU 1000h (1K, 2K, 4K)
 9005h/A005h LSB/MSB of VROM bank at PPU 1400h (1K)
 9006h/A006h LSB/MSB of VROM bank at PPU 1800h (1K, 2K)
 9007h/A007h LSB/MSB of VROM bank at PPU 1C00h (1K)
 B000h/B004h LSB/MSB of 1K VROM bank at PPU 2000h (Name Table VROM mode)
 B001h/B005h LSB/MSB of 1K VROM bank at PPU 2400h (Name Table VROM mode)
 B002h/B006h LSB/MSB of 1K VROM bank at PPU 2800h (Name Table VROM mode)
 B003h/B007h LSB/MSB of 1K VROM bank at PPU 2C00h (Name Table VROM mode)

\$C000 irq registers Unknown
 \$C001 irq registers Unknown
 \$C006 irq registers Unknown
 \$C007 irq registers Unknown
 \$C002 irq clear irq_flag=0 and INT signal is clear

```

$C003      irq reset          if $C005=0, irq_flag=0
                                     else, irq_flag=1 and irq_counter=irq_latch
$C004      irq reset          It seems same of $C003
$C005      irq counter        irq_flag=1, irq_latch = irq_counter = value

```

IRQs work like MMC3 does.

IRQ counter is decremented at every scanline, always while not blanking (scanline < 240), and background or sprites are enabled. When it reaches zero (or a negative value), IRQ is triggered `_IF_` the `irq_flag` is set, clearing the `irq_flag` and `irq_latch`.

```

D000h Bank Mode
      Bit1-0 PRG Bank Size
                0 Fixed last 32K at 8000h-FFFFh (initial setting)
                1 16K Banks, and Fixed last 16K at C000h-FFFFh
                2 8K Banks, via Bits 2,7, and Ports 8000h-8003h
                3 8K in reverse mode?
      Bit2 PRG Bank at E000h in 8K Mode (0=Last 8K, 1=Port 8003h)
      Bit4-3 VROM Bank Size (0=8K, 1=4K, 2=2K, 3=1K)
      Bit5 Name Table Source (0=VRAM via D001h, 1=VROM via B00Xh)
      Bit6 Not used
      Bit7 PRG Bank at 6000h (1=enabled) (Similiar/Instead E000h?)
D001h Name Table Control (in VRAM mode) (only lower 2bit used)
      0 Two-Screen, Vertical mirroring
      1 Two-Screen, Horizontal mirroring
      2,3 One-Screen, BLK0

$D002      unknown          Unused?
$D003      bank page        Only used by larger carts

```

if (bankmode.bit5), map CHR data in the nametable area using \$B00x values,
 But, if (high byte, low byte) != (0,0) or (0,1) or (0,2) or (0,3),
 so bankmode.bit5 is cleared, and mirroring does not change.

```

for(i=0;i<4;i++) {
  if(!nam_high_byte[i] && (nam_low_byte[i] == i)) {
    bankmode &= 0xdf; //clear bit5 --> use VRAM with mirroring
    return;
  }
}
next

```

If you ignore it, a lot of crappy gfx might be displayed.

Mapper 91: HK-SF3 - PRG/8K, VROM/2K, IRQ

This mapper is used on the pirate cart with a title screen reading "Street Fighter 3". It may or may not have been used in other bootleg games.

```

6000h Select 2K VROM bank at PPU 0000h-07FFh
6001h Select 2K VROM bank at PPU 0800h-0FFFh
6002h Select 2K VROM bank at PPU 1000h-17FFh
6003h Select 2K VROM bank at PPU 1800h-1FFFh
7000h Select 8K ROM bank at 8000h-9FFFh
7001h Select 8K ROM bank at A000h-BFFFh
N/A Fixed 16K ROM bank at C000h-FFFFh (always last 16K)
7006h IRQ Disable/Acknowledge (write any value)
7007h IRQ Enable (write any value)

```

When enabled, IRQs are requested every 8 scanlines, except during VBlank.
 Vertical mirroring is always active.

Mapper 92: Jaleco Early Mapper 1 - PRG-HI, VROM/8K

Used by Moero Pro Soccer, Moero Pro Baseball'88.

```
8000h-FFFFh Memory Control
Bit7-6 Function Select
0 Confirm Selection
1 Select 8K VROM bank at PPU 0000h-1FFFh
2 Select 16K ROM bank at C000h-FFFFh (upper half of PRG memory)
3 Reserved (would probably select both PRG+VROM)
Bit5-4 Not used
Bit0-3 ROM or VROM Bank Number for above Selection
```

Bus-conflicts. Example: To select PRG Bank 7, first write 87h, then 07h.

Same as Mapper 72, except that this one maps the UPPER half of PRG memory.

Mapper 93: 74161/32 - PRG/16K

Used only by Fantasy Zone.

```
8000h-FFFFh Memory Control
Bit0 Unknown, seems to be always set.
Bit1-3 Always zero
Bit4-6 Select 16K ROM bank at 8000h-BFFFh
Bit7 Always zero
```

Bus-conflicts. Uses VRAM.

Mapper 94: 74161/32 - PRG/16K

Used only by Senjou no Okami (Capcom's Commando, japanese version).

```
8000h-FFFFh Memory Control
Bit0-1 Always zero
Bit2-4 Select 16K ROM bank at 8000h-BFFFh
Bit5-7 Always zero
```

Bus-conflicts. Uses VRAM.

Mapper 95: Namcot MMC3-Style

Looks like MMC3, but doesn't seem to have IRQs and various other functions.

Used by Dragon Buster 1, and maybe many other "MMC3" games.

```
8000h Index/Control (3bit)
Bit2-0 Command Number
0 - Select 2x1K VROM at PPU 0000h-07FFh
1 - Select 2x1K VROM at PPU 0800h-0FFFh
2 - Select 1K VROM at PPU 1000h-13FFh
3 - Select 1K VROM at PPU 1400h-17FFh
4 - Select 1K VROM at PPU 1800h-1BFFh
5 - Select 1K VROM at PPU 1C00h-1FFFh
6 - Select 8K ROM at 8000h-FFFFh
7 - Select 8K ROM at 8000h-FFFFh
```

Used by Oeka Kids - Anpanman no Hiragana Daisuki, Anpanman to Oekaki Shiyou.

(These games seem to require a special controller.)

Bus-conflicts. Same as AOROM, but without BEK0/BLK1 select. Has VRAM.

```
Bit0-1N/A
Bit2-4 Select 8K ROM Bank at C000h-FFFFh (always last bank)
8B01B Data Register (may be used for performance ROMs)
```

Mapper 97: Irem - PRG HI

Used by Kaiketsu Yanchamaru.

```
8000h-FFFFh Memory Control
  Bit7-6 Unknown (used values are 1,2 - values 0,3 unused)
           (Maybe Name Table Mirroring)
  Bit5-4 Not used (always zero)
  Bit3-0 Select 16K ROM bank at C000h-FFFFh (upper block!)
  N/A     Fixed 16K ROM bank at 8000h-BFFFh (always LAST 16K bank)
```

Bus-conflicts.

Mapper 99: VS Unisystem Port 4016h - VROM/8K

Used by VS Super Mario Bros, VS Ice Climber, VS Excitebike and others.

```
Port 4016h/Write:
  Bit2 Select 8K VROM bank at PPU 0000h-1FFFh (VS Unisystem only)
  Bit0 Joypad Strobe
```

Most of these games use 4-screen mirroring. Above bank selection uses an expansion port output, which does not show up on NES/Famicom cartridge bus, and thus works only with VS Unisystem arcade machines. The machine also includes special DIP-Switch and Credit I/O ports as described in Controllers chapter.

Mapper 100: Whatever

No info. Don't have any such ROM-images.

Reportedly "MMC3/Nestice/Trainer/Buugy Mode Used in hacked roms !!!"

Sounds like homebrew hacks that work only on certain emulators, but not on real MMC3 hardware. Or it is just meant to be corrupted .NES files with garbage in reserved header entries at [07h..0Fh].

Mapper 105: X-in-1 MMC1

Used by Nintendo World Championships 1990. Works similar like normal MMC1:

[Mapper 1: MMC1 - PRG/32K/16K, VROM/8K/4K, NT](#)

However, the four registers are used like this:

```
Register 0 Configuration Register (same as MMC1)
Register 1 ROM Bank Base (Bit4 unknown)
Register 2 Not used
Register 3 ROM Bank (same as MMC1, but ORed with Base)
```

And, accessing Register 3 via Port FFF0h (instead normal FFFFh) appears to mask (zero) the new Register 3 bank number, until writing to Register 1.

Initially first 32K.

NB. The Championships 3-in-1 multicart doesn't have a game selection menu, it runs game 1 until reaching a certain score, and then switches to game 2, and so on. The controls are strange: it appears one can start the cartridge only when connecting a zapper to port 1, or a joystick to port 2, whilst gameplay works only with joystick at port 1.

Mapper 112: Asder - PRG/8K, VROM/2K/1K

Used by Huang Di, and San Guo Zhi - Qun Xiong Zheng Ba.

```
8000h Index (0-7)
      0 Select 8K ROM at 8000h-9FFFh
      1 Select 8K ROM at A000h-BFFFh
      2 Select 2x1K VROM at PPU 0000h-07FFh
      3 Select 2x1K VROM at PPU 0800h-0FFFh
      4 Select 1K VROM at PPU 1000h-13FFh
      5 Select 1K VROM at PPU 1400h-17FFh
      6 Select 1K VROM at PPU 1800h-1BFFh
      7 Select 1K VROM at PPU 1C00h-1FFFh
      N/A Fixed 16K ROM at C000h-FFFFh (always last 16K)
A000h Data (indexed via Port 8000h)
C000h Unknown, always 00h
E000h Unknown, always 00h
```

Mapper 113: Sachen/Hacker/Nina

[Seems to be same as Nina-3 and/or Nina-6]

[Mapper 79: AVE Nina-3 - VROM/8K](#)

[Mapper 81: AVE Nina-6](#)

Used by Metal Fighter, Side Winder, Rad Racket - Deluxe Tennis II, AV Hanafadu Club, AV Soccer, Papillion, Deathbots, Mahjong Companion, 4-in-1 Total Funpack.

```
4100h-41FFh Memory Control (commonly used addresses: 4100h, 4101h, 4120h)
      Bit0-2 Select 8K VROM bank at PPU 0000h-1FFFh
      Bit3-4 Select 32K ROM bank at 8000h-FFFFh (bigger carts only)
```

Many of these games seem to have been originally designed for mappers at 8000h-FFFFh, and do still write to these addresses. Also, "16 Mahjong" is declared as Mapper 113, though it uses 8000h-FFFFh only?

There's also a variant in which ROM selection is moved to Bit2:

[Mapper 133: Sachen](#)

Mapper 114: Super Games

Used by Lion King, Super Donkey Kong, and Pocohontos. Mask addresses by E001h.

```
6000h Unknown (usually zero, except Lion King before crashing?)
8000h Unknown (see notes below)
A000h Memory Control Index (see list below)
C000h Memory Control Data (indexed via A000h)
E000h IRQ Acknowledge (write any value)
6001h Unknown (always zero)
8001h Unknown (see notes below)
A001h IRQ Counter (MMC3-style, decremented per scanline, paused in VBlank)
C001h IRQ Counter
E001h IRQ Start
```

Memory Control Indexes are:

```
0 Select 2x1K VROM at PPU 0000h-07FFh
1 Select 1K VROM at PPU 1400h-17FFh
2 Select 2x1K VROM at PPU 0800h-0FFFh
3 Select 1K VROM at PPU 1C00h-1FFFh
4 Select 8K ROM at 8000h-9FFFh
5 Select 8K ROM at A000h-BFFFh
```

- 6 Select 1K VROM at PPU 1000h-13FFh
- 7 Select 1K VROM at PPU 1800h-1BFFh

Port 8001h, Bit0 seems to be used as IRQ disable in Lion King. All games seem to use Vertical Mirroring, but Pocohontos seems to toggle Name Tables via Port 8000h and 8001h during initialization.

Mapper 115: MMC3 Cart Saint

Used by Yuu Yuu Hakusho Final - Makai Saikyou Retsuden.

- 6000h Unknown (used values 00h, A0h, A4h)
- 6001h Unknown (always 00h)

No idea how it does <really> work, but the game appears to work when selecting 8K ROM bank number 8 at 8000h-9FFFh when [6000h]=A4h. Otherwise the mapper works like MMC3:

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

Note: MMC3 Register 7 <must> be initialized to 01h on reset.

Mapper 116: Whatever

No info. Don't have a ROM-image.

Reportedly "PC-Reserved" used by "AV beautiy fighting(not playable yet)".

Mapper 117: Future

Used by Sangokushi 4 (a clone of Warrior of Fate).

Appears related with Mapper 90.

- 8000h Select 8K ROM at 8000h-9FFFh
- 8001h Select 8K ROM at A000h-BFFFh
- 8002h Select 8K ROM at C000h-DFFFh
- N/A Fixed 8K ROM at E000h-FFFFh (last bank)
- 9000h Unknown (always FFh)
- 9001h Unknown (always 08h)
- 9003h Unknown (always 00h)
- A000h Select 1K VROM at PPU 0000h-03FFh
- A001h Select 1K VROM at PPU 0400h-07FFh
- A002h Select 1K VROM at PPU 0800h-0BFFh
- A003h Select 1K VROM at PPU 0C00h-0FFFh
- A004h Select 1K VROM at PPU 1000h-13FFh
- A005h Select 1K VROM at PPU 1400h-17FFh
- A006h Select 1K VROM at PPU 1800h-1BFFh
- A007h Select 1K VROM at PPU 1C00h-1FFFh
- A008h-A00Fh Unknown (always 01h, probably VROM bank related)
- C001h IRQ Counter/Start (MMC3, decremented per scanline, paused in VBlank)
- C002h IRQ Acknowledge (write any value)
- C003h IRQ Counter/Start (always write same value as to C001h)
- D000h Unknown (always 00h)
- E000h IRQ Enable (Bit0), upper 7bit unknown (always 0000011b)
- F000h Unknown (always 00h)

Mapper 118: MMC3 with different Name Tables

Used by Goal 2, Pro Sport Hockey, Armadillo, and Ys III.

Name Tables working somehow different, otherwise same as MMC3:
[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

Mapper 119: MMC3 TQROM with VROM+VRAM Pattern Tables

Used by Pinbot and High Speed. Consists of MMC3B plus 74HC32.
Allows to select between VROM and 8K VRAM as Pattern Tables (presumably by certain VROM bank numbers?), otherwise same as MMC3:
[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

Mapper 122: Whatever

No info. Don't have a ROM-image.
Reportedly "74161/32" used by "Madoola No Tsubasa".
Maybe "Madoola No Tsubasa" is same as "Wing of Madoola" (?)
[Mapper 184: Sunsoft - VROM/4K](#)

Mapper 133: Sachen

Used by Jovial Race.

```
4120h Memory Control
  Bit1-0 Select 8K VROM at PPU 0000h-1FFFh
  Bit2   Select 32K ROM at 8000h-FFFFh
```

Appears to be a variant of Mapper 113, with PRG ROM selection moved to Bit2.

Mapper 151: VS Unisystem - PRG/8K, VROM/4K

Used by VS Gradius and VS Goonies. On VS Unisystem arcade machines.

```
8000h Select 8K ROM bank at 8000h-9FFFh
A000h Select 8K ROM bank at A000h-BFFFh
C000h Select 8K ROM bank at C000h-DFFFh
N/A   Fixed 8K ROM bank at E000h-FFFFh
E000h Select 4K VROM bank at PPU 0000h-0FFFh
F000h Select 4K VROM bank at PPU 1000h-1FFFh
```

VS Goonies and VS Gradius use Vertical Mirroring (hard-wired).
Note: Other VS Games use Mappers 1,2,68,99 and maybe other mappers.
The machine also includes special DIP-Switch and Credit I/O ports as described in Controllers chapter.

Mapper 152: Whatever

The readme of Pocketnes (for Gameboy Advance) mentions existence Mapper 152.
No info. Don't have any ROM-images.

Mapper 160: Same as Mapper 90

Seems to be duplicate/nonsense, same as Mapper 90, used by Aladdin.

[Mapper 90: Pirate MMC5-style](#)

Mapper 161: Same as Mapper 1

Seems to be duplicate/nonsense, same as Mapper 1, used by Hanjuku Eiyuu.

[Mapper 1: MMC1 - PRG/32K/16K, VROM/8K/4K, NT](#)

Mapper 180: Nihon Bussan - PRG HI

Used by Crazy Climber.

```
8000h-FFFFh Memory Control
  Bit7-3  Not used (always zero)
  Bit2-0  Select 16K ROM bank at C000h-FFFFh (upper block!)
  N/A     Fixed 16K ROM bank at 8000h-BFFFh (always FIRST 16K bank)
```

Bus-conflicts.

Mapper 182: Same as Mapper 114

[Mapper 114: Super Games](#)

Mapper 184: Sunsoft - VROM/4K

Used by Atlantis no Nazo, Kanshakudama, and Wing of Madoola.

```
6000h Select VROM Banks
  Bit2-0 Select 4K VROM at PPU 0000h-0FFFh
  Bit6-4 Select 4K VROM at PPU 1000h-1FFFh
```

Mapper 185: VROM-disable

Appears to be used for CNROM games with only one (used) 8K VROM bank, however, VROM appears to be bigger than 8K, the games verify the the presence of the unused (empty) VROM banks on startup, and get caught in an endless loop if they do not exist.

```
8000h-FFFFh (De-)select VROM bank
```

Values used to switch VROM on/off are:

Off	On	Title
F0h	0Fh	Bird Week
00h	33h	B-Wings
00h	11h	Mighty Bomb Jack
20h	22h	Sansuu 1 Nen - Keisan Game
20h	22h	Sansuu 2 Nen - Keisan Game
00h	FFh	Sansuu 3 Nen - Keisan Game
13h	21h	Spy vs Spy

Above games are working when mapping an empty VROM bank (FFh-filled) either when (X)=13h, or

when (X AND 0Fh)=0.

Mapper 188: UNROM-reversed

Used by Karaoke Studio. Appears to be same as UNROM, but the first/second 128K are exchanged in the ROM-image (for unknown reason), ie. all bank numbers (including the fixed "last" bank) are XORed by 08h.

[Mapper 2: UNROM - PRG/16K](#)

Or, maybe only the first 8 banks are used, and the further banks are garbage?

Bus-conflicts.

Mapper 189: MMC3 Variant

Used by Master Fighter 2, and Street Fighter 2. There are three Master Fighter 2 versions, the 1st works as described below, the 2nd works but has distorted background, the 3rd doesn't work - ROM addresses appear corrupted (?). And, Street Fighter 2 works completely different - uses Ports 4132h for ROM, and 4122h/4123h for VROM (?) Anyways, the one working one works as such:

```
610xh Select 32K ROM Block (D7-D0 should match A7-A0, eg. [6103h]=03h)
```

The rest of the mapper is same as MMC3 (for the selected block of memory),

[Mapper 4: MMC3 - PRG/8K, VROM/2K/1K, VT, SRAM, IRQ](#)

Mapper 222: Dragon Ninja

Used by a pirate copy of Dragon Ninja.

```
8000h      Select 8K ROM at 8000h-9FFFh
A000h      Select 8K ROM at A000h-BFFFh
N/A        Fixed 16K ROM at C000h-FFFFh (last 16K)
9000h      Unknown (always E0h = Vertical Mirroring)
B000h/B001h Lower/upper 4bit of 1K VROM bank at PPU 0000h-03FFh
B002h/B003h Lower/upper 4bit of 1K VROM bank at PPU 0400h-07FFh
C000h/C001h Lower/upper 4bit of 1K VROM bank at PPU 0800h-0BFFh
C002h/C003h Lower/upper 4bit of 1K VROM bank at PPU 0C00h-0FFFh
D000h/D001h Lower/upper 4bit of 1K VROM bank at PPU 1000h-13FFh
D002h/D003h Lower/upper 4bit of 1K VROM bank at PPU 1400h-17FFh
E000h/E001h Lower/upper 4bit of 1K VROM bank at PPU 1800h-1BFFh
E002h/E003h Lower/upper 4bit of 1K VROM bank at PPU 1C00h-1FFFh
F000h      IRQ Counter/Stop/Set/Ack
F001h      IRQ Counter/Stop/Set/Ack
F002h      IRQ Counter/Start (incrementing approx every 120 (?) cycles)
```

Mapper 225: X-in-1

Used by 52-in-1, 58-in-1, 64-in-1, 72-in-1, 110-in-1, 115-in-1 carts. The reset vectors in all games are redirected to the game selection menu, and all copyright messages have been shamelessly removed.

```
8000h-FFFFh Memory Control (Write any data, port decoded by address lines)
  A14,A5-0   Select 8K VROM bank at PPU 0000h-1FFFh
  A14,A11-A6 Select PRG 2x16K ROM bank at 8000h-FFFFh
  A12        Select PRG page size (0=32K, 1=16K)
            0 32K page at 8000h-FFFFh (LSB/A6 of bank number ignored)
            1 16K page mirrored to 8000h-BFFFh and C000h-FFFFh
```

A13 ?Mirroring select (0=Vertical, 1=Horizontal Mirroring)
 A15 Must be "1"
 5800h-5FFFh 4x4bit Register File (D0-D3 data bits, addressed via A0-A1)

The 4x4bit latch is used as 16bit "RAM", used to restore the old menu selection when re-entering the menu by pushing the Reset button.

A14 is shared for both PRG/VROM in larger 2048K+1024K carts (those with more than 100 games), smaller 1024K+512K carts don't use A14.

Mapper 226: X-in-1

Used by Super 42-in-1 (1024K), and 76-in-1 (2048K).

Typically booted with opcode 8E 8E 8E 00 - MOV [8E8E],X; BRK in RAM.

Mapper 226:

8000h,8E8Eh Memory Control
 Bit4-0 Bank Number Bit4-0
 Bit5 Mode
 0 Map 32K ROM at 8000h-FFFFh (bank bits 6-1 used, bit0 ignored)
 1 Map the same 16K ROM bank at both 8000h-BFFFh and C000h-FFFFh
 Bit6 Name Table (0=Horizontal, 1=Vertical Mirroring)
 Bit7 Bank Number Bit5
 8001h Upper Bit of bank selection (2048K carts only)
 Bit0 Bank Number Bit6

The "VROM" banks of the original games are contained in PRG ROM, and are copied to 8K VRAM at PPU 0000h-1FFFh when starting a game.

See also:

[Mapper 233: X-in-1 plus Reset](#)

[Mapper 230: X-in-1 plus Contra](#)

FROM 226.TXT

Mapper hardware is provided by five 74-series ICs; LS74A, LS273, LS139, LS02 and LS153. A diode and capacitor are arranged to reset the mapping when the Reset button is pressed.

Register 1, Bit 1 - controls whether the CHR-RAM is write-protected:
 0 - not write-protected
 1 - write-protected

When the Reset button is pressed, both registers are reset to all zero bits.

Mapper 227: X-in-1

Used by 1200-in-1 (a fake containing only 14 different games).

8000h-FFFFh Memory Control (Write any data, port decoded by address lines)
 A6-A2 Select 16K ROM at 8000h-BFFFh (X)
 A1 Mirroring (0=Vertical, 1=Horizontal Mirroring)
 A14-A13 Menu mode (00b=Menu, 11b=Other)
 A9 128K Mode (1=128K, 0=Other)
 A0 32K Mode (1=32K, 0=Other)
 A11-A10,A8 Always 0
 A12 Usually 1 (except when initializing VRAM for game)
 A7 Usually 1 (except menu/contra/galaxian)

16K ROM at C000h-FFFFh is Bank 0 in Menu Mode (and on reset), Bank (X OR 1) in 32K Mode, Bank (X OR 7) in 128K Mode, or otherwise Bank (X) in 16K mode.

Mapper 228: X-in-1 Homebrewn

Used in two carts with incredible crude homebrewn games: Action 52 (multicart 52-in-1) and Cheetah Men 2 (single game cart).

```
8000h-FFFFh Memory Control (Decoded by address AND data lines)
  A3-A0,D1-D0      Select 8K VROM at PPU 0000h-1FFFh
  A12-A7           Select 32K ROM at 8000h-FFFFh
  A14-A13,A6-A4,D7-D2 Not used (always zero)
5FF0h-5FF3h 4x4bit Register File (D0-D3 data bits, addressed via A0-A1)
```

The 4x4bit latch is used as 16bit "RAM", used to restore the old menu selection when re-entering the main menu (used in Action52 multicart only).

Notes

Action52 has an odd ROM-size of 1.5MB, Banks 30h-3Fh are probably mirrors.

There seems to be no Name Table control bit (unless it is shared with bank-selection bits), most games look better at Vertical Mirroring (eg. cheetahmen, silversword), though some look better at Horizontal Mirroring (eg. criticalbp).

Mapper 229: 31-in-1

Used by 31-in-1 (512K+256K).

```
8000h-FFFFh Memory Control (Write any data, port decoded by address lines)
  A4-A0 Bank Selection, shared for PRG and VROM:
    Select 8K VROM bank at PPU 0000h-1FFFh
    Select 16K ROM bank at 8000h-BFFFh and same bank at C000h-FFFFh
  A selection of 01h works special, it maps 16K ROM banks 0 and 1,
  and bank 1 VROM, used for Super Mario which has 32K PRG ROM.
  A6-A5 Name Table
    0 Two-Screen, Vertical Mirroring
    1 Two-Screen, Horizontal Mirroring
    2 Probably one-screen, used on boot
    3 Probably one-screen, used in menu
  A14 The menu sets this bit when accessing bank 0
```

Mapper 230: X-in-1 plus Contra

Used in a 640K ROM-image. The 1st 128K contain a single game (Contra), the remaining 512K contain a 22-in-1 multicart. NB. the multicart menu also tries to detect further ROM, and if any such found, displays a 63-in-one menu.

No idea if/how selection between Contra and 22-in-1 works. The 22-in-1 part is identical with Mapper 226 (banks 0..31 located AFTER the 1st 128K):

[Mapper 226: X-in-1](#)

Mapper 231: 20-in-1

Used by 20-in-1.

```
8000h-FFFFh Memory Control (Write any data, port decoded by address lines)
  A7-A6 Name Table Setting (0-3)
    0 Probably one-screen (used by menu only)
    1 Two-Screen Vertical Mirroring
```

```

2 Two-Screen Horizontal Mirroring
3 Not used
A5 Always opposite of A1, ie. A5=(A0 XOR 1), probably 2nd chip-select
A4-A1 Select 32K ROM bank at 8000h-FFFFh
A0 Mode (0=Normal, 1=Mirror 1st half selected 32K bank to C000h-FFFFh)

```

The "VROM" banks of the original games are contained in PRG ROM, and are copied to 8K VRAM at PPU 0000h-1FFFh when starting a game.

There's also a version of the same cartridge with slightly different mapper:

[Mapper 61: 20-in-1](#)

Mapper 232: 4-in-1 Quattro Camerica

Used by Camerica 4-in-1 games with 256K ROM and 8K VRAM - Quattro Adventure, Quattro Arcade, and Quattro Sports.

```

9000h Select 64K block for 8000h-FFFFh (block number in Bit4-3)
C000h-Fxxxh Select 16K ROM bank at 8000h-BFFFh (within current 64K)
N/A High 16K ROM bank at C000h-FFFFh (last 16K of current 64K)
FFF0h,FFF1h Unknown - Write any value at proper timing (maybe lockout)

```

In some (not all) ROM-images, 2nd/3rd game seem to be mis-exchanged.

Mapper 233: X-in-1 plus Reset

Used by "42 Games" which is almost the same as "Super 42-in-1" (Mapper 226, which comes with separate 22 and 20 games menus, prompting the user to press Select to switch to the next menu).

42 Games (1024K ROM) comes with a "Level 1-4" main menu, each "Level" allows to select from 10 or 11 games. That main menu shows up only if the cartridge detects a special reset function, if that detection fails, then it assumes to be a 512K ROM, and enters a 22 or 20 games menu. That means, the cartridge can be split into two fully functional 512K ROMs when removing the reset function. That reset function appears to work like this:

```

FFFDh Reading from this address (the MSB of reset vector) destroys the
current Bank selection, probably setting it to a value of FFh, at
least anything different than 00h or 80h

```

Aside from the extra reset function, it is same as Mapper 226,

There's also ROM named "Unknown" numbered as Mapper 233 with other functionality?

Mapper 234: Maxi-15

Used by the AVE Maxi 15 Game Cartridge.

```

Registers are set by writing *or reading* certain locations. In the case
of writing, the programmer would need to ensure that the written value and
that put on the data bus by the program ROM do not conflict.

```

On power-up, and when the Reset button is pressed, registers R1 and R2 are cleared. R3 is not cleared when Reset is pressed. After R1 has been set to a non-zero value, it cannot be changed until the Reset button is pressed.

```

FF80h-FF9Fh Configuration Register (R1)
Bit7 Name Table Control (0=Vertical, 1=Horizontal Mirroring)
Bit6 Page Mode ROM/VROM Size (0=32K, 1=64K)

```

Bit5-0 Select 32K ROM/VROM bank (LSB ignored in 64K Page Mode)
 Bit5 is wired to /CS or /OE of the ROM chips, ie. both ROM and VROM are disabled when bit5 is set (unless additional ROMs would be connected to inverted Bit5, in larger carts).

FFE8h-FFF7h Memory Banking Register (R2)
 Bit7 Not Used
 Bit6-4 Select 8K VROM at PPU 0000h-1FFFh (Bit6 not used in 32K Page mode)
 Bit3-1 Not Used
 Bit0 Select 32K ROM at 8000h-FFFFh (Bit0 not used in 32K Page mode)

FFC0h-FFDFh Lockout Register (R3)
 Initially it is not possible to access R3. This is only possible after R1 has been set to a non-zero value.
 Bit7-2 Not Used
 Bit1 CIC RST
 Bit0 CIC OUT

Memory-mapping hardware consists of eleven chips: 74LS273, 2x74LS322, 2x74LS175, 2x74LS138, 74LS30, 74HC08, 74HC04 and a 4053. There are several discrete components, mostly related to the CIC-defeating function.

Mapper 240: C&E/Supertone - PRG/32K, VROM/8K

Used by Jing Ke Xin Zhuan (via Port 4800h), and Sheng Huo Lie Zhuan (via Port 4120h, or alternately GNROM-style via Port 8000h-FFFFh with bus-conflicts).

4120h, 4800h, 8000h-FFFFh Memory Control
 Bit7-6 Not used (always zero)
 Bit5-4 Select 32K ROM at 8000h-FFFFh (initially any 32K bank)
 Bit3-0 Select 8K VROM at PPU 0000h-1FFFh

Mapper 241: X-in-1 Education

Used by Education Games 18-in-1, and Study and Game 32-in-1.

8000h-FFFFh Select 32K ROM at 8000h-FFFFh (initially 1st 32K bank)
 5FF0h-5FFFh/Write Unknown (No info)
 5FF0h-5FFFh/Read Unknown (somewhat Bit6: 1=Ready/Okay)

Both cartridges require a special keyboard controller, similar as the Famicom keyboard, but with different keyboard matrix.

[Controllers - Keyboard](#)

Mapper 242: Waixing - PRG/32K, NT

Used by Wai Xing Zhan Shi.

8000h-FFFFh Memory Control (Write any data, port decoded by address lines)
 A6-A3 Select 32K ROM at 8000h-FFFFh (initially 1st 32K bank)
 A1 Mirroring (0=Vertical, 1=Horizontal Mirroring)
 A7,A0 Always 1
 A14-A8,A2 Always 0

Similar as Mapper 227 (without using the various memory modes though),

[Mapper 227: X-in-1](#)

Mapper 243: Sachen Poker - PRG/32K, VROM/8K

Used by Mei Nu Quan (Honey Peach), and Poker III 5-in-1.

4100h Index (0-7)
4101h Data for above index

The separate register indexes are:

0 Unknown, always 00h
1 Unknown, always 00h
2 Bit3 of 8K VROM at PPU 0000h-1FFFh
3 Unknown, always 00h
4 Bit0 of 8K VROM at PPU 0000h-1FFFh
5 Select 32K ROM at 8000h-FFFFh
6 Bit2,1 of 8K VROM at PPU 0000h-1FFFh
7 Unknown, always 05h

Formula for VROM Registers 2,4,6: Bank=(R2*8)+(R4 AND 1)+(R6 AND 3)*2

Mapper 244: C&E - PRG/32K, VROM/8K

Used by Decathlon only. Note: Cat Ninden Teyandee translations declared as "Mapper 244" seem to be MMC3 games.

8000h-FFFFh Memory Control
Bit7 Not used (zero)
Bit6-4 Swap bits (some sort of confusion / copy protection, see below)
Bit3 Set ROM or VROM bank (0=ROM, 1=VROM)
Bit2-0 Select 32K ROM at 8000h-FFFFh or 8K VROM at PPU 0000h-1FFFh

Swap bits for ROM Bank Number:

Bit4=1: XOR bank number by 03h
Bit5=1: Exchange bank number Bit0,1
Bit6=1: Not used

Swap bits for VROM Bank Number:

Bit4=1, Bit5=1, Bit6=1: XOR bank number by 07h (without further exchanges)
Bit4=0, Bit5=1, Bit6=1: Not used
Bit4=1: Exchange bank number Bit 0,1 (processed first)
Bit5=1: Exchange bank number Bit 1,2
Bit6=1; Exchange bank number Bit 2,0 (processed last)

Bus-conflicts.

Mapper 246: C&E - PRG/8K, VROM/2K, SRAM

Used by Fong Shen Bang - Zhu Lu Zhi Zhan.

6000h Select 8K ROM at 8000h-9FFFh
6001h Select 8K ROM at A000h-BFFFh
6002h Select 8K ROM at C000h-DFFFh
6003h Select 8K ROM at E000h-FFFFh (initially probably last bank, or bank 3)
6004h Select 2K VROM at PPU 0000h-07FFh
6005h Select 2K VROM at PPU 0800h-0FFFh
6006h Select 2K VROM at PPU 1000h-17FFh
6007h Select 2K VROM at PPU 1800h-1FFFh

The cartridge also contains SRAM at 6000h-7FFFh.
Not sure if SRAM at 6000h-6007h can be used.

Mapper 255: X-in-1 - (Same as Mapper 225)

Duplicated/nonsense mapper number, 255 is functional same as 225.

[Mapper 225: X-in-1](#)

Famicom Disk System (FDS)

Famicom Disk System (FDS) is a Famicom extension unit which was produced by Nintendo and only sold in Asian countries. It consists of a disk drive accepting 2.5" or 3" (?) floppies, 32K of RAM to load programs into, 8K of VRAM, and some other hardware described below.

[FDS Memory and I/O Maps](#)

[FDS I/O Ports - Timer](#)

[FDS I/O Ports - Disk](#)

[FDS I/O Ports - Sound](#)

[FDS BIOS Disk Format](#)

[FDS BIOS Disk Functions](#)

[FDS BIOS Disk Errors](#)

[FDS BIOS Data Areas in WRAM](#)

[FDS Disk Drive Operation](#)

FDS Memory and I/O Maps

FDS Memory Map

```
4020h-40FFh  I/O Ports (2C33) (Disk, Sound, Timer)
6000h-DFFFh  32K WRAM
E000h-FFFFh  8K FDS BIOS ROM
```

Caution: Parts of the FDS 32K WRAM, and of the built-in 2K WRAM are reserved for use by the FDS BIOS: 0000h-000Eh, 00F9h-0103h, and DFF6h-DFFFh.

FDS VRAM Map

```
0000h-1FFFh  Pattern Tables - 8K VRAM
```

Note: Horizontal/Vertical Name Table Mirroring can be selected via Port 4025h.

FDS I/O Map (2C33 Registers)

```
4020h  Timer IRQ Counter Reload value LSB (W)
4021h  Timer IRQ Counter Reload value MSB (W)
4022h  Timer IRQ Enable/Disable (W)
4023h  2C33 I/O Control Port
4024h  Disk Data Write Register (W)
4025h  Disk Control Register (W)
4026h  Disk External Connector Output (W)
4030h  Disk Status Register 0 (R)
4031h  Disk Data Read Register (R)
4032h  Disk Status Register 1 (R)
4033h  Disk External Connector Input (R)
4040h..407Fh  Sound Wave RAM - 64 x 6bit sample data (R/W)
4080h  Sound Volume Envelope (W)
4082h  Sound Wave RAM Sample Rate LSB (W)
4083h  Sound Wave RAM Sample Rate MSB and Control (W)
```

4084h Sound Sweep Envelope (W)
 4085h Sound Sweep Bias (W)
 4086h Sound Modulation Frequency LSB (W)
 4087h Sound Modulation Frequency MSB (W)
 4088h Sound Modulation Table (W)
 4089h Sound Wave RAM Control (W)
 408Ah Sound Envelope Base Frequency (W)
 4090h Sound Current Volume Gain Level (6bit) (R)
 4092h Sound Current Sweep Gain Level (6bit) (R)

FDS I/O Ports - Timer

Interrupt Timer intended to produce mid-screen scanline interrupts.

4020h - Timer IRQ Counter Reload value LSB (W)

4021h - Timer IRQ Counter Reload value MSB (W)

Reload value loaded to actual 16bit counter register on write to 4022h, and on counter underflow. Counter is decremented once per CPU clock cycle.

4022h - Timer IRQ Enable/Disable (W)

Bit1 Enable (0=Stop/Acknowledge Timer IRQ, 1=Start/Enable Timer IRQ)

Note: Timer IRQ Flag is found in Bit0 of Port 4030h, Disk Status Register 0).

The IRQ Vector is controlled by the BIOS via [0101h] and [DFFEh],

[FDS BIOS Data Areas in WRAM](#)

FDS I/O Ports - Disk

Disk access can be handled by using the FDS BIOS, so there's usually no need to write to below ports directly, the only exception would be the Screen Mirroring flag, to change it: Read the current value from [FAh], change Bit3, then write the new value to both [FAh] and [4025h].

4025h - Disk Control Register (W) (Read-able copy in WRAM at 00FAh)

Bit0 Drive Motor (0=On, 1=Off)
 When active (0), causes disk drive motor to stop. During this time, \$4025.1 has no effect. Uh, Active=0=Stop ?

Bit1 \ = Set drive head to the start of the first track.
 When active (0), causes disk drive motor to turn on. This bit must stay active throughout a disk transfer, otherwise \$4032.1 will always return 1. When deactivated, disk drive motor stays on until disk head reaches most inner track of disk.

Bit2 Disk Data Direction (0=Write, 1=Read)

Bit3 Screen Mirroring (0=Vertical, 1=Horizontal Mirroring)

Bit4 Enable CRC Phase (0=Read/Write Data, 1=Verify/Write CRC)

Bit5 Unknown (Should be always 1)

Bit6 GAP Control, Read Mode: 1=Reset CRC, and wait for end of GAP. Write Mode: 1=Reset CRC, and start writing data. 0=Write GAP (zeros)

Bit7 Disk IRQs on every byte transfer (0=Disable, 1=Enable)

4030h - Disk Status Register 0 (R)

Bit0 Timer IRQ Flag (0=None, 1=IRQ: Timer Underflow)

Bit1 Disk IRQ Flag (0=None, 1=IRQ: Request Data Transfer via 4024h/4031h)
 Reset when \$4024, \$4031, or \$4030 has been serviced.

Bit4 CRC Status (0=Okay, 1=Error, Checksum at end of block not matching)
Bit6 Lost Data (0=Okay, 1=Error, CPU didn't process 4024h/4031h in time)
Bit7 Unknown

4032h - Disk Status Register 1 (R)

Bit0 Disk Presence (0=Inserted, 1=Not inserted)
Bit1 Disk Rewind Flag (0=Ready/Playback, 1=Rewind Active)
Bit2 Write Protection (0=Writeable, 1=Read-only, or Disk not inserted)
Bit6 Usually 1 (probably relict of recent opcode byte)

4031h - Disk Data Read Register (R)

4024h - Disk Data Write Register (W)

8bit data received from / to be written to disk (least significant first).

Note: Disk IRQ Flag indicates when next byte is to be transferred.

4026h - External Connector Output (W) (Read-able copy in WRAM at 00F9h)

4033h - External Connector Input (R) (Inputs work only if Outputs=High)

Bit0-6 External Connector Pins 3-9 (0=Low, 1=High/Input)
Bit7 Power Good (0=Okay, 1=Battery power low)

Port 4026h should output High to any input pins, especially Bit7 should be always set to configure Power Good as input.

4023h - 2C33 I/O Control Port

Bit0 Disk I/O (0=Disable, 1=Enable)
Bit1 Sound (0=Disable, 1=Enable)

FDS I/O Ports - Sound

4040h..407Fh - Wave RAM - 64 x 6bit sample data (Read/Write)

Writes to these registers are ignored unless Write Mode is turned on (see register 4089h).

4089h - Wave RAM Control (Write Only)

Bit7 Wave Write Mode (1=Stop Sound output & Allow to write to Wave RAM)
Bit6-2 Not used
Bit1-0 Master Volume (0-3 = 100%,66%,50%,40% = 30/30,20/30,15/30,12/30)

4082h - Wave RAM Sample Rate LSB (Write Only)

Bit7-0 Lower 8 bits of the main unit's frequency (upper 4 bits in 4083h)

4083h - Wave RAM Sample Rate MSB and Control (Write Only)

Bit7 Main Unit disable (0=Enable, 1=Disable Sound Output)
Bit6 Envelope disable (0=Normal, 1=Disable Volume/Sweep Envelopes)
Bit5-4 Not used
Bit3-0 Upper 4 bits of the main unit's frequency

Main Unit / Sample Rate: (per entry of the 64-entry wave ram)

$$F = 1.79\text{MHz} * (\text{Freq} + \text{Mod}) / 65536$$

Mod = Frequency change based on the Modulation unit

If the 12bit frequency is zero, the Main unit is disabled (channel silent).

408Ah - Envelope Base Frequency (Write Only)

Bit7-0 Envelope Base Frequency, $F_{base}=1.79\text{MHz}/8/N$

F_{base} used by 4080h and 4084h. Volume/Sweep Envelope are disabled if $N=0$.

4080h - Volume Envelope (Write Only)

Bit7 Volume Envelope Mode (0=Volume Envelope, 1=Fixed Volume)
Bit6 Volume Envelope Direction (When enabled / at specified rate)
0=Decrease Volume by 1 (only if Volume>00h)
1=Increase Volume by 1 (only if Volume<20h)
Bit5-0 When Bit7=1: Volume Level (0-20h=Muted-Loudest, 21h-3Fh=Same as 20h)
Bit5-0 When Bit7=0: Volume Envelope Rate, $F=F_{base}/(N+1)$

The volume level can be set to 00h-3Fh by write with Bit7=1, this level is also used as initial volume when switching to envelope mode by setting Bit7=0.

In decrease mode, initial values 21h-3Fh are resulting delayed decrease; volume stays at maximum level until the value gets smaller than 20h.

4084h - Sweep Envelope (Write Only)

Bit7 Sweep Envelope Disable (1=Disable)
Bit6 Sweep Envelope Mode (0=Decrease, 1=Increase sweep gain)
Bit5-0 When Bit7=1: Sweep Gain
Bit5-0 When Bit7=0: Sweep Envelope Rate, $F=F_{base}/(N+1)$

4085h - Sweep Bias (Write Only)

Bit7 Not used
Bit6-0 Sweep Bias (signed 7bit; -40h..+3Fh)

Sweep Bias is used by the Modulation unit in calculating frequency bend.

Sweep Bias negative: Modulation unit will be bending frequency down.

Sweep Bias positive: Modulation unit will be bending frequency up.

Any write to Sweep Bias register resets Modulation Unit's address to zero. This address is used by the Modulation Unit when looking up entries written to the Modulation table (via \$4088).

4086h - Modulation Frequency LSB (Write Only)

Bit7-0 Lower 8bit of 12bit Modulation frequency

4087h - Modulation Frequency MSB (Write Only)

Bit7 Modulation Enable/Disable (0=Enable, 1=Disable)
Bit6-4 Not used
Bit3-0 Upper 4bit of 12bit Modulation frequency

Modulation Unit: Modulation Rate (per entry of the 64-entry modulation table)

$$F = 1.79\text{MHz} * \text{ModFreq} / 65536$$

If the 12bit frequency is zero, the Modulation unit is disabled.

4088h - Modulation Table (Write Only)

Bit7-3 Not used
Bit2-0 Modulation input

Writing to this register puts the value written at the END of the modulation table ****twice****, and shifts each entry already in the table 2 places to the front. The first 2 entries of the Modulation table are shifted out and lost.

old, old <-- ModTable_0 <-- ModTable_1 <-- ... <-- ModTable_63 <-- new, new

4090h - Current Volume Gain Level (6bit) (Read Only)

4092h - Current Sweep Gain Level (6bit) (Read Only)

4023h - 2C33 I/O Control Port

Bit0 Disk I/O (0=Disable, 1=Enable)
Bit1 Sound (0=Disable, 1=Enable)

FDS Sound by Disch, Release 1, 07/14/2004, based on info from Nori.

Sweep Envelope and Modulation Units

Sweep Envelope unit behaves just like the Volume Envelope, only it alters Sweep Gain instead of Volume Gain. The Envelope Unit never pushes Sweep Gain above \$20, but it still can get above \$20 if set that way via \$4084.

Increase/Decrease mode is determined by bit 6 of \$4084

Sweep Gain is used when calculating the Frequency change in the Modulation Unit...

The Modulation Unit, when clocked, takes 1 step through the Modulation Table (set by writes to \$4088).

The Sweep Bias is adjusted based on the 3-bit value in the table:

0: Bias=Bias+0 1: Bias=Bias+1 2: Bias=Bias+2 3: Bias=Bias+4
4: Bias=0 5: Bias=Bias-4 6: Bias=Bias-2 7: Bias=Bias-1

The address of the Modulation unit is incremented so that next clock it will use the next 3-bit value in the table. This address wraps at 64 and can be reset to zero by any write to \$4085.

Sweep Bias wraps to fit within a signed 7-bit value, if it goes greater than 63, it wraps around to -64, and if it goes below -64, it wraps to 63.

The Modulation Unit works by altering the Frequency of the Main Unit by a value calculated from the Sweep Gain and Sweep Bias values:

```
temp = Sweep_Bias * Sweep_Gain;
if temp AND 0Fh then
  if Sweep_Bias<0 then temp=temp-10h else temp=temp+20h
temp=temp/10h
if temp>193 then temp -= 258; // not a typo... for some reason the wraps
if temp<-64 then temp += 256; // are inconsistent
Mod = Freq * temp / 64;
```

In this code, Freq is the 12-bit MAIN UNIT frequency, and Mod is the amount that frequency is altered. This generated 'Mod' value is used in the frequency calculation of the main unit (given earlier):

```
Hz = NES * (Freq + Mod) / 65536
```

If at any time the Modulation unit is off, 'Mod' is zero. Otherwise 'Mod' is the above calculated value. If Freq + Mod produces a number less than or equal to zero, the channel is presumably silenced.

Unit Activity

There are many factors that could disable a unit, here's an overview section to cover all the needed requirements for the channel to be active.

Remember that each unit can be active regardless of the activity of other units. For example... even though the main unit is off and the channel is silent, this does not mean the Volume Envelope or Modulation units are inactive.

If any of the supplied conditions are false... the unit is inactive and will not be clocked. All conditions must be true for the unit to be active.

Volume Envelope Unit:

- Volume Envelope must be enabled (bit 7 of \$4080 must be off)
- Envelope Speed must be nonzero (set by \$408A)
- Envelope must be enabled (bit 6 of \$4083 must be off)

Sweep Envelope Unit:

- Sweep Envelope must be enabled (bit 7 of \$4084 must be off)

- Envelope Speed must be nonzero (set by \$408A)
- Envelope must be enabled (bit 6 of \$4083 must be off)

Modulation Unit:

- Modulation must be enabled (bit 7 of \$4087 must be off)
- Modulation frequency must be non-zero (set by \$4086/\$4087)

Main Unit (Wave RAM Sound Output):

- Main Unit must be enabled (bit 7 of \$4083 must be off)
- Main Unit Frequency must be non-zero (set by \$4082/\$4083)
- 'Freq + Mod' must be greater than zero (see Frequency Calculation section)
- Write Mode must be off (bit 7 of \$4089 must be off)

FDS BIOS Disk Format

Each disk has two sides, each side having a capacity of circa 64K, typically less than 64K data because some space is used for gaps and headers, also the exact capacity may vary depending on the transfer rate/rotation speed of the drive that has recorded the disk. To change an active side, the disk has to be removed, flipped, and inserted back into the drive.

The drive doesn't support random access, and the disk is NOT split into tracks and sectors. The data is stored sequentially on a single "track" which is wound in a spiral, starting at the outer edge, towards the center of the disk.

Side Header Block (56 bytes) (1st block on disk)

00h	Block Type	(01h)
01h-0Eh	Disk ID	(Must be ASCII string "*NINTENDO-HVC*")
0Fh	Maker ID	
10h-13h	Game Name	(usually 4 letter ASCII)
14h	Version Number	(usually 00h)
15h	Side Number	(00h=Side A, 01h=Side B) (00h=bootable)
16h	Disk Number	(00h=First, 01h=Second, etc.) (00h=bootable)
17h-18h	Extra Disk ID Field	
19h	Highest File ID for Boot files	(all files whose File ID is less or equal than this value are loaded automatically on power-up)
1Ah-37h	Reserved Space	(30 bytes, ignored by BIOS)

File Number Block (2 bytes) (2nd block on disk)

00h	Block Type	(02h)
01h	Number of Files on this side	

File Header Block (16 bytes) (for each file, 3rd,5th,7th... block on disk)

00h	Block Type	(03h)
01h	File Number	(00h=First file on this side, 01h=Second, etc.)
02h	File ID	(used to access files by Load Files function)
03h-0Ah	File Name	(not used, the BIOS access files by above File ID)
0Bh-0Ch	Target Address	(LSB, MSB)
0Dh-0Eh	File Size	(LSB, MSB)
0Fh	Target Area	(00h=WRAM, Other=VRAM)

File Data Block (1+LEN bytes) (for each file, 4th,6th,8th... block on disk)

00h	Block Type	(04h)
01h-LEN	Data	(LEN=File Size in File Header Block)

Gaps, Start Bits, CRC Values

Each block is preceded by a GAP (a stream of "0" bits), followed by a Start Bit ("1"), followed by the actual bytes contained in the block, followed by a 16bit CRC value.

FDS BIOS Disk Functions

Disk Boot

On power-up, the FDS call the Load Files function to load the boot files. The DiskID is FFh-filled (wildcards), except the Side Number and Disk Number entries which must be both 00h on boot-able disks. LoadList is empty, indicating to load all boot files, ie. all files whose File ID is less or equal than the Boot ID value defined in Disk Header block.

There must be at least two boot files on the disk, one containing the program with entrypoint (16bit pointer, which must be loaded to DFFCh), the other file containing the Nintendo License string (E0h bytes, which must be loaded to PPU 2800h, and which is verified against copy in BIOS at ED37h).

DiskID

Used by most BIOS functions to ensure that the correct disk/side is inserted. DiskID consists of 10 bytes which are compared against Disk Header Block [0Fh..18h]. Each DiskID byte may be set to FFh, which is used as wildcard, comparison always passes okay for that bytes. If the comparison does not match then error codes 04h..10h are returned, indicating which entry didn't match.

E1F8h - Load Files

The function scans <all> files on disk, respectively, the execution time is the same, no matter how many/how large files are loaded. On the contrary, multiple calls to LoadFiles would be unnecessarily slow.

```
RETaddr:      pointer to DiskID
RETaddr+2:    pointer to LoadList
A on return:  error code
Y on return:  count of files actually found
```

LoadList is a list of up to 20 File IDs, if the list contains less than 20 IDs then it must be terminated by FFh. If LoadList is empty (FFh in the first byte) then the boot files are loaded, that are all files whose File ID is less or equal than the Boot ID value which is defined in the Disk Header.

E32Ah - Get Disk Information

```
RETaddr:      pointer to DiskInfo
A on return:  error code
```

The DiskInfo pointer should point to a free memory location, which will receive the following data:

```
0Ah bytes   Disk Header Block [0Fh..18h], manufacturer, disk name, etc.
1 byte     File Number Block [01h], number of files on disk (N)
N*9 bytes  File Header Block [02h..0Ah], File ID and Filename, for each file
2 bytes    Disk Size (MSB,LSB)
```

Disk size is equal to the sum of each file's size entry, plus an extra 261 per file.

E237h - Append File

Sets A=FFh (append after last file), ie. same as A=FileCount, then continues at E239h (Write File).

E239h - Write File

Register A specifies how many old files are to be kept preserved on disk, these files are read/skipped, and the new file is then written to disk after those files, and the disks FileCount is set to A+1, making the new file to be the last file on disk, any further files are deleted/hidden.

In a second cycle, the written data is verified, if the verification fails (error 26h), then the file count is decremented, ie. the new file is deleted.

```
RETaddr:      pointer to DiskID
```

```
RETaddr+2:    pointer to FileInfo
A on call:    File Number (00h=First) (FFh=Append after last file)
A on return:  error code
```

FileInfo occupies 17 bytes, the first 14 bytes contain File Header entries [02h..0Fh], ie. File ID, Filename, Load Address, Filesize, and Load Area.

The last 3 bytes contain the Source Address and Source Area (which may or may not be same as Load Address and Load Area).

E2BBh - Adjust File count

```
RETaddr:      pointer to DiskID
A on call:     number to reduce current file count by
A on return:   error code
Special error: #$31 if A is less than the disk's file count
```

Reads in disk's file count, decrements it by A, then writes the new value back.

E2B7h - Check File count

```
RETaddr:      pointer to DiskID
A on call:     number to set file count to
A on return:   error code
Special error: #$31 if A is less than the disk's file count
```

Reads in disk's file count, compares it to A, then sets the disk's file count to A.

E305h - Set File count (alt. 1)

```
RETaddr:      pointer to DiskID
A on call:     number to set file count to
A on return:   error code
```

Sets the disk's file count to A.

E301h - Set File count (alt. 2)

```
RETaddr:      pointer to DiskID
A on call:     number to set file count to minus 1
A on return:   error code
```

Sets the disk's file count to A+1.

Don't expect disk calls to return quick; it may take several seconds to complete. The ROM BIOS always uses disk IRQ's to transfer data between the disk, so programs must surrender IRQ control to the ROM BIOS during these disk calls. The value at [\$0101] however, is preserved on entry, and restored on exit.

WRAM Target Addresses

Target Addresses should be in range 0200h-07FFh or 6000h-DFFFh. The BIOS rejects (silently ignores) most attempts to load data to 0000h-01FFh. In particular, it rejects Target Addresses at 0-1FFh (including mirrors at 800h,1000h,1800h), it also rejects wraps from FFFFh to 0000h.

However, it does not reject wraps to mirrors (eg. from 7FFh to 800h), clever use of this feature might allow to modify values on stack, and to bypass the Boot License.

Furthermore, target address 2000h can be used to enable NMIs during loading, the games Bislol, Bisyosya, and Bishojo Control are using this trick to abort the boot process and to start the game - without Boot License - via NMI vector at [DFFAh].

VRAM Source/Target Addresses

Mind that the screen should be disabled when loading/writing VRAM data, Port 2001h/Bit3-4 should be zero, otherwise VRAM could be accessed only in VBlank. Mind that physical content of VRAM addresses 2400h-2BFFh changes depending on current mirroring. The BIOS re-initializes parts of VRAM after loading the boot files on power up, overwriting any boot-files loaded to that areas.

Boot License String

32x7 characters in VRAM 2800h-28DFh (and copy in BIOS at ED37h)

```
"          NINTENDO r          "  
"          FAMILY COMPUTER TM   "  
"  
" THIS PRODUCT IS MANUFACTURED "  
" AND SOLD BY NINTENDO CO;LTD. "  
" OR BY OTHER COMPANY UNDER   "  
" LICENSE OF NINTENDO CO;LTD.. "
```

By using Non-ASCII BIOS Tile Numbers: A..Z=0Ah..23h SPC=24h .=26h ;=27h r=28h.
See WRAM Target Addresses above for methods to bypass the Boot License.

FDS BIOS Disk Errors

Error codes are returned in both A and X registers (plus zero flag, Z=okay)

```
00h Okay (no error) (zero flag set)  
01h No disk inserted (Port 4032h, Bit0)  
02h No battery/power (Port 4033h, Bit7)  
03h Disk write-protected (Port 4032h, Bit2)  
04h Bad Side Header [0Fh], Maker ID  
05h Bad Side Header [10h..13h], Game name  
06h Bad Side Header [14h], Game version  
07h Bad Side Header [15h], Side number (flip the disk)  
08h Bad Side Header [16h], Disk number  
09h Bad Side Header [17h], Extra ID Value 1  
10h Bad Side Header [18h], Extra ID Value 2  
20h Bad Nintendo License String (must be loaded to PPU 2800h-28DFh on boot)  
21h Bad Side Header [01h..0Eh], Disk ID (must be "*NINTENDO-HVC*")  
22h Bad Side Header [00h], Block ID must be 01h  
23h Bad File Number [00h], Block ID must be 02h  
24h Bad File Header [00h], Block ID must be 03h  
25h Bad File Data [00h], Block ID must be 04h  
26h Write-Verify Error (verification of written data failed)  
27h Block CRC Read Failure (Port 4030h, Bit 4)  
28h Lost Data (Port 4030h, Bit 6), CPU didn't read from 4031h in time  
29h Lost Data (Port 4030h, Bit 6), CPU didn't write to 4024h in time  
30h Disk Full (Port 4032h, Bit 1), Disk head has reached most inner track  
31h Data number of a disk doesn't match up (?)
```

FDS BIOS Data Areas in WRAM

The BIOS uses several places in memory, but only some of them are expected to be maintained by game code.

Addr Size Expl.

Scratch Area (destroyed by any calls to BIOS disk functions)

Copies of I/O Ports (used to READ content of Write-Only I/O Ports)

The BIOS does (and the game should) keep these bytes in sync with the ports.

```
0000h 1 file found counter  
0002h 2 second 16bit parameter  
0004h 1 previous stack frame  
0059h 1 value last written to [$4026] $FF on reset (disk ext connector)  
005Ah 1 value last written to [$4025] $2E on reset (disk control)  
005Bh 1 value last written to [$4016] 0'd on reset (joypad)  
005Ch 1 value last written to [$2005]#2 0'd on reset (ppu scrolling)  
005Dh 1 value last written to [$2005]#1 0'd on reset (ppu scrolling)  
005Eh 2 value destination address [$2001] $06 on reset (ppu control)  
005Fh 2 value transfer length count [$2001] $80 on reset (ppu control)  
000Eh 1 file found counter
```

IRQ/NMI/Reset Control

0100h	1	Action on NMI	(set to C0h on reset)
0101h	1	Action on IRQ	(set to 80h on reset)
0102h	2	Action on Reset	(AC35h after disk-boot, 5335h after warm-boot)

IRQ/NMI/Reset Vectors

DFF6h	2	Game NMI vector 1,	used if [0100h]=01xxxxxxb
DFF8h	2	Game NMI vector 2,	used if [0100h]=10xxxxxxb
DFFAh	2	Game NMI vector 3,	used if [0100h]=11xxxxxxb
DFFCh	2	Game Reset vector,	used if [0102h]=5335h or =AC35h
DFFEh	2	Game IRQ vector,	used if [0101h]=11xxxxxxb

If [0100h..0102h] don't match then the IRQ/NMI/Reset is handled internally by the BIOS, without using (and without changing) the Game vectors.

Address DFFCh contains the initial entrypoint at disk boot, and is also used as warm-boot vector when pushing the Reset button at a later time.

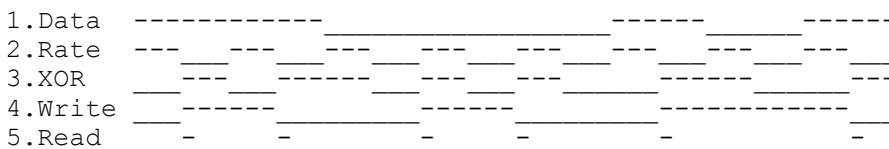
There may be more structured data areas in the zero page (for example, the BIOS joystick routines use \$F5..\$F8 for storing controller reads), but only the listed ones are used by the disk call subroutines.

FDS Disk Drive Operation

When the head reaches the end of the disk (most inner track), it returns to the beginning of the disk (most outer track) and the cycle repeats, upon request from the RAM adaptor. This means that on every scan, the entire disk is read (which takes about 6 seconds). The disk drive signals the RAM adaptor when the head has been positioned to the outer most track, and is starting a new scan.

FDS data transfer protocol

Like most disk drive units, the FDS disk drive is sending it's data out via serial connection.



The 1st row shows a 8bit data value, in this example A3h, or 10100011b, transferred LSB first. The 2nd row shows the transfer rate clock. The data/rate signals are XORed, as shown in the 3rd row. The actual signal written to disk is shown in the 4th row, magnetic polarity changes on any Low-to-High transitions of the XOR-signal. When reading from disk, spikes of one microsecond length are received on any polarity changes, as shown in the 4th row.

The RAM adaptor expects a transfer rate of 96.4kHz, although the tolerance it has for this rate is +/- 10%. This tolerance is necessary since, the disk drive can NOT turn the disk at a constant speed.

First GAP

The length of the first GAP period present on typical FDS disks (relative to the instant the disk drive's "-ready" signal is activated) is about 40000 bits, after which the first block start mark (indicating the beginning of the first file) will appear.

The disk drive unit signals the RAM adaptor when the head has moved to the beginning of the disk via the "-ready" signal it sends out (more on this later). The "-ready" signal is based on a mechanical switch inside the drive which is activated when the head is brought back to the outer most edge of the disk (the beginning). Because the switch will usually be triggered prematurely, the first 13000 bits (approx.) of data the drive will send out immediately after this switch is activated will be invalid. To compensate for this, the RAM adaptor purposely ignores the first 26100 bits (approx.) sent to it since it receives the "-ready" signal from the disk drive.

Further GAPS

The typical GAP period size used between files on FDS disks is roughly 976 bits (this includes the bits that are ignored by the RAM adaptor).

the RAM adaptor always ignores the first 488 bits (approx.) to follow after the immediate end of any file. This period allows the RAM adaptor (or the game rather) an opportunity to make the switch from reading from the disk to writing or vice-versa.

Final "GAP"

- The rest of the disk is filled with 0's after the last file is recorded (although it really shouldn't matter what exists on the disk after this).

CRC calculation

CRC appended to the immediate end of every file. The CRC is 16-bits, and is generated with a 17 bit poly. The poly used is 10001000000100001b (the X25 standard). Right shift operations are used to calculate the CRC (this effectively reverses the bit order of the polynomial, resulting in the 16-bit poly of 8408h). The file this algorithm is designed to work on has no block start mark in it (\$80), and has 2 extra bytes at the end (where a CRC calculation would normally reside) which are 0'd. While the block start mark is actually used in the calculation of a FDS file CRC, you'll see in the algo below that the block start mark (\$80) is moved directly into a register.

```
// ax is used as CRC accumulator
// si is the array element counter
// di is a temp reg
// Size is the size of the file + 2 (with the last 2 bytes as 0)
// Buf points to the file data (with the 2 appended bytes)
mov ax,8000h // this is the block start mark
sub si,si // zero out file byte index ptr
@@lop1:
mov dl,byte ptr Buf[si]
inc si
REPT 8
    shr dl,1; rcr ax,1; sbb di,di; and di,8408h; xor ax,di
ENDM
cmp si,Size
jc @@lop1
```

// ax now contains the CRC.

Special thanks to Val Blant for assistance in cracking the CRC algorithm used by the FDS.

- Some unlicensed FDS games activate the "-stop motor" signal (and possibly even "-write", even though the storage media is not intended to be written to) when a media transfer is to be discontinued, while "-scan media" is still active. While this is an unorthodox method of doing this, the best way to handle this situation is to give the "-stop motor" signal priority over any others, and force data transfer termination during it's activation.

- Check out the FDS loader project (which uploads *.FDS files to the RAM adaptor) for source code to my working disk drive emulator.

Hardware disk copy protection

Apparently, Nintendo had designed FDS disk drive units so that they cannot reprogram entire disks, while still somehow being able to write the contents of individual files to the end of disks. Now, there's a lot of undocumented things going on inside the disk drive unit, so I'm just going to say that there are two evil IC's you've got to watch out for inside the FDS disk drive- the 3213, and the 3206. There is a collection of 6 "FDS-COPY" jpegs over at NESdev which (pg. 4 right side, and pg. 5) give a pretty graphic overview of the steps involved in modding a stock FDS disk drive, so that it may reprogram disks. Although I haven't built the specific circuit described in the jpegs, I had designed & built a similar working circuit to defeat the FDS's evil copy protection circuitry, with excellent results.

Software disk copy protection

Special thanks to Chris Covell for bringing this to attention.

Apparently, some FDS disks implement a very simple copy protection scheme, which the game relies on in order for the game to refuse to work on the copied disk. Normally, the number of files that exist on an FDS disk is stored in the second block recorded on it. However, some games maintain "invisible" files, which are basically files that exist beyond what the file count number in the file count block indicates. This poses somewhat of a problem for copy software like FDSLOADR, since these tools rely on the file count block, and don't assume that there is any valid data past the last file found on the disk. This means that when these types of disks are copied, the invisible files will be lost, and when the game loads the files that do exist, the game's going to give the user heat about there being a file missing or something, gumming up the works. However in practice, when an FDS disk is programmed, the unused end of the disk is usually completely zeroed out, and this makes detecting the end of the disk simple: just wait to find a GAP period of extreme length. Except in rare cases, this model for detecting the true end of an FDS disk should generally provide the best results for copying the complete contents for all types of FDS disks. [That may be as well a trick to improve disk boot speed, not necessarily a copy protection.]

Physical disk lockout mechanism

Ever wonder why Nintendo engraved their company's name along the handle edge of all FDS disks? Inside the FDS disk drive bay, sitting just behind the lower part of the front black plastic faceplate, is a little plastic block with the letters "Nintendo" carved out of a hard plastic block. This basically forces disks that don't have holes in those locations from completely loading into the drive, circumventing usage. Now while many companies made FDS disks with those holes cut out, I'm sure there must be some disks out there that are compatible with the FDS, but don't have the holes. So, the solution is to simply disassemble the FDS disk drive, remove the disk cage, and remove the two screws securing the "Nintendo" letterblock.

Nintendo Playchoice 10

Nintendo Playchoice 10 Hardware Description v0.2 by Oliver Achten

Overview

The Nintendo Playchoice 10 hardware consists of 2 main parts: the control and the game part. The control part is driven by a Z80 CPU, which handles the game selection, menu display and controls the game part. The game part is 100% compatible to the NES game console, using a N2A03 CPU and a N2B03 PPU.

Z80 Memory Map

0000h-3FFFh	16KB BIOS ROM
8000h-87FFh	2KB Work RAM
8800h-8FFFh	2KB Battery backed RAM
9000h-97FFh	2KB Video RAM (write only)
C000h-DFFFh	8Kb Cartridge BIOS (resides on each game cartridge)
E000h-FFFFh	Protection

Z80 I/O Map - READ (by IN opcodes)

00h	Button/Status
bit 0:	Channel select button
bit 1:	Enter button
bit 2:	Reset button
bit 3:	N2A03 interrupt detect
bit 4:	<zero>
bit 5:	Coin 2 button
bit 6:	Service button
bit 7:	Coin 1 button
01h	DIP-switch 1, Bits 0-7
02h	DIP-switch 2, Bits 0-7
03h	Reading from this address clears Bit 3 of read port 00h

Z80 I/O Map - WRITE (by OUT opcodes) (all write ports are using bit 0 only)

00h	VRAM Access	(0=by Z80 CPU, 1=by Video circuit)
01h	Game Controls	(0=Disable, 1=Enable)
02h	PPU N2B03 Display output	(0=Disable, 1=Enable)
03h	APU N2A03 Sound output	(0=Disable, 1=Enable)
04h	Reset N2A03 CPU	(0=Reset, 1=Run)
05h	Stop N2A03 CPU	(0=Stop, 1=Run)
06h	Display Output Select	(0=Z80/Video circuit, 1=N2B03 PPU) (Only on single monitor version)
08h	Z80 NMI Control	(0=Disable, 1=Enable)
09h	Watchdog Control	(0=Enable, 1=Disable)
0Ah	N2B03 PPU Control	(0=Reset PPU, 1=Run PPU)
0Bh-0Eh	Bits 0-3 of Game Channel Select	(0-9)
0Fh	Upper KB of battery ram	(0=Disable, 1=Enable)

Video circuit

Video ram resides at \$9000 - \$97FF, and can only be accessed by the Z80 when bit 0 of write port \$00 is 0 (which also turns off the display). The screen matrix consists of 32*28 characters. Due to the design of the video circuit, the first visible line starts at \$9080. Character and color codes are stored in two bytes: byte 0 - cccccccc byte 1 - pppppccc (bit 0 - 7)

c: character code (\$000 - \$7FF)

p: color code (\$00 - \$1F)

The video circuit generates a nmi each vertical blank (when port \$40 is set to 1). Screen refresh is 60 Hz.

Watchdog

When enabled by port \$09, the watchdog resets the Z80 cpu after 8 screen refreshes, which shall prevent a machine lockup. The watchdog is reset by toggling port \$09 from 0 to 1, which should be done by the nmi routine.

Game part

As mentioned before, the game part consists practically of a whole NES. Memory map and registers are the same, however, the PPU is different since it has separate RGB and SYNC outputs, which provide a MUCH clearer picture than the N2A02 NES PPU. Unlike VS Unisystem PPU, the Playchoice 10 PPU holds the same color palette than its NES equivalent, so the game part of the Playchoice 10 can be considered 100% NES compatible!

Ports 0Bh-0Eh determine which one of the 10 game slots is activated. It affects also the C000h-FFFFh area in the Z80 memory map (cartridge bios and protection).

Protection

Protection is done by a RP5H01 unit. No technical description... Theoretically, it holds 16 bytes of data, which is accessed by both system and cartridge bios.

Theoretically, the unit would not need the protection when a new Z80 bios is written (Oliver Achten has written such a bios, so check out the web), which would allow the possibility of building a NES cart adaptor for the PC 10.

Unpredictable Things

Reading Garbage from unused PPU Bits

Semi-stable garbage is returned for 8bit write-only PPU registers (2000h, 2001h, 2003h, 2005h, 2006h), for lower 5bits of the PPU status register (Port 2002h), and for upper 2bit of palette values (Port 2007h at PPU address 3F00h-3FFFh). That garbage value is:

- 1) The 8bit-value most recently written to any PPU port (2000h-2007h).
- 2) The 8bit-value most recently read from 2004h or 2007h,

- 3) The 3bit-value most recently read from 2002h (lower 5bit unchanged).
- 4) Zero if none of the above has "updated" the garbage for longer period.

Reading from Palette Memory

Palette entries are 6bit values, when reading from palette memory, the upper two bits are garbage (see above). In monochrome mode (Port 2001h/Bit0=1) the returned lower 4bit are zero. Also, palette reads appear a bit unstable, and occasionally return incorrect values (at least on my PAL NES console).

Reading from empty Expansion / SRAM area at 4100h-7FFFh

Returns the most recently fetched data byte. That is: The third opcode byte (direct 16bit addressing), or the second zero-page byte (indirect addressing). In either case, the returned value is the MSB of 16bit BASE address, regardless of any index value which may wrap the MSB to the next page. For example: [4510h]=45h and [44FFh+11h]=44h are receiving different values, even though both are reading from the same memory address.

Reading from empty Expansion / APU area at 4000h-40FFh

Empty bits and bytes are: Write-only APU Ports 4000h-4014h, unused expansion addresses 4018h-40FFh, and unused bits in APU/Joypad Ports: 4015h/Bit5, 4016h/Bit7-5 (NES) or Bit7-3 (Famicom), and 4017h/Bit7-5. Normally returned garbage is 40h (base MSB, as described for 4100h-7FFFh), unless when indexing causes a page-wrap (from 3Fxxh+yyh to 40zzh), in that case several "unpredictable" things are happening:

The CPU adds the index to the address LSB, and reads from that address (3Fzzh) by mistake, in a second cycle the CPU adds the carry-out to the address MSB, and tries to read from the correct address (40zzh). The hardware doesn't output data for 40zzh, so that the CPU receives the most recently fetched data byte, which has been [3Fzzh], which is a mirror of PPU register [200zh]. To the worst, most PPU registers are either write-only, or cannot be read during rendering, see PPU Garbage above.

Cleverly used, this allows to detect the number of unused bits in 4016h, and to separate between NES or Famicom hardware.

CPU 65XX Microprocessor

Overview

[CPU Registers and Flags](#)

[CPU Memory Addressing](#)

Instruction Set

[CPU Memory and Register Transfers](#)

[CPU Arithmetic/Logical Operations](#)

[CPU Rotate and Shift Instructions](#)

[CPU Jump and Control Instructions](#)

[CPU Illegal Opcodes](#)

Other Info

[CPU Assembler Directives](#)

[CPU The 65XX Family](#)

[CPU Local Usage](#)

CPU Registers and Flags

The 65XX CPUs are equipped with not more than three 8bit general purpose registers (A, X, Y). However, the limited number of registers (and complete lack of 16bit registers other than PC) is partly covered by

comfortable memory operations, especially page 0 of memory (address 0000h-00FFh) may be used for relative fast and complicated operations, in so far one might say that the CPU has about 256 8bit 'registers' (or 128 16bit 'registers') in memory. For details see Memory Addressing chapter.

Registers

Bits	Name	Expl.
8	A	Accumulator
8	X	Index Register X
8	Y	Index Register Y
16	PC	Program Counter
8	S	Stack Pointer (see below)
8	P	Processor Status Register (see below)

Stack Pointer

The stack pointer is addressing 256 bytes in page 1 of memory, ie. values 00h-FFh will address memory at 0100h-01FFh. As for most other CPUs, the stack pointer is decrementing when storing data. However, in the 65XX world, it points to the first FREE byte on stack, so, when initializing stack to top set S=(1)FFh (rather than S=(2)00h).

Processor Status Register (Flags)

Bit	Name	Expl.
0	C	Carry (0=No Carry, 1=Carry)
1	Z	Zero (0=Nonzero, 1=Zero)
2	I	IRQ Disable (0=IRQ Enable, 1=IRQ Disable)
3	D	Decimal Mode (0=Normal, 1=BCD Mode for ADC/SBC opcodes)
4	B	Break Flag (0=IRQ/NMI, 1=BRK/PHP opcode)
5	-	Not used (Always 1)
6	V	Overflow (0=No Overflow, 1=Overflow)
7	N	Negative/Sign (0=Positive, 1=Negative)

Carry Flag (C)

Caution: When used for subtractions (SBC and CMP), the carry flag is having opposite meaning as for normal 80x86 and Z80 CPUs, ie. it is SET when above-or-equal. For all other instructions (ADC, ASL, LSR, ROL, ROR) it works as normal, whereas ROL/ROR are rotating <through> carry (ie. much like 80x86 RCL/RCR and not like ROL/ROR).

Zero Flag (Z), Negative/Sign Flag (N), Overflow Flag (V)

Works just as everywhere, Z it is set when result (or destination register, in case of some 'move' instructions) is zero, N is set when signed (ie. same as Bit 7 of result/destination). V is set when an addition/subtraction exceeded the maximum range for signed numbers (-128..+127).

IRQ Disable Flag (I)

Disables IRQs when set. NMIs (non maskable interrupts) and BRK instructions cannot be disabled.

Decimal Mode Flag (D)

Packed BCD mode (range 00h..99h) for ADC and SBC opcodes.

Break Flag (B)

The Break flag is intended to separate between IRQ and BRK which are both using the same vector, [FFFEh]. The flag cannot be accessed directly, but there are 4 situations which are writing the P register to stack, which are then allowing the examine the B-bit in the pushed value: The BRK and PHP opcodes always write "1" into the bit, IRQ/NMI execution always write "0".

XXX/???

On-Chip Bi-directional I/O port

Addresses (00)00h and (00)01h are occupied by an I/O port which is built-in into 6510 and 8500 CPUs (eg. used in C64), be sure not to use the addresses as normal memory. For description read chapter about I/O ports.

Caution

Because of the identical format, assemblers will be more or less unable to separate between [XXh+r] and [00XXh+r], the assembler will most likely produce [XXh+r] when address is already known to be located in page 0, and [00XXh+r] in case of forward references.

Beside for different opcode size/time, [XXh+r] will always access page 0 memory (even when XXh+r>FFh), while [00XXh+r] may direct to memory in page 0 or 1, to avoid unpredictable results be sure not to use (00)XXh+r>FFh if possible.

CPU Memory and Register Transfers

Register to Register Transfer

A8	nz----	2	TAY	Transfer Accumulator to Y	Y=A
AA	nz----	2	TAX	Transfer Accumulator to X	X=A
BA	nz----	2	TSX	Transfer Stack pointer to X	X=S
98	nz----	2	TYA	Transfer Y to Accumulator	A=Y
8A	nz----	2	TXA	Transfer X to Accumulator	A=X
9A	-----	2	TXS	Transfer X to Stack pointer	S=X

Load Register from Memory

A9	nn	nz----	2	LDA #nn	Load A with Immediate	A=nn
A5	nn	nz----	3	LDA nn	Load A with Zero Page	A=[nn]
B5	nn	nz----	4	LDA nn,X	Load A with Zero Page,X	A=[nn+X]
AD	nn nn	nz----	4	LDA nnnn	Load A with Absolute	A=[nnnn]
BD	nn nn	nz----	4*	LDA nnnn,X	Load A with Absolute,X	A=[nnnn+X]
B9	nn nn	nz----	4*	LDA nnnn,Y	Load A with Absolute,Y	A=[nnnn+Y]
A1	nn	nz----	6	LDA (nn,X)	Load A with (Indirect,X)	A=[WORD[nn+X]]
B1	nn	nz----	5*	LDA (nn),Y	Load A with (Indirect),Y	A=[WORD[nn]+Y]
A2	nn	nz----	2	LDX #nn	Load X with Immediate	X=nn
A6	nn	nz----	3	LDX nn	Load X with Zero Page	X=[nn]
B6	nn	nz----	4	LDX nn,Y	Load X with Zero Page,Y	X=[nn+Y]
AE	nn nn	nz----	4	LDX nnnn	Load X with Absolute	X=[nnnn]
BE	nn nn	nz----	4*	LDX nnnn,Y	Load X with Absolute,Y	X=[nnnn+Y]
A0	nn	nz----	2	LDY #nn	Load Y with Immediate	Y=nn
A4	nn	nz----	3	LDY nn	Load Y with Zero Page	Y=[nn]
B4	nn	nz----	4	LDY nn,X	Load Y with Zero Page,X	Y=[nn+X]
AC	nn nn	nz----	4	LDY nnnn	Load Y with Absolute	Y=[nnnn]
BC	nn nn	nz----	4*	LDY nnnn,X	Load Y with Absolute,X	Y=[nnnn+X]

* Add one cycle if indexing crosses a page boundary.

Store Register in Memory

85	nn	-----	3	STA nn	Store A in Zero Page	[nn]=A
95	nn	-----	4	STA nn,X	Store A in Zero Page,X	[nn+X]=A
8D	nn nn	-----	4	STA nnnn	Store A in Absolute	[nnnn]=A
9D	nn nn	-----	5	STA nnnn,X	Store A in Absolute,X	[nnnn+X]=A
99	nn nn	-----	5	STA nnnn,Y	Store A in Absolute,Y	[nnnn+Y]=A
81	nn	-----	6	STA (nn,X)	Store A in (Indirect,X)	[[nn+x]=A
91	nn	-----	6	STA (nn),Y	Store A in (Indirect),Y	[[nn]+y]=A
86	nn	-----	3	STX nn	Store X in Zero Page	[nn]=X
96	nn	-----	4	STX nn,Y	Store X in Zero Page,Y	[nn+Y]=X
8E	nn nn	-----	4	STX nnnn	Store X in Absolute	[nnnn]=X
84	nn	-----	3	STY nn	Store Y in Zero Page	[nn]=Y

94	nn	-----	4	STY nn,X	Store Y in Zero Page,X	[nn+X]=Y
8C	nn nn	-----	4	STY nnnn	Store Y in Absolute	[nnnn]=Y

Push/Pull

48		-----	3	PHA	Push accumulator on stack	[S]=A
08		-----	3	PHP	Push processor status on stack	[S]=P
68	nz----		4	PLA	Pull accumulator from stack	A=[S]
28	nzcidv		4	PLP	Pull processor status from stack	P=[S]

Notes: PLA sets Z and N according to content of A. The B-flag and unused flags cannot be changed by PLP, these flags are always written as "1" by PHP.

CPU Arithmetic/Logical Operations

Add memory to accumulator with carry

69	nn	nzc--v	2	ADC #nn	Add Immediate	A=A+C+nn
65	nn	nzc--v	3	ADC nn	Add Zero Page	A=A+C+[nn]
75	nn	nzc--v	4	ADC nn,X	Add Zero Page,X	A=A+C+[nn+X]
6D	nn nn	nzc--v	4	ADC nnnn	Add Absolute	A=A+C+[nnnn]
7D	nn nn	nzc--v	4*	ADC nnnn,X	Add Absolute,X	A=A+C+[nnnn+X]
79	nn nn	nzc--v	4*	ADC nnnn,Y	Add Absolute,Y	A=A+C+[nnnn+Y]
61	nn	nzc--v	6	ADC (nn,X)	Add (Indirect,X)	A=A+C+[[nn+X]]
71	nn	nzc--v	5*	ADC (nn),Y	Add (Indirect),Y	A=A+C+[[nn]+Y]

* Add one cycle if indexing crosses a page boundary.

Subtract memory from accumulator with borrow

E9	nn	nzc--v	2	SBC #nn	Subtract Immediate	A=A+C-1-nn
E5	nn	nzc--v	3	SBC nn	Subtract Zero Page	A=A+C-1-[nn]
F5	nn	nzc--v	4	SBC nn,X	Subtract Zero Page,X	A=A+C-1-[nn+X]
ED	nn nn	nzc--v	4	SBC nnnn	Subtract Absolute	A=A+C-1-[nnnn]
FD	nn nn	nzc--v	4*	SBC nnnn,X	Subtract Absolute,X	A=A+C-1-[nnnn+X]
F9	nn nn	nzc--v	4*	SBC nnnn,Y	Subtract Absolute,Y	A=A+C-1-[nnnn+Y]
E1	nn	nzc--v	6	SBC (nn,X)	Subtract (Indirect,X)	A=A+C-1-[[nn+X]]
F1	nn	nzc--v	5*	SBC (nn),Y	Subtract (Indirect),Y	A=A+C-1-[[nn]+Y]

* Add one cycle if indexing crosses a page boundary.

Note: Compared with normal 80x86 and Z80 CPUs, incoming and resulting Carry Flag are reversed.

Logical AND memory with accumulator

29	nn	nz----	2	AND #nn	AND Immediate	A=A AND nn
25	nn	nz----	3	AND nn	AND Zero Page	A=A AND [nn]
35	nn	nz----	4	AND nn,X	AND Zero Page,X	A=A AND [nn+X]
2D	nn nn	nz----	4	AND nnnn	AND Absolute	A=A AND [nnnn]
3D	nn nn	nz----	4*	AND nnnn,X	AND Absolute,X	A=A AND [nnnn+X]
39	nn nn	nz----	4*	AND nnnn,Y	AND Absolute,Y	A=A AND [nnnn+Y]
21	nn	nz----	6	AND (nn,X)	AND (Indirect,X)	A=A AND [[nn+X]]
31	nn	nz----	5*	AND (nn),Y	AND (Indirect),Y	A=A AND [[nn]+Y]

* Add one cycle if indexing crosses a page boundary.

Exclusive-OR memory with accumulator

49	nn	nz----	2	EOR #nn	XOR Immediate	A=A XOR nn
45	nn	nz----	3	EOR nn	XOR Zero Page	A=A XOR [nn]
55	nn	nz----	4	EOR nn,X	XOR Zero Page,X	A=A XOR [nn+X]
4D	nn nn	nz----	4	EOR nnnn	XOR Absolute	A=A XOR [nnnn]
5D	nn nn	nz----	4*	EOR nnnn,X	XOR Absolute,X	A=A XOR [nnnn+X]
59	nn nn	nz----	4*	EOR nnnn,Y	XOR Absolute,Y	A=A XOR [nnnn+Y]
45	nn	nz----	6	EOR (nn,X)	XOR (Indirect,X)	A=A XOR [[nn+X]]
5D	nn nn	nz----	5*	EOR (nn),Y	XOR (Indirect),Y	A=A XOR [[nn]+Y]

1D	nn nn	nz----	4*	ORA nnnn,X	OR Absolute,X	A=A OR [nnnn+X]
19	nn nn	nz----	4*	ORA nnnn,Y	OR Absolute,Y	A=A OR [nnnn+Y]
01	nn	nz----	6	ORA (nn,X)	OR (Indirect,X)	A=A OR [[nn+X]]
11	nn	nz----	5*	ORA (nn),Y	OR (Indirect),Y	A=A OR [[nn]+Y]

* Add one cycle if indexing crosses a page boundary.

Compare

C9	nn	nzc---	2	CMP #nn	Compare A with Immediate	A-[nn]
C5	nn	nzc---	3	CMP nn	Compare A with Zero Page	A-[nn]
D5	nn	nzc---	4	CMP nn,X	Compare A with Zero Page,X	A-[nn+X]
CD	nn nn	nzc---	4	CMP nnnn	Compare A with Absolute	A-[nnnn]
DD	nn nn	nzc---	4*	CMP nnnn,X	Compare A with Absolute,X	A-[nnnn+X]
D9	nn nn	nzc---	4*	CMP nnnn,Y	Compare A with Absolute,Y	A-[nnnn+Y]
C1	nn	nzc---	6	CMP (nn,X)	Compare A with (Indirect,X)	A-[[nn+X]]
D1	nn	nzc---	5*	CMP (nn),Y	Compare A with (Indirect),Y	A-[[nn]+Y]
E0	nn	nzc---	2	CPX #nn	Compare X with Immediate	X- nn
E4	nn	nzc---	3	CPX nn	Compare X with Zero Page	X-[nn]
EC	nn nn	nzc---	4	CPX nnnn	Compare X with Absolute	X-[nnnn]
C0	nn	nzc---	2	CPY #nn	Compare Y with Immediate	Y- nn
C4	nn	nzc---	3	CPY nn	Compare Y with Zero Page	Y-[nn]
CC	nn nn	nzc---	4	CPY nnnn	Compare Y with Absolute	Y-[nnnn]

* Add one cycle if indexing crosses a page boundary.

Note: Compared with normal 80x86 and Z80 CPUs, resulting Carry Flag is reversed.

Bit Test

24	nn	xz---x	3	BIT nn	Bit Test	A AND [nn], N=[nn].7, V=[nn].6
2C	nn nn	xz---x	4	BIT nnnn	Bit Test	A AND [...], N=[...].7, V=[...].6

Increment by one

E6	nn	nz----	5	INC nn	Increment Zero Page	[nn]=[nn]+1
F6	nn	nz----	6	INC nn,X	Increment Zero Page,X	[nn+X]=[nn+X]+1
EE	nn nn	nz----	6	INC nnnn	Increment Absolute	[nnnn]=[nnnn]+1
FE	nn nn	nz----	7	INC nnnn,X	Increment Absolute,X	[nnnn+X]=[nnnn+X]+1
E8		nz----	2	INX	Increment X	X=X+1
C8		nz----	2	INY	Increment Y	Y=Y+1

Decrement by one

C6	nn	nz----	5	DEC nn	Decrement Zero Page	[nn]=[nn]-1
D6	nn	nz----	6	DEC nn,X	Decrement Zero Page,X	[nn+X]=[nn+X]-1
CE	nn nn	nz----	6	DEC nnnn	Decrement Absolute	[nnnn]=[nnnn]-1
DE	nn nn	nz----	7	DEC nnnn,X	Decrement Absolute,X	[nnnn+X]=[nnnn+X]-1
CA		nz----	2	DEX	Decrement X	X=X-1
88		nz----	2	DEY	Decrement Y	Y=Y-1

CPU Rotate and Shift Instructions

Shift Left

0A		nzc---	2	ASL A	Shift Left Accumulator	SHL A
06	nn	nzc---	5	ASL nn	Shift Left Zero Page	SHL [nn]
16	nn	nzc---	6	ASL nn,X	Shift Left Zero Page,X	SHL [nn+X]
0E	nn nn	nzc---	6	ASL nnnn	Shift Left Absolute	SHL [nnnn]
1E	nn nn	nzc---	7	ASL nnnn,X	Shift Left Absolute,X	SHL [nnnn+X]

Shift Right

4A		0zc---	2	LSR A	Shift Right Accumulator	SHR A
----	--	--------	---	-------	-------------------------	-------

46	nn	0zc---	5	LSR nn	Shift Right Zero Page	SHR [nn]
56	nn	0zc---	6	LSR nn,X	Shift Right Zero Page,X	SHR [nn+X]
4E	nn nn	0zc---	6	LSR nnnn	Shift Right Absolute	SHR [nnnn]
5E	nn nn	0zc---	7	LSR nnnn,X	Shift Right Absolute,X	SHR [nnnn+X]

Rotate Left through Carry

2A		nzc---	2	ROL A	Rotate Left Accumulator	RCL A
26	nn	nzc---	5	ROL nn	Rotate Left Zero Page	RCL [nn]
36	nn	nzc---	6	ROL nn,X	Rotate Left Zero Page,X	RCL [nn+X]
2E	nn nn	nzc---	6	ROL nnnn	Rotate Left Absolute	RCL [nnnn]
3E	nn nn	nzc---	7	ROL nnnn,X	Rotate Left Absolute,X	RCL [nnnn+X]

Rotate Right through Carry

6A		nzc---	2	ROR A	Rotate Right Accumulator	RCR A
66	nn	nzc---	5	ROR nn	Rotate Right Zero Page	RCR [nn]
76	nn	nzc---	6	ROR nn,X	Rotate Right Zero Page,X	RCR [nn+X]
6E	nn nn	nzc---	6	ROR nnnn	Rotate Right Absolute	RCR [nnnn]
7E	nn nn	nzc---	7	ROR nnnn,X	Rotate Right Absolute,X	RCR [nnnn+X]

Notes:

ROR instruction is available on MCS650X microprocessors after June, 1976.

ROL and ROR rotate an 8bit value through carry (rotates 9bits in total).

CPU Jump and Control Instructions

Normal Jumps

4C	nn nn	-----	3	JMP nnnn	Jump Absolute	PC=nnnn
6C	nn nn	-----	5	JMP (nnnn)	Jump Indirect	PC=WORD[nnnn]
20	nn nn	-----	6	JSR nnnn	Jump and Save Return Addr.	[S]=PC+2, PC=nnnn
40		nzcidv	6	RTI	Return from BRK/IRQ/NMI	P=[S], PC=[S]
60		-----	6	RTS	Return from Subroutine	PC=[S]+1

Note: RTI cannot modify the B-Flag or the unused flag.

Glitch: For JMP [nnnn] the operand word cannot cross page boundaries, ie. JMP [03FFh] would fetch the MSB from [0300h] instead of [0400h]. Very simple workaround would be to place a ALIGN 2 before the data word.

Conditional Branches

10	dd	-----	2**	BPL disp	Branch on result plus	if N=0 PC=PC+/-nn
30	dd	-----	2**	BMI disp	Branch on result minus	if N=1 PC=PC+/-nn
50	dd	-----	2**	BVC disp	Branch on overflow clear	if V=0 PC=PC+/-nn
70	dd	-----	2**	BVS disp	Branch on overflow set	if V=1 PC=PC+/-nn
90	dd	-----	2**	BCC disp	Branch on carry clear	if C=0 PC=PC+/-nn
B0	dd	-----	2**	BCS disp	Branch on carry set	if C=1 PC=PC+/-nn
D0	dd	-----	2**	BNE disp	Branch on result not zero	if Z=0 PC=PC+/-nn
F0	dd	-----	2**	BEQ disp	Branch on result zero	if Z=1 PC=PC+/-nn

** The execution time is 2 cycles if the condition is false (no branch executed). Otherwise, 3 cycles if the destination is in the same memory page, or 4 cycles if it crosses a page boundary (see below for exact info).

Note: After subtractions (SBC or CMP) carry=set indicates above-or-equal, unlike as for 80x86 and Z80 CPUs. Obviously, this still applies even when using 80XX-style syntax.

Interrupts, Exceptions, Breakpoints

00		---1--	7	BRK	Force Break	B=1 [S]=PC+1, [S]=P, I=1, PC=[FFFE]
--		---1--	??	/IRQ	Interrupt	B=0 [S]=PC, [S]=P, I=1, PC=[FFFE]

```

--          ---1--  ??  /NMI  NMI          B=0 [S]=PC, [S]=P, I=1, PC=[FFFA]
--          ---1--  T+6 /RESET Reset      PC=[FFFC], I=1

```

Notes: IRQs can be disabled by setting the I-flag, a BRK command, a NMI, and a /RESET signal cannot be masked by setting I.

BRK/IRQ/NMI first change the B-flag, then write P to stack, and then set the I-flag, the D-flag is NOT changed and should be cleared by software.

The same vector is shared for BRK and IRQ, software can separate between BRK and IRQ by examining the pushed B-flag only.

The RTI opcode can be used to return from BRK/IRQ/NMI, note that using the return address from BRK skips one dummy/parameter byte following after the BRK opcode.

Software or hardware must take care to acknowledge or reset /IRQ or /NMI signals after processing it.

IRQs are executed whenever "/IRQ=LOW AND I=0".

NMIs are executed whenever "/NMI changes from HIGH to LOW".

If /IRQ is kept LOW then same (old) interrupt is executed again as soon as setting I=0. If /NMI is kept LOW then no further NMIs can be executed.

CPU Control

18	--0---	2	CLC	Clear carry flag	C=0
58	---0--	2	CLI	Clear interrupt disable bit	I=0
D8	----0-	2	CLD	Clear decimal mode	D=0
B8	-----0	2	CLV	Clear overflow flag	V=0
38	--1---	2	SEC	Set carry flag	C=1
78	---1--	2	SEI	Set interrupt disable bit	I=1
F8	----1-	2	SED	Set decimal mode	D=1

No Operation

EA	-----	2	NOP	No operation	No operation
----	-------	---	-----	--------------	--------------

Conditional Branch Page Crossing

The branch opcode with parameter takes up two bytes, causing the PC to get incremented twice (PC=PC+2), without any extra boundary cycle. The signed parameter is then added to the PC (PC+disp), the extra clock cycle occurs if the addition crosses a page boundary (next or previous 100h-page).

CPU Illegal Opcodes

SAX and LAX

87	nn	-----	3	SAX nn	STA+STX	[nn]=A AND X
97	nn	-----	4	SAX nn, Y	STA+STX	[nn+Y]=A AND X
8F	nn nn	-----	4	SAX nnnn	STA+STX	[nnnn]=A AND X
83	nn	-----	6	SAX (nn, X)	STA+STX	[WORD[nn+X]]=A AND X
A7	nn	nz----	3	LAX nn	LDA+LDX	A, X=[nn]
B7	nn	nz----	4	LAX nn, Y	LDA+LDX	A, X=[nn+Y]
AF	nn nn	nz----	4	LAX nnnn	LDA+LDX	A, X=[nnnn]
BF	nn nn	nz----	4*	LAX nnnn, X	LDA+LDX	A, X=[nnnn+X]
A3	nn	nz----	6	LAX (nn, X)	LDA+LDX	A, X=[WORD[nn+X]]
B3	nn	nz----	5*	LAX (nn), Y	LDA+LDX	A, X=[WORD[nn]+Y]

For SAX, both A and X are output to databus, LOW-bits are stronger than HIGH-bits, resulting in a "forceful" AND operation.

For LAX, the same value is written to both A and X.

Combined ALU-Opcodes

Opcode high-bits, flags, commands:

```

00+yy          nzc---  SLO op    ASL+ORA    op=op  SHL 1 // A=A OR op

```

20+yy	nzc---	RLA op	ROL+AND	op=op RCL 1 //	A=A AND op
40+yy	nzc---	SRE op	LSR+EOR	op=op SHR 1 //	A=A XOR op
60+yy	nzc--v	RRA op	ROR+ADC	op=op RCR 1 //	A=A ADC op
C0+yy	nzc---	DCP op	DEC+CMP	op=op-1	// A-op
E0+yy	nzc--v	ISC op	INC+SBC	op=op+1	// A=A-op cy?

Opcode low-bits, clock cycles, operands:

07+xx nn	5	nn	[nn]
17+xx nn	6	nn, X	[nn+X]
03+xx nn	8	(nn, X)	[WORD[nn+X]]
13+xx nn	8	(nn), Y	[WORD[nn]+Y]
0F+xx nn nn	6	nnnn	[nnnn]
1F+xx nn nn	7	nnnn, X	[nnnn+X]
1B+xx nn nn	7	nnnn, Y	[nnnn+Y]

Other Illegal Opcodes

0B nn	nzc---	2	ANC #nn	AND+ASL	A=A AND nn
2B nn	nzc---	2	ANC #nn	AND+ROL	A=A AND nn
4B nn	nzc---	2	ALR #nn	AND+LSR	A=(A AND nn) *2 MUL2???
6B nn	nzc--v	2	ARR #nn	AND+ROR	A=(A AND nn) /2
8B nn	nz----	2	XAA #nn ((2))	TXA+AND	A=X AND nn
AB nn	nz----	2	LAX #nn ((2))	LDA+TAX	A, X=nn
CB nn	nzc---	2	AXS #nn	CMP+DEX	X=A AND X -nn cy?
EB nn	nzc--v	2	SBC #nn	SBC+NOP	A=A-nn cy?
93 nn	-----	6	AHX (nn), Y ((1))		[WORD[nn]+Y] = A AND X AND H
9F nn nn	-----	5	AHX nnnn, Y ((1))		[nnnn+Y] = A AND X AND H
9C nn nn	-----	5	SHY nnnn, X ((1))		[nnnn+X] = Y AND H
9E nn nn	-----	5	SHX nnnn, Y ((1))		[nnnn+Y] = X AND H
9B nn nn	-----	5	TAS nnnn, Y ((1))	STA+TXS	S=A AND X // [nnnn+Y]=S AND H
BB nn nn	nz----	4*	LAS nnnn, Y	LDA+TSX	A, X, S = [nnnn+Y] AND S

NUL/NOP and KIL/JAM/HLT

xx	-----	2	NOP	(xx=1A, 3A, 5A, 7A, DA, FA)
xx nn	-----	2	NOP #nn	(xx=80, 82, 89, C2, E2)
xx nn	-----	3	NOP nn	(xx=04, 44, 64)
xx nn	-----	4	NOP nn, X	(xx=14, 34, 54, 74, D4, F4)
xx nn nn	-----	4	NOP nnnn	(xx=0C)
xx nn nn	-----	4*	NOP nnnn, X	(xx=1C, 3C, 5C, 7C, DC, FC)
xx	-----	-	KIL	(xx=02, 12, 22, 32, 42, 52, 62, 72, 92, B2, D2, F2)

NOP doesn't change any registers or flags, the operand (if any) is fetched, may be useful for delays, patches, or for read-sensitive I/O ports. KIL halts the CPU, the data bus will be set to #FFF, KIL can be suspended by /RESET signal (not sure if also by /IRQ or /NMI ???).

note to ANC: this command performs an AND operation only, but bit 7 is put into the carry, as if the ASL/ROL would have been executed.

note to ARR: part of this command are some ADC mechanisms. following effects appear after AND but before ROR: the V-Flag is set according to (A and #{imm})+#{imm}, bit 0 does NOT go into carry, but bit 7 is exchanged with the carry.

note to XAA: DO NOT USE!!! Highly unstable!!!

note to LAX: DO NOT USE!!! On my C128, this opcode is stable, but on my C64-II it loses bits so that the operation looks like this: ORA #? AND #{imm} TAX.

note to AXS: performs CMP and DEX at the same time, so that the MINUS sets the flag like CMP, not SBC.

Combinations of STA/STX/STY:

note: sometimes the &H drops off. Also page boundary crossing will not work as expected (the bank where the value is stored may be equal to the value stored).

~~STX {adr} = stores X&H into {adr}
 SHX {adr} = stores X&H into {adr}
 SHY {adr} = stores Y&H into {adr}~~

CPU Assembler Directives

Below are some common 65XX assembler directives, and the corresponding expressions in 80XX-style language.

65XX-style	80XX-style	Expl.
*= $\$c100$	org 0c100h	sets the assumed origin in memory
=+8	org \$+8	increments origin, does NOT produce data
label	label:	sets a label equal to the current address
label= $\$dc00$	label equ 0dc00h	assigns a value or address to label
.by \$00	db 00h	defines a (list of) byte(s) in memory
.byt \$00	defb 00h	same as .by and db
.wd \$0000	dw 0000h	defines a (list of) word(s) in memory
.end	end	indicates end of source code file
nn	[nn]	force 16bit "00nn" instead 8bit "nn"
#<nnnn	nnnn AND 0FFh	isolate lower 8bits of 16bit value
#>nnnn	nnnn DIV 100h	isolate upper 8bits of 16bit value

CPU The 65XX Family

Different versions of the 6502:

All of these processors are the same concerning the software-side:

```
6501
6502  Used in the CBM floppies and some other 8 bit computers.
6507  Used in Atari 2600, 28pins (only 13 address lines, no /IRQ, no /NMI).
6510  Used in C64, with one built-in I/O port.
8500  Used in C64-II, with different pin-outs.
8502  Used in C128s.
```

Some processors of the family which are not 100% compatible:

```
65C02  Extension of the 6502, used in the C16, C116 and the Plus/4 computers.
65SC02 Small version of the 65C02 which lost a few opcodes again.
65CE02 Extension of the 65C02, used in the C65.
65816  Extended 6502 with new opcodes and 16 bit operation modes.
2A03   Nintendo NES/Famicom, modified CPU, built-in sound/video registers.
```

CPU Local Usage

CPU

The NES uses a customized NMOS 6502 CPU, engineered and produced by Ricoh. It's primary customization adds audio. Audio registers are mapped internal to the CPU; all waveform generation is done internal to the CPU as well.

The NES's 6502 does not contain support for decimal mode. Both the CLD and SED opcodes function normally, but the 'd' bit of P is unused in both ADC and SBC. It is common practice for games to CLD prior to code execution, as the status of 'd' is unknown on power-on and on reset.

NMIs may be generated by PPU each VBlank.

IRQs may be generated by APU and by external hardware.

The CPU does include undocumented opcodes, just like normal 6502 CPUs.

The NTSC NES runs at 1.7897725MHz, and 1.773447MHz for PAL. Which is pretty fast for a 6502

compatible CPU, for example C64 used only 1MHz, and Atari 2600 only 1.2MHz.

Hardware Pin-Outs

Pin-Outs

[Cartridge Pin-Outs](#)

[Controllers - Pin-Outs](#)

[Chipset Pin-Outs](#)

[NES Expansion Port](#)

Upgrading

[Nocash SRAM Circuit](#)

Chipset Pin-Outs

2A03 Pin-Outs & Signal Description (CPU and APU)

Pin	Name	Dir	Expl.
1	ROUT	Out	Sound channel 1+2 output
2	COU2	Out	Sound channel 3+4+5 output
3	/RES	In	Resets several internal 2A03 registers, and the 6502.
4-19	A0-15	Out	Address Bus
20	GND	-	Supply Ground
21-28	D7-0	I/O	Data Bus
29	CLK	In	Master clock input (236,250/11 MHz), clocks an internal divide-by-12 counter.
30	?	In	Normally grounded in NES/FC consoles, this pin has unknown functionality. I suspect that it is an input controlling something, since the pin does draw a little current.
31	PHI2	Out	Divide-by-12 result of the CLK signal (1.79 MHz). The internal 6502 along with function generating hardware, is clocked off this frequency, and is available externally here so that it can be used as a data bus enable signal (when at logic level 1) for external 6502 address decoder logic. The signal has a 62.5% duty cycle.
32	/IRQ	In	Interrupt Request (Low)
33	/NMI	In	Non-Maskable Interrupt (on High-to-Low Transition)
34	R/W	Out	Direction of 6502's data bus (0=Write/Out, 1=Read/In)
35	/JOY2	Out	Low if A0-A15=4017h, R/W=0, PHI2=1
36	/JOY1	Out	Low if A0-A15=4016h, R/W=0, PHI2=1
37-39	J2-0	Out	Bit2-0 of internal register 4016h (Bit0 = Joystick strobe)
40	VCC	-	Supply +5VDC

2C02 Pin-Outs & Signal Descriptions (PPU)

(*) On Famicom consoles, /SYNC is always tied to logical one. On the NES however, this pin is tied in with the 2A03's reset input, and as a result, the picture is always disabled while the reset switch is held in on an NES.

Pin	Name	Dir	Expl.
1	In	CPU R/W	Direction when /CS=LOW
2-9	I/O	CPU D0-D7	Data when /CS=LOW
11	In	CPU A2-A0	Register Select when /CS=LOW
13	In	CPU /CS	CPU read/write to/from PPU Registers
14-17	I/O	EXT0-EXT3	External Master/Slave Video signal (not used)
18	In	CLK	21.47727MHz NTSC, 26.601712MHz PAL

NES Expansion Port

20	In	VEE	Blank, LOW max 20 scanlines or until acknowledged
20	In	VEE GND	Supply Ground
21	Out	VOUT	Composite Video output
29	In	/VSYNC	External Master / Slave for use by slave (not used) (*)

NES Expansion Port, 48 pins (at bottom of console, rarely used)

23, 24	Out	PPU /W, /R	Video memory Write/Read requests
25-30	Out	PPU A13-A8	Video memory MSB-address lines
31-38	I/O	PPU AD7	Video memory LSB address and data lines
39	Out	PPU ALE	VDD voltage from external power supply (usually +10VDC)
39	Out	PPU ALE	Address Latch Enable, HIGH when A0-A7 output at AD0-AD7
40	In	VCC	AIN (Audio Input) Supply

21,22,23,24	Out	VOUT, AOUT (Video and Audio Outputs)
4,14,25-32	I/O	CPU /NMI, /IRQ, D7, D6, D5, D4, D3, D2, D1, D0
5,24	Out	CPU A15, CIC 4MHz
6-10,38-42	I/O	Cart Pin 51-55,20-16
43,44,45	Out	OUT0, OUT1, OUT2 (Port 4016h Bit0-2 Outputs)
34 and 37	Out	PORT0-CLK (both pins) (CPU Read from Port 4016h)
11 and 17	Out	PORT1-CLK (both pins) (CPU Read from Port 4017h)
35,12,33,13,36	In	PORT0-0,1,2,3,4 (Port 4016h Bit0-3 Inverted Inputs)
19,20,15,16,18	In	PORT1-0,1,2,3,4 (Port 4017h Bit0-3 Inverted Inputs)
46	-	Unused

Nocash SRAM Circuit

Step 1 - Basic Connection, NROM support (32K+8K), Horizontal Mirroring

VCC	----- A13-A19	VCC	-----tmp- A15-A18	VCC	-----tmp- A13-A18
CPU A0-A12	-- A0-A12	CPU A0-A14	--- A0-A14	PPU A0-A12	-- A0-A12
CPU D0-D7	--- D0-D7	CPU D0-D7	---- D0-D7	PPU D0-D7	--- D0-D7
CPU /PRG	---- /CS	CPU /PRG	----- /CS	PPU A13 -tmp-	/CS
LPT /LF -tmp-	/OE BIOS	LPT /SEL	----- /OE WRAM	PPU /R	----- /OE VRAM
	_____	CPU R/W	----- /WE	PPU /W	----- /WE
VCC	- _____ -	CPU /RESET			
CIC /RESET	--cut--	CPU /RESET	FLOPPY 5VDC	-----	VCC (supply)
CIC /RESET	-- < --	CPU /RESET	LPT GND	-----	GND
LPT /INIT	-- < --	CPU /RESET	PPU /A13	--tmp--	NES /VCS
LPT /STROBE	-- < --	CPU /NMI	PPU A10	--tmp--	NES VA10
CIC MODE	--cut--	VCC (lockout)	LPT BUSY	-----	CPU OUT2
CIC MODE	-----	GND (no lockout)	LPT D7	-----	EXP PORT0-1

At this stage, the console won't work if an external cartridge is inserted.

Step 2 - Horizontal or Vertical Mirroring Control

LPT D0	---- OR	_____ AND	_____ NES VA10 (out)	PPU A10	--undo-- NES VA10
PPU A10	--- _____	7411			
LPT D1	---- OR	_____	tmp VCC (third AND-input, used in Step 3)		
PPU A11	--- _____	_____			

Step 3 - Internal Circuit Disable (Required for Internal Circuit only)

CPU /PRG	-- OR	SLOT /PRG	LPT /LF	-- AND	CART	VCC	- _____ -	LPT /LF
/CART	----- _____		LPT /SEL	- _____		VCC	- _____ -	LPT /SEL
PPU /R	---- OR	SLOT /R	CART	----- NAND	/CART	VCC	- _____ -	LPT D0
/CART	----- _____		CART	----- _____		VCC	- _____ -	LPT D1
CART	----- OR	VRAM /CS	CPU A14	-- AND	SLOT A14			
PPU A13	--- _____		CART	----- _____	NES /VCS	--cut--	SLOT /VCS	
/CART	----- OR	AND (Step 2)			NES VA10	--cut--	SLOT VA10	
SLOT VA10	- _____		CPU A14	--cut--	SLOT A14			
CART	----- OR		CPU /PRG	--cut--	SLOT /PRG			
PPU /A13	-- _____	AND	PPU /R	--cut--	SLOT /R			
SLOT /VCS	- OR	_____	VCC	--undo--	AND (Step 2)			
/CART	----- _____	PPU A13 -undo-	VRAM /CS	PPU /A13	--undo-	NES /VCS		

Also allows to disable the internal circuit so that external cartridges can be used when LPT cable is disconnected (or when LPT signals are all HIGH).

Step 4 - UNROM (N*16K+8K) and CNROM (32K+N*8K) Bank Switching

LPT /LF ---- /CLKEN1	LPT LF ----- /CLKEN1	WRAM A14 -- -- VCC
CPU R/W ---- /CLKEN2	CPU R/W ---- /CLKEN2	WRAM A15 -- -- VCC
CPU /PRG --- CLK	CPU /PRG --- CLK	WRAM A16 -- -- VCC
CPU D0..3 -- Q0..3	CPU D0..3 -- Q0..3	LPT /LF --- NAND\ LPT LF
GND ----- /OE1	CPU A14 ---- /OE1	LPT /LF ---
GND ----- /OE2 74173	GND ----- /OE2 74173	LPT /SEL -- NAND\ BIOS
GND ----- RST CNROM	GND ----- RST UNROM	LPT LF --- /OE
VRAM A13-16- D0..3	WRAM A14-16- D0..2	WRAM A15-A16 --undo-- VCC
		VRAM A13-A16 --undo-- VCC
LPT /LF --undo-- BIOS /OE		WRAM A14 --undo-- CPU A14

Step 5 - Optional 8bit high-speed upload connection

CPU A13 -- AND	CPU /PRG - AND	CPU D0-7-- Q0-7 D0-7	--LPT D0-7
CPU A14 -- 7411	CPU R/W -- 7411	VCC- NAND\	/OE1 /OE2
CPU PHI2 -	-----	-----	74541

The 1bit PORT0-1 connection is no longer used (may be disconnected if desired).

Compatibility Notes

WRAM and VRAM are not write-protected, and may get overwritten by accidental writes to ROM/VRAM area, that applies also for writes to bank selection ports (no problem for cartridges that handle bus-conflicts, it will simply replace the value in RAM by the (same) written value). In NROM mode, bank selection ports are not protected against accidental ROM-area writes.

Soldering Notes

To reduce the amount of wires, the WRAM/VRAM chips can be stacked on top of the internal 2K SRAMs with 1:1 connection for most pins, also the BIOS EPROM socket can be stacked on the WRAM chip.

Optionally, the circuit could be connected externally to the cartridge slot (with /RESET and /NMI connected to unused cartridge/expansion port pins), the /CART and CART signals would be not required, Step 3 could be left out.

Parts List

2	SRAM	WRAM/VRAM, min 32K/8K, recommended 128K/32K, max 128K/128K
1	EPROM	BIOS, 27C64 or similar, min 8K
2	74LS32	quad 2-input OR gates
1	74LS08	quad 2-input AND gates
1	74LS00	quad 2-input NAND gates
1	74LS11	triple 3-input AND gates
2	74LS173	4-bit 3-state flip-flops
1	74LS541	8-bit 3-state buffer/line driver
8	10K	pull-up resistors
3	1N4148	diodes for /RESET and /NMI

Plus, eprom socket, optionally also sockets for all other chips, 100nF capacitors for power supply of all chips, centronics printer cable, centronics socket, wire, board, solder, eprom burner, etc.

BIOS ROM-Image

```

0000 85 04 48 8A 48 A9 19 8D FA FF A9 04 8D FB FF A2
0010 08 E0 00 D0 FC 68 AA 68 60 A9 00 06 04 69 03 8D
0020 16 40 CA 40 8A 48 A9 3B 8D FA FF A9 04 8D FB FF
0030 A2 08 E0 00 D0 FC 68 AA A5 04 60 24 0D 30 09 AD
0040 16 40 4A 4A 26 04 CA 40 AD 00 60 85 04 A2 00 40
0050 20 24 04 A2 7E A0 04 24 0D 30 04 A2 8D A0 04 8E
0060 FA FF 8C FB FF 8D FF FF A9 00 8D 01 20 8D 06 20
0070 8D 06 20 A2 00 A0 20 A9 01 85 04 4C 7B 04 AD 00

```

```

0080 60 8D 07 20 CA D0 FE 88 D0 FE 4C 1C 06 AD 16 40
0090 4A 4A 26 04 90 FE A5 04 8D 07 20 A9 01 85 04 CA
00A0 D0 FE 88 D0 FE 4C 1C 06 20 24 04 A2 DC A0 04 24
00B0 0D 30 04 A2 EE A0 04 8E FA FF 8C FB FF 8D FF FF
00C0 A0 BF A2 FF C9 FF D0 04 A0 FF A2 F9 8C FC 04 8C
00D0 E2 04 E8 A0 40 A9 01 85 04 4C D9 04 AD 00 60 CA
00E0 9D 00 FF D0 FE CE E2 04 88 D0 FE 4C 1C 06 AD 16
00F0 40 4A 4A 26 04 90 FE A5 04 CA 9D 00 FF A9 01 85
0100 04 E0 00 D0 FE CE FC 04 88 D0 FE 4C 1C 06 A2 00
0110 20 24 04 95 05 E8 E0 08 D0 F6 A2 00 B5 05 9D FA
0120 FF E8 E0 06 D0 F6 A1 05 81 05 4C 2A 05 A2 55 A0
0130 AA 8E FE FF 8C FF FF EC FE FF D0 F5 CC FF FF D0
0140 F0 8C FE FF 8E FF FF CC FE FF D0 E5 EC FF FF D0
0150 E0 60 A2 00 BD 61 05 20 00 04 E8 BD 61 05 D0 F4
0160 60 4E 4F 24 4E 45 53 20 42 49 4F 53 20 56 31 2E
0170 30 00 A9 00 85 0D 20 87 05 A9 80 85 0D 20 87 05
0180 F0 04 A9 00 85 0D 60 A2 00 A0 2B 20 24 04 DD A1
0190 05 F0 02 A0 2D E8 E0 08 D0 F1 98 20 00 04 C9 2B
01A0 60 00 FF 55 AA 0F F0 3C C3 A9 57 20 00 04 A2 FF
01B0 8D FF FF E8 8E 00 80 8E FF BF EC FF FF F0 06 E0
01C0 1F D0 F0 A2 01 E8 8A 20 00 04 60 A9 56 20 00 04
01D0 A2 40 A0 56 A9 00 8D 01 20 CA 8E FF FF 8D 06 20
01E0 8D 06 20 8C 07 20 8E 07 20 D0 EE A2 00 8E FF FF
01F0 8D 06 20 8D 06 20 CD 07 20 CC 07 20 D0 0A EC 07
0200 20 D0 05 E8 E0 40 D0 E5 8A 20 00 04 60 20 2D 05
0210 20 52 05 20 72 05 20 A9 05 20 CB 05 A9 52 20 00
0220 04 20 24 04 C9 57 D0 03 4C A8 04 C9 56 D0 03 4C
0230 50 04 C9 46 D0 03 4C 0E 05 4C 39 06 A2 00 BD 00
0240 E0 9D 00 04 BD 00 E1 9D 00 05 BD 00 E2 9D 00 06
0250 BD 00 E3 9D 00 07 E8 D0 E5 60 78 D8 A9 00 8D 00
0260 20 AD 02 20 A2 FF 9A 20 3C E2 4C 0D 06 FF FF FF
0270 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
.... FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1FF0 FF FF FF FF FF FF FF FF FF FF 00 00 5A E2 00 00

```

To be copied to the highest memory location, ie. E000h-FFFFh for a 64K EPROM.

About Everynes

Everynes

Everything about NES and Famicom.

Nocash Technical Specifications written 2004 by Martin Korth.

Everynes Text and Html and Debugger versions and updates available at:

<http://nocash.emubase.de/nes.htm>

I've originally written Everynes when collecting and sorting-out relevant info for making the no\$nes emulator/debugger. I've included a copy of the resulting document in the debuggers help text, and also released raw txt/htm versions, which may be eventually of some use to NES/Famicom programmers.

Help welcome

Please let me know if you come across anything that is incomplete, incorrect, or unclear. A couple of details (marked by question marks) are definitely unclear to me - additional info would be very welcome! My email address hides in no\$nes.exe about box (for anti-spam reasons).

Thanks & Credits

Most of the Everynes document is based on information from many other documents found at nesdev.parodius.com - hoping that nobody gets angry about picking info from his/her docs - I'd like to send many thanks to the authors of that great documents, and to all people whom have contributed information to those docs, complete list as far as known to me - many thanks to:

MAPPERS.NFO

Comprehensive NES Mapper Document v0.80 by \Firebug\

Thanks to FanWen, Y0SHi, D, Jim Geffre, Goroh, Paul Robson, Mark Knibbs.

2A03TECH.TXT

2A03 technical reference by Brad Taylor

Thanks to Matthew Conte, Kentaro Ishihara, Goroh, Memblers, FluBBa, Izumi, Chibi-Tech, Quietust, SnowBro, Bananmos, Kevin Horton, and many others for their time and help on and off the NESdev mailing list, and the Membled Messageboards.

2C02TECH.TXT

NTSC 2C02 technical reference by Brad Taylor

Thanks to the NES community. <http://nesdev.parodius.com>.

Special thanks to Neal Tew for scrolling information.

NESTECH2.TXT

Nintendo Entertainment System Documentation, Version: 2.00

Alex Krasivsky

Andrew Davie

Avatar Z

Barubary

Bluefoot

CiXeL

Chi-Wen Yang

Chris Hickman

D

Dan Boris

David de Regt

Donald Moore

Fredrik Olsson

Icer Addis

Jon Merkel

Kevin Horton

Loopy

Marat Fayzullin

Mark Knibbs

Martin Nielsen

Matt Conte

Matthew Richey

Memblers

MiKael Iushin

Mike Perry

Morgan Johansson

Neill Corlett

Pat Mccomack

Patrik Alexandersson

Paul Robson

Ryan Auge

Stumble

Tennessee Carmel-Veilleux

Thomas Steen

Tony Young

Vince Indriolo

\FireBug\

FFPA.TXT

Famicom Four-Player Adapters Technical Document by Richard Hoelscher

NES4PLAY.TXT

NES 4player-adapter documentation by Fredrik Olsson

Special thanks to:

Juan Antonio Gomez Galvarez

Yoshi

Marat Fayzulin

Morgan Johansson

Pin-Outs

drk421

Siudym'2001

6205BUGS.TXT

Ivo van Poorten

NLOCKOUT.TXT

by Mark (Knipps?)

VSDOC.TXT

VS Unisystem information version 1.0, by Fx3

PC10DOC.TXT

Nintendo Playchoice 10 Hardware Description by Oliver Achten

NESGG.TXT

NES Game Genie Code Format DOC v0.71 by Benzene of Digital Emulations

Special thanks to Sardu, Opcode, Deuce, DrSplat, KingPin

NESFQ.HTM

NES Tech FAQ by Chris Covell

NES.HTM

Nintendo Entertainment System Architecture by Marat Fayzullin

Pascal Felber	Patrick Lesaard	Tink
Goroh	Pan of Anthrox	Bas Vijfwinkel
Kawasedo	Paul Robson	
Marcel de Kogel	Serge Skorobogatov	
Alex Krasivsky	John Stiles	

KEYBOARD.TXT

Reverse Engineering the Keyboard of Family Computer

by goroh, english translation by Ki

LIGHTGUN.TXT

Family Computer Gun

by goroh, english translation by Ki

POWERPAD.TXT

Power Pad information Version: 1.2 (03/12/00) by Tennessee Carmel-Veilleux

Thanks to Jeremy D. Chadwick, Kevin Horton

CPU

Project64, Graham

