Nintendo Entertainment System Documentation

Version: 2.00

Y0SHi

Contents

1 Introduction

A Disclaimer

I am in no way held responsible for the results of this information. This is public-domain information, and should not be used for commercial purposes. If you wish to use this document for commercial purposes, please contact me prior to development, so that I may discuss the outline of your project with you. I am not trying to hinder anyone financially: if you wish to do real Nintendo Entertainment System development, contacting either Nintendo of America or Nintendo Company, Ltd. would be wise. Their addresses are listed here:

Nintendo of America
P.O. Box 957
Redmond, WA 98073
USA

Nintendo Company, Ltd.
60 Fukuine
Kamitakamatsu-cho,
Higashiyama-ku,
Koyoto 602, Japan

All titles of cartridges and console systems are registered trademarks of their respective owners. (I just don't deem it necessary to list every single one by hand).

B Why?

At the time this document was created, there was only one piece of literature covering the internals to the NES: Marat Fayzullin's documentation, otherwise known as "NES.DOC".

While Fayzullin's documentation was lacking in many areas, it provided a strong base for the basics, and in itself truly stated how complex the little grey box was.

I took the opportunity to expand on "NES.DOC," basing other people's findings, as well as my own, on experience; experience which has helped make this document what it has become today. The beginning stages of this document looked almost like a replica of Fayzullin's documentation, with both minor and severe changes. Marat Fayzullin himself later picked up a copy of my documentation, and later began referring people to it.

Keep in mind, without Marat's "NES.DOC" document, I would have never had any incentive to write this one.

C Mission

The goal of this document is simplistic: to provide accurate and up-to-date information regarding the Nintendo Entertainment System, and it's Famicom counterpart.

D Dedications

I'd like to dedicate this document to Alex Krasivsky. Alex has been a great friend, and in my eyes, truly started the ball of emulation rolling. During the good times, and the bad times, Alex was there. Spasibo, Alex; umnyj Russki...

E "Thank You"s

I'd like to take the time and thank all the people who helped make this document what it is today. I couldn't have done it without all of you.

Alex Krasivsky	- bcat@lapkin.rosprint.ru
Andrew Davie	
Avatar Z	- swahlen@nfinity.com
Barubary	- barubary@mailexcite.com
Bluefoot	- danmcc@injersey.com
CiXeL	
Chi-Wen Yang	- yangfanw@ms4.hinet.net
Chris Hickman	- typhoonz@parodius.com
D	- slf05@cc.usu.edu
Dan Boris	- dan.boris@coat.com
David de Regt	- akilla@earthlink.net
Donald Moore	- moore@futureone.com
Fredrik Olsson	- flubba@hem2.passagen.se
Icer Addis	- bldlust@maelstrom.net
Jon Merkel	- jpm5974@omega.uta.edu
Kevin Horton	- khorton@iquest.net
Loopy	- zxcvzxcv@netzero.net
Marat Fayzullin	- fms@cs.umd.edu
Mark Knibbs	- mark_k@iname.com
Martin Nielsen	- mnielsen@get2net.dk
Matt Conte	- itsbroke@classicgaming.com
Matthew Richey	- mr6v@andrew.cmu.edu
Memblers	- 5010.0951@tcon.net
MiKael Iushin	- acc@tulatelecom.ru
Mike Perry	- mj-perry@uiuc.edu
Morgan Johansson	- morgan.johansson@mbox301.swipnet.se
Neill Corlett	- corlett@elwha.nrrc.ncsu.edu
Pat Mccomack	- splat@primenet.com
Patrik Alexandersson	- patrikus@hem2.passagen.se
Paul Robson	- AutismUK@aol.com

Ryan Auge - rauge@hay.net
Stumble - stumble@alpha.pulsar.net
Tennessee Carmel-Veilleux - veilleux@ameth.org
Thomas Steen - Thomas.Steen@no.jotankers.com
Tony Young - KBAAA@aol.com
Vince Indriolo - indriolo@nm.picker.com
\FireBug\ - lavos999@aol.com

Special thanks goes out to Stumble, for providing a myriad of information over IRC, while avoiding sleep to do so.

2 Acronymns

A Internals

CPU - Central Processing Unit: Self-explanitory. The NES uses a standard 6502 (NMOS).

PPU - Picture Processing Unit: Used to control graphics, sprites, and other video-oriented features.

pAPU - pseuedo-Audio Processing Unit: Native to the CPU; generates waveforms via (5) audio channels:: four (4) analogue, and one (1) digital. There is no physical IC for audio processing nor generation inside the NES.

MMC - Multi-Memory Controller: Micro-controllers used in NES games to access memory beyond the 6502 64Kbyte boundary. They can also be used to access extra CHR-ROM, and may be used for "special effects" such as forcing and IRQ, and other things.

VRAM - Video RAM: The RAM which is internal to the PPU. There is 16kbits of VRAM installed in the NES.

SPR-RAM - Sprite RAM: 256 bytes of RAM which is used for sprites. It is not part of VRAM nor ROM, though it's local to the PPU.

PRG-ROM - Program ROM: The actual program-code area of memory. Also can be used to describe areas of code which are external to the actual code area and are swapped in via an MMC.

PRG-RAM - Program RAM: Synonymous with PRG-ROM, except that it's RAM.

CHR-ROM - Character ROM: The VRAM data which is kept external to the PPU, swapped in and out via an MMC, or "loaded" into VRAM during the power-on sequence.

VROM - Synonymous with CHR-ROM.

SRAM - Save RAM: RAM which is commonly used in RPGs such as "Crystalis," the Final Fantasy series, and "The Legend of Zelda."

WRAM - Synonymous with SRAM.

DMC - Delta Modulation Channel: The channel of the APU which handles digital data. Commonly referred to as the PCM (Pulse Code Modulation) channel.

EX-RAM - Expansion RAM: This is the memory used within Nintendo's MMC5,

allowing games to extend the capabilities of VRAM.

B Hardware

NES - Nintendo Entertainment System: Self-explanitory.
Dandy - Synonymous (hardware-wise) with the Famicom.
Famicom - Synonymous with the NES, except for not supporting the RAW method of DMC digital audio playback.
FDS - Famicom Disk System: Unit which sits atop the Famicom, supporting the use of 3" double-sided floppy disks for games.

3 CPU

A General Information

The NES uses a customized NMOS 6502 CPU, engineered and produced by Ricoh. It's primary customization adds audio.

The NTSC NES runs at 1.7897725MHz, and 1.773447MHz for PAL.

B Memory Map

Address	Size	Flags	Description
$0000	$800		RAM
$0800	$800	M	RAM
$1000	$800	M	RAM
$1800	$800	M	RAM
$2000	8		Registers
$2008	$1FF8	R	Registers
$4000	$20		Registers
$4020	$1FDF		Expansion ROM
$6000	$2000		SRAM
$8000	$4000		PRG-ROM
$C000	$4000		PRG-ROM

Flag Legend: M = Mirror of $0000
R = Mirror of $2000-2008 every 8 bytes
(e.g. $2008=$2000, $2018=$2000, etc.)

C Interrupts

The 6502 has three (3) interrupts: IRQ/BRK, NMI, and RESET.

Each interrupt has a vector. A vector is a 16-bit address which specifies a location to "jump to" when the interrupt is triggered.

IRQ/BRK is triggered under two circumstances, hence it's split name: when a software IRQ is executed (the BRK instruction), or when a hardware IRQ is executed (via the IRQ line).

RESET is triggered on power-up. The ROM is loaded into memory, and the 6502 jumps to the address specified in the RESET vector. No registers are modified, and no memory is cleared; these only occur during power-up.

NMI stands for Non-Maskable Interrupt, and is generated by each refresh (VBlank), which occurs at different intervals depending upon the system used (PAL/NTSC).

NMI is updated 60 times/sec. on NTSC consoles, and 50 times/sec on PAL. Interrupt latency on the 6502 is seven (7) cycles; this means it takes seven (7) cycles to move in and out of an interrupt.

Most interrupts should return using the RTI instruction. Some NES carts do not use this method, such as SquareSoft's "Final Fantasy 1" title. These carts return from interrupts in a very odd fashion: by manipulating the stack by hand, and then doing an RTS. This is technically valid, but morally shunned.

The aforementioned nterrupts have the following vector addresses, mapped to areas of ROM:

```
$FFFA = NMI
$FFFC = RESET
$FFFE = IRQ/BRK
```

Interrupt priorities are as follows:

```
Highest = RESET
          NMI
 Lowest = IRQ/BRK
```

D NES-specific Customizations

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.

Audio registers are mapped internal to the CPU; all waveform generation is done internal to the CPU as well.

E Notes

Please note the two separate 16K PRG-ROM segments; they may be linear, but they play separate roles depending upon the size of the cartridge. Some games only hold one (1) 16K bank of PRG-ROM, which should be loaded into both $C000 and $8000.

Most games load themselves into $8000, using 32K of PRG-ROM space. The first game to use this method is Super Mario Brothers. However, all games wit more than one (1) 16K bank of PRG-ROM load themselves into $8000 as well. These games use Memory Mappers to swap in and out PRG-ROM data, as well as CHR-ROM.

When a BRK is encountered, the NES's 6502 pushes the CPU status flag onto the stack with the 'b' CPU bit set. On an IRQ or NMI, the CPU pushes the status flag onto the stack with the 'b' bit clear. This is done because of the fact that a hardware IRQ (IRQ) and a software IRQ (BRK) both share the same vector. For example, one could use the following code to distinguish the difference between the two:

```
C134: PLA                ; Copy CPU status flag
C135: PHA                ; Return status flag to stack
C136: AND #$10           ; Check D4 ('b' CPU bit)
C138: BNE is_BRK_opcode  ; If set then it is a software IRQ (BRK)
```

Executing BRK inside of NMI will result in the pushed 'b' bit being set.

The 6502 has a bug in opcode $6C (jump absolute indirect). The CPU does not correctly calculate the effective address if the low-byte is $FF. Example:

```
C100: 4F
C1FF: 00
C200: 23
  ..
D000: 6C FF C1 - JMP ($C1FF)
```

Logically, this will jump to address $2300. However, due to the fact that the high-byte of the calculate address is *NOT* increased on a page-wrap, this will actually jump to $4F00.

It should be noted that page wrapping does *NOT* occur in indexed-indirect addressing modes. Due to limitations of zero-page, all indexed-indirect read/writes should apply a logical AND #$FF to the effective address after calculation. Example:

```
C000: LDX #3        ; Reads indirect address from $0002+$0003,
C002: LDA ($FF,X)   ;   not $0102+$0103.
```

4 PPU

A General Information

Mirroring (also referred to as "shadowing") is the process of mapping particular addresses or address ranges to other addresses/ranges via hardware.

B Memory Map

Included here are two (2) memory maps. The first is a "RAM Memory Map," which despite being less verbose describes the actual regions which point to physical RAM in the NES itself. The second is a "Programmer Memory Map" which is quite verbose and describes the entire memory region of the NES and how it's used/manipulated.

RAM Memory Map

Address	Size	Description
$0000	$1000	Pattern Table #0
$1000	$1000	Pattern Table #1
$2000	$800	Name Tables
$3F00	$20	Palettes

Programmer Memory Map

Address	Size	Flags	Description
$0000	$1000	C	Pattern Table #0
$1000	$1000	C	Pattern Table #1
$2000	$3C0		Name Table #0
$23C0	$40	N	Attribute Table #0
$2400	$3C0	N	Name Table #1
$27C0	$40	N	Attribute Table #1
$2800	$3C0	N	Name Table #2
$2BC0	$40	N	Attribute Table #2

```
| $2C00   | $3C0  |  N    | Name Table #3      |
| $2FC0   | $40   |  N    | Attribute Table #3 |
| $3000   | $F00  |   R   |                    |
| $3F00   | $10   |       | Image Palette #1   |
| $3F10   | $10   |       | Sprite Palette #1  |
| $3F20   | $E0   |    P  |                    |
| $4000   | $C000 |     F |                    |
+---------+-------+-------+--------------------+
                   C = Possibly CHR-ROM
                   N = Mirrored (see Subsection G)
                   P = Mirrored (see Subsection H)
                   R = Mirror of $2000-2EFF (VRAM)
                   F = Mirror of $0000-3FFF (VRAM)
```

C Name Tables

The NES displays graphics using a matrix of "tiles"; this grid is called a Name Table. Tiles themselves are 8x8 pixels. The entire Name Table itself is 32x30 tiles (256x240 pixels). Keep in mind that the displayed resolution differs between NTSC and PAL units.

The Name Tables holds the tile number of the data kept in the Pattern Table (continue on).

D Pattern Tables

The Pattern Table contains the actual 8x8 tiles which the Name Table refers to. It also holds the lower two (2) bits of the 4-bit colour matrix needed to access all 16 colours of the NES palette. Example:

```
 VRAM    Contents of                    Colour
 Addr   Pattern Table                   Result
------ ---------------                 --------
$0000: %00010000 = $10 --+             ...1.... Periods are used to
  ..   %00000000 = $00   |             ..2.2... represent colour 0.
  ..   %01000100 = $44   |             .3...3.. Numbers represent
  ..   %00000000 = $00   +-- Bit 0     2.....2. the actual palette
  ..   %11111110 = $FE   |             1111111. colour #.
  ..   %00000000 = $00   |             2.....2.
  ..   %10000010 = $82   |             3.....3.
$0007: %00000000 = $00 --+             ........

$0008: %00000000 = $00 --+
  ..   %00101000 = $28   |
  ..   %01000100 = $44   |
  ..   %10000010 = $82   +-- Bit 1
```

```
    ..   %00000000 = $00   |
    ..   %10000010 = $82   |
    ..   %10000010 = $82   |
  $000F: %00000000 = $00 --+
```

The result of the above Pattern Table is the character 'A', as shown in the "Colour Result" section in the upper right.

E Attribute Tables

Each byte in an Attribute Table represents a 4x4 group of tiles on the screen. There's multiple ways to describe what the function of one (1) byte in the Attribute Table is:

* Holds the upper two (2) bits of a 32x32 pixel grid, per 16x16 pixels.
* Holds the upper two (2) bits of sixteen (16) 8x8 tiles.
* Holds the upper two (2) bits of four (4) 4x4 tile grids.

It's quite confusing; two graphical diagrams may help:

```
+------------+------------+
|  Square 0  |  Square 1  |  #0-F represents an 8x8 tile
|   #0  #1   |   #4  #5   |
|   #2  #3   |   #6  #7   |  Square [x] represents four (4) 8x8 tiles
+------------+------------+   (i.e. a 16x16 pixel grid)
|  Square 2  |  Square 3  |
|   #8  #9   |   #C  #D   |
|   #A  #B   |   #E  #F   |
+------------+------------+
```

The actual format of the attribute byte is the following (and corrisponds to the above example):

```
 Attribute Byte
   (Square #)
----------------
    33221100
    ||||||+--- Upper two (2) colour bits for Square 0 (Tiles #0,1,2,3)
    ||||+----- Upper two (2) colour bits for Square 1 (Tiles #4,5,6,7)
    ||+------- Upper two (2) colour bits for Square 2 (Tiles #8,9,A,B)
    +--------- Upper two (2) colour bits for Square 3 (Tiles #C,D,E,F)
```

F Palettes

The NES has two 16-colour "palettes": the Image Palette and the Sprite Palette. These palettes are more of a "lookup table" than an actual

palette, since they do not hold physical RGB values.

D7-D6 of bytes written to $3F00-3FFF are ignored.

G Name Table Mirroring

One should keep in mind that there are many forms of mirroring when understanding the NES. Some methods even use CHR-ROM-mapped Name Tables (mapper-specific).

The NES itself only contains 2048 ($800) bytes of RAM used for Name Tables. However, as shown in Subsection B, the NES has the capability of addressing up to four (4) Name Tables.

By default, many carts come with "horizontal" and "vertical" mirroring, allowing you to change where the Name Tables point into the NES's PPU RAM. This form of mirroring affects two (2) Name Tables simultaneously; you cannot switch Name Tables independently.

The following chart should assist in understanding all the types of mirroring encountered on the NES. Please note that the addresses shown (12-bit in size) refer to the Name Table portion of the NES's PPU RAM; one may consider these synonymous with "$2xxx" in the VRAM region:

Name	NT#0	NT#1	NT#2	NT#3	Flags
Horizontal	$000	$000	$400	$400	
Vertical	$000	$400	$000	$400	
Four-screen	$000	$400	$800	$C00	F
Single-screen					S
CHR-ROM mirroring					C

F = Four-screen mirroring relies on an extra 2048 ($800) of RAM (kept on the cart), resulting in four (4) physical independent Name Tables.

S = Single-screen games have mappers which allow you to select which PPU RAM area you want to use ($000, $400, $800, or $C00); all the NTs point to the same PPU RAM address.

C = Mapper #68 (Afterburner 2) allows you to map CHR-ROM to the Name Table region of the NES's PPU RAM area. Naturally this makes the Name Table ROM-based, and one cannot write to it. However, this feature can be controlled via the mapper itself, allowing you to enable or disable this feature.

H Palette Mirroring

Mirroring occurs between the Image Palette and the Sprite Palette. Any data which is written to $3F00 is mirrored to $3F10. Any data written to $3F04 is mirrored to $3F14, etc. etc...

Colour #0 in the upper three (3) palettes of both the Image and Sprite palette defines transparency (the actual colour stored there is not drawn on-screen).

The PPU uses the value in $3F00 to define background colour.

For a more verbose explanation, assume the following:

* $0D has been written to $3F00 (mirrored to $3F10)
* $03 has been written to $3F08 (mirrored to $3F18)
* $1A has been written to $3F18
* $3F08 is read into the accumulator

The PPU will use $0D as the background colour, despite $3F08 holding a value of $03 (since colour #0 in all the palette entries defines transparency, it is not drawn). Finally, the accumulator will hold a value of $1A, which is mirrored from $3F18. Again, the value of $1A is not drawn, since colour #0 defines transparency.

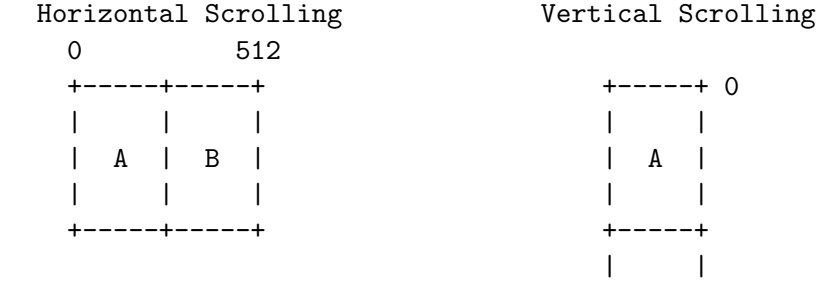
The entire Image and Sprite Palettes are both mirrored to other areas of VRAM as well; $3F20-3FFF are mirrors of both palettes, respectively.

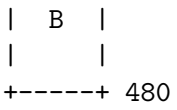
D7-D6 of bytes written to $3F00-3FFF are ignored.

I Background Scrolling

The NES can scroll the background (pre-rendered Name Table + Pattern Table + Attribute Table) independently of the sprites which are overlayed on top of it. The background can be scrolled horizontally and vertically.

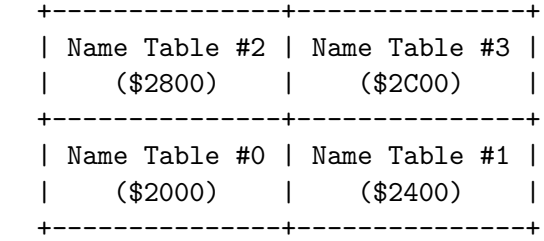
Scrolling works as follows:





Name Table "A" is specified via Bits D1-D0 in register $2000, and "B" is the Name Table after (due to mirroring, this is dynamic). This doesn't work for game which use Horizontal & Vertical scrolling simultaneously.

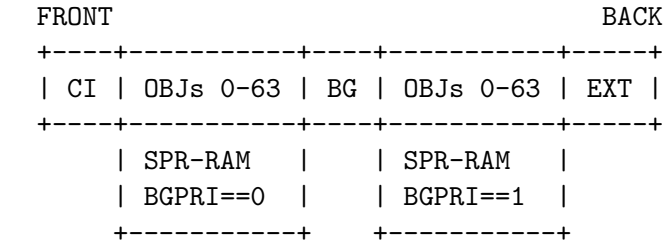
The background will span across multiple Name Tables, as shown here:



Writes to the Horizontal Scroll value in $2005 range from 0 to 256. Writes to the Vertical Scroll value range from 0-239; values above 239 are considered negative (e.g. a write of 248 is really -8).

J Screen and Sprite Layering

There is a particular order in which the NES draws it's contents:



CI stands for 'Colour Intensity', which is synonmous with D7-D5 of $2001. BG is the BackGround, and EXT is for the EXTension port video signal.

'BGPRI' represents the 'Background Priority' bit in SPR-RAM, on a per-sprite basis (D5, Byte 2).

OBJ numbers represent actual Sprite numbers, not Tile Index values.

FRONT is considered what is seen atop all other layers (drawn last), and BACK is deemed what is below most other layers (drawn first).

K Sprites and SPR-RAM

The NES supports 64 sprites, which can be either 8x8 or 8x16 pixels in size. The sprite data is kept within the Pattern Table region of VRAM.

Sprite attributes such as flipping and priority, are stored in SPR-RAM, which is a separate 256 byte area of memory, independent of ROM and VRAM. The format of SPR-RAM is as follows:

```
+-----------+-----------+-----+------------+
| Sprite #0 | Sprite #1 | ... | Sprite #63 |
+-+------+--+-----------+-----+------------+
  |      |
  +------+----------+--------------------------------------+
  + Byte | Bits     | Description                          |
  +------+----------+--------------------------------------+
  |  0   | YYYYYYYY | Y Coordinate - 1. Consider the coor- |
  |      |          | dinate the upper-left corner of the  |
  |      |          | sprite itself.                       |
  |  1   | IIIIIIII | Tile Index #                         |
  |  2   | vhp000cc | Attributes                           |
  |      |          |   v = Vertical Flip   (1=Flip)       |
  |      |          |   h = Horizontal Flip (1=Flip)       |
  |      |          |   p = Background Priority            |
  |      |          |         0 = In front                 |
  |      |          |         1 = Behind                   |
  |      |          |   c = Upper two (2) bits of colour   |
  |  3   | XXXXXXXX | X Coordinate (upper-left corner)     |
  +------+----------+--------------------------------------+
```

The Tile Index # is obtained the same way as Name Table data.

Sprites which are 8x16 in size function a little bit differently. A 8x16 sprite which has an even-numbered Tile Index # use the Pattern Table at $0000 in VRAM; odd-numbered Tile Index #s use $1000. *NOTE*: Register $2000 has no effect on 8x16 sprites.

All 64 sprites contain an internal priority; sprite #0 is of a higher priority than sprites #63 (sprite #0 should be drawn last, etc.).

Only eight (8) sprites can be displayed per scan-line. Each entry in SPR-RAM is checked to see if it's in a horizontal range with the other sprites. Remember, this is done on a per scan-line basis, not on a per sprite basis (e.g. done 256 times, not 256/8 or 256/16 times).

(NOTE: On a real NES unit, if sprites are disabled (D4 of $2001 is 0) for a long period of time, SPR-RAM will gradually degrade. A proposed concept is that SPR-RAM is actually DRAM, and D4 controls the DRAM

refresh cycle).

L Sprite #0 Hit Flag

The PPU is capable of figuring out where Sprite #0 is, and stores it's findings in D6 of $2002. The way this works is as follows:

The PPU scans for the first actual non-transparent "sprite pixel" and the first non-transparent "background pixel." A "background pixel" is a tile which is in use by the Name Table. Remember that colour #0 defines transparency.

The pixel which causes D6 to be set *IS* drawn.

The following example should help. The following are two tiles. Transparent colours (colour #0) are defined via the underscore ('_') character. An asterisk ('*') represents when D6 will be set.

```
 Sprite          BG          Result
 ------          --          ------
__1111__     ________     __1111__
_111111_     _______2     _1111112
11222211     ______21     11222211
112__211  +  _____211  =  112__*11  '*' will be drawn as colour #2
112__211     ____2111     112_2211
11222211     ___21111     11222211
_111111_     __211111     _1111111
__1111__     _2111111     _2111111
```

This also applies to sprites that are underneathe the BG (via the 'Background Priority' SPR-RAM bit), though the above example would be 'BG+Sprite'.

Also, D6 is cleared (set to 0) after each VBlank.

M Horizontal and Vertical Blanking

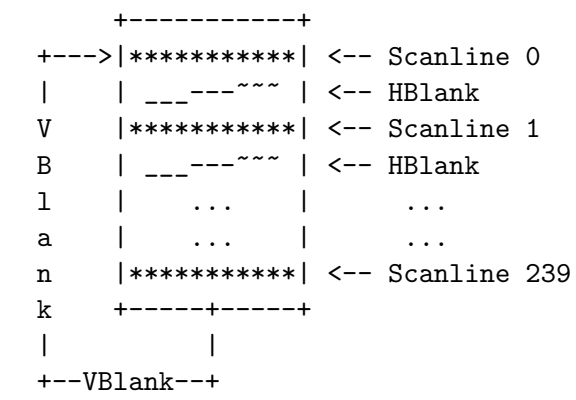
The NES, like every console, has a refresh: where the display device relocates the electron gun to display visible data. The most common display device is a television set. The refresh occurs 60 times a second on an NTSC device, and 50 on a PAL device.

The gun itself draws pixels left to right: this process results in one (1) horizontal scanline being drawn. After the gun is done drawing the entire scanline, the gun must return to the left side of the display device, becoming ready to draw the next scanline. The process

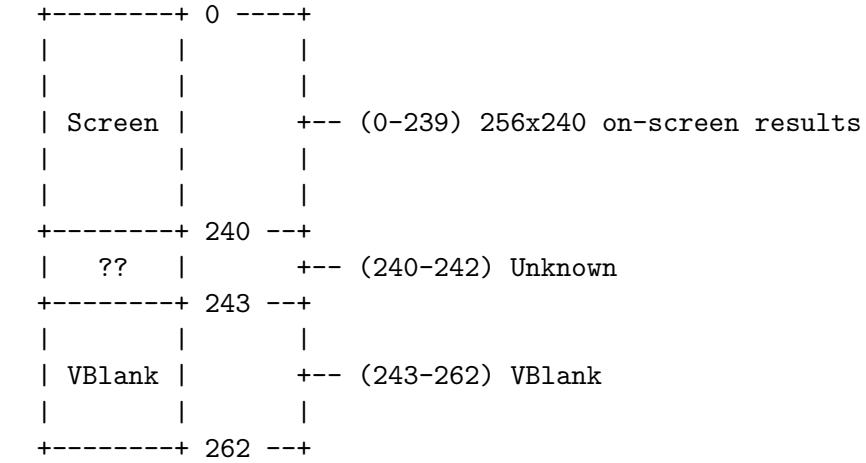
of the gun returning to the left side of the display is the Horizontal Blank period (HBlank).

When the gun has completed drawing all of the scanlines, it must return to the top of the display device; the time it takes for the gun to re-position itself atop the device is called the Vertical Blank period (VBlank).

As you can see from the below diagram, the gun more or less works in a zig-zag pattern until VBlank is reached, then the process repeats:



An NTSC NES has the following refresh and screen layout:



The Vertical Blank (VBlank) flag is contained in D7 of $2002. It indicates whether PPU is in VBlank or not. A program can reset D7 by reading $2002.

N $2005/2006 Magic

For detailed information pertaining to the $2005 and $2006 registers, refer to Loopy's $2005/2006 document. His document provides entirely

accurate information regarding how these registers work. Contact Loopy for more information.

O PPU Quirks

The first read from VRAM is invalid. Due to this aspect, the NES will returned pseudo-buffered values from VRAM rather than linear as expected. See the below example:

VRAM $2000 contains $AA $BB $CC $DD.
VRAM incrementation value is 1.
The result of execution is printed in the comment field.

```
  LDA #$20
  STA $2006
  LDA #$00
  STA $2006        ; VRAM address now set at $2000
  LDA $2007        ; A=??     VRAM Buffer=$AA
  LDA $2007        ; A=$AA    VRAM Buffer=$BB
  LDA $2007        ; A=$BB    VRAM Buffer=$CC
  LDA #$20
  STA $2006
  LDA #$00
  STA $2006        ; VRAM address now set at $2000
  LDA $2007        ; A=$CC    VRAM Buffer=$AA
  LDA $2007        ; A=$AA    VRAM Buffer=$BB
```

As shown, the PPU will post-increment it's internal address data after the first read is performed. This *ONLY APPLIES* to VRAM $0000-3EFF (e.g. Palette data and their respective mirrors do not suffer from this phenomenon).

P Notes

The PPU will auto-increment the VRAM address by 1 or 32 (based on D2 of $2000) after accessing $2007.

5 pAPU

To be written. Prior information was inaccurate or incorrect. No one has 100% accurate sound information at this time. This section will be completed when someone decides to reverse engineer the pAPU section

of the NES, and provide me with information (or a reference to information).

6 Joypads, paddles, expansion ports

A General Information

The NES supports a myriad of input devices, including joypads, Zappers (light guns), and four-player devices.

Joypad #1 and #2 are accessed via $4016 and $4017, respectively.

The joypads are reset via a strobing-method: writing 1, then 0, to $4016. See Subsection H for information regarding "half-strobing."

On a full strobe, the joypad's button status will be returned in a single-bit stream (D0). Multiple reads need to be made to read all the information about the controller.

```
  1 = A          9 = Ignored   17 = +--+
  2 = B         10 = Ignored   18 =    +-- Signature
  3 = SELECT    11 = Ignored   19 =    |
  4 = START     12 = Ignored   20 = +--+
  5 = UP        13 = Ignored   21 = 0
  6 = DOWN      14 = Ignored   22 = 0
  7 = LEFT      15 = Ignored   23 = 0
  8 = RIGHT     16 = Ignored   24 = 0
```

See Subsection G for information about Signatures.

B The Zapper

The Zapper (otherwise known as the "Light Gun") simply uses bits within $4016 and $4017, described in Section 8. See bits D4, D3, and D0.

It is possible to have two Zapper units connected to both joypad ports simultaneously.

C Four-player devices

Some NES games allow the use of a four-player adapter, extending the number of usable joypads from two (2) to four (4). Carts which use

the quad-player device are Tengen's "Gauntlet II," and Nintendo's "RC Pro Am 2."

All four (4) controllers read their status-bits from D0 of $4016 or $4017, as Subsection A states.

For register $4016, reads #1-8 control joypad #1, and reads #9-16 control joypad #3. For $4017, it is respective for joypad #2 and #4.

The following is a list of read #s and their results.

```
 1 = A          9 = A         17 = +--+
 2 = B         10 = B         18 =    +-- Signature
 3 = SELECT    11 = SELECT    19 =    |
 4 = START     12 = START     20 = +--+
 5 = UP        13 = UP        21 = 0
 6 = DOWN      14 = DOWN      22 = 0
 7 = LEFT      15 = LEFT      23 = 0
 8 = RIGHT     16 = RIGHT     24 = 0
```

See Subsection G for information about Signatures.

D Paddles

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.

E Power Pad

No information is currently available.

F R.O.B. (Robot Operated Buddy)

No information is currently available.

G Signatures

A signature allows the programmer to detect if a device is connected to one of the four (4) ports or not, and if so, what type of device it is. Valid/known signatures are:

```
%0000 = Disconnected
%0001 = Joypad ($4016 only)
%0010 = Joypad ($4017 only)
```

H Expansion ports

The joypad strobing process requires dual writes: 1, then 0. If the strobing process is not completed, or occurs in a non-standard order, the joypads are no longer the item of communication: the expansion port is.

For NES users, the expansion port is located on the bottom of the unit, covered by a small grey piece of plastic. Famicom users have a limited expansion port on the front of their unit, which was commonly used for joypads or turbo-joypads.

Such an example of communicating with the expansion port would be the following code:

```
LDA #%00000001
STA $4016
STA $4017           ; Begin read mode of expansion port
LDA #%00000011      ; Write %110 to the expansion port
STA $4016
```

I have yet to encounter a cart which actually uses this method of communication.

I Notes

None.

7 Memory Mapping Hardware

Due to the large number of mappers used (over 64), the "MMC" section

which was once fluid in v0.53 of this document, has now been removed.

All is not lost, as another document by \FireBug\ of Vertigo 2099 contains accurate information about nearly every mapper in existence. You can retrieve a copy via one of the following URLs:

http://free.prohosting.com/~nintendo/mappers.nfo

Please note I take no responsibility for the information contained in the aforementioned document. Contact lavos999@aol.com for more information.

8 Registers

Programmers communicate with the PPU and pAPU via registers, which are nothing more than pre-set memory locations which allow the coder to make changes to the NES. Without registers, programs wouldn't work: period.

Each register is a 16-bit address. Each register has a statistics field in parentheses located immediately after its description. The legend:

```
R = Readable                    W = Writable
2 = Double-write register      16 = 16-bit register
```

NOTE: 16-bit registers actually consist of two linear 8-bit registers, which can (and will be) *INDEPENDANTLY* assigned. The reason for specifying them as 16-bit is for ease of documentation. For instance, "$4002+$4003" would mean that D15-D8 would be in $4003, and D7-D0 would be in $4002.
NOTE: Bits not listed are to be considered unused.

```
+---------+-------------------------------------------------------+
| Address | Description                                           |
+---------+-------------------------------------------------------+
|  $2000  | PPU Control Register #1 (W)                           |
|         |                                                       |
|         |    D7: Execute NMI on VBlank                          |
|         |           0 = Disabled                                |
|         |           1 = Enabled                                 |
|         |    D6: PPU Master/Slave Selection --+                 |
|         |           0 = Master                +-- UNUSED        |
|         |           1 = Slave               --+                 |
|         |    D5: Sprite Size                                    |
|         |           0 = 8x8                                     |
|         |           1 = 8x16                                    |
|         |    D4: Background Pattern Table Address               |
|         |           0 = $0000 (VRAM)                            |
```

```
|         |           1 = $1000 (VRAM)                                  |
|         |    D3: Sprite Pattern Table Address                         |
|         |           0 = $0000 (VRAM)                                  |
|         |           1 = $1000 (VRAM)                                  |
|         |    D2: PPU Address Increment                                |
|         |           0 = Increment by 1                                |
|         |           1 = Increment by 32                               |
|         | D1-D0: Name Table Address                                   |
|         |          00 = $2000 (VRAM)                                  |
|         |          01 = $2400 (VRAM)                                  |
|         |          10 = $2800 (VRAM)                                  |
|         |          11 = $2C00 (VRAM)                                  |
+---------+-------------------------------------------------------------+
|  $2001  | PPU Control Register #2 (W)                                 |
|         |                                                             |
|         | D7-D5: Full Background Colour (when D0 == 1)                |
|         |          000 = None  +------------+                         |
|         |          001 = Green               | NOTE: Do not use more  |
|         |          010 = Blue                |       than one type    |
|         |          100 = Red   +------------+                         |
|         | D7-D5: Colour Intensity (when D0 == 0)                      |
|         |          000 = None             +--+                        |
|         |          001 = Intensify green     | NOTE: Do not use more  |
|         |          010 = Intensify blue      |       than one type    |
|         |          100 = Intensify red    +--+                        |
|         |    D4: Sprite Visibility                                    |
|         |           0 = Sprites not displayed                         |
|         |           1 = Sprites visible                               |
|         |    D3: Background Visibility                                |
|         |           0 = Background not displayed                      |
|         |           1 = Background visible                            |
|         |    D2: Sprite Clipping                                      |
|         |           0 = Sprites invisible in left 8-pixel column      |
|         |           1 = No clipping                                   |
|         |    D1: Background Clipping                                  |
|         |           0 = BG invisible in left 8-pixel column           |
|         |           1 = No clipping                                   |
|         |    D0: Display Type                                         |
|         |           0 = Colour display                                |
|         |           1 = Monochrome display                            |
+---------+-------------------------------------------------------------+
|  $2002  | PPU Status Register (R)                                     |
|         |                                                             |
|         |    D7: VBlank Occurance                                     |
|         |          0 = Not occuring                                   |
|         |          1 = In VBlank                                      |
|         |    D6: Sprite #0 Occurance                                  |
|         |          0 = Sprite #0 not found                            |
|         |          1 = PPU has hit Sprite #0                          |
```

	D5: Scanline Sprite Count 0 = Eight (8) sprites or less on current scanline 1 = More than 8 sprites on current scanline D4: VRAM Write Flag 0 = Writes to VRAM are respected 1 = Writes to VRAM are ignored NOTE: D7 is set to 0 after read occurs. NOTE: After a read occurs, $2005 is reset, hence the next write to $2005 will be Horizontal. NOTE: After a read occurs, $2006 is reset, hence the next write to $2006 will be the high byte portion. For detailed information regarding D6, see Section 4, Subsection L.
$2003	SPR-RAM Address Register (W) D7-D0: 8-bit address in SPR-RAM to access via $2004.
$2004	SPR-RAM I/O Register (W) D7-D0: 8-bit data written to SPR-RAM.
$2005	VRAM Address Register #1 (W2) Commonly used used to "pan/scroll" the screen (sprites excluded) horizontally and vertically. However, there is no actual panning hardware inside the NES. This register controls VRAM addressing lines. Refer to Section 4, Subsection N, for more information.
$2006	VRAM Address Register #2 (W2) Commonly used to specify the 16-bit address in VRAM to access via $2007. However, this register controls VRAM addressing bits, and therefore should be used with knowledge of how it works, and when it works. Refer to Section 4, Subsection N, for more information.
$2007	VRAM I/O Register (RW) D7-D0: 8-bit data read/written from/to VRAM.
$4000 $4001	pAPU Pulse #1 Control Register (W) pAPU Pulse #1 Ramp Control Register (W)

```
|  $4002  | pAPU Pulse #1 Fine Tune (FT) Register (W)                  |
|  $4003  | pAPU Pulse #1 Coarse Tune (CT) Register (W)                |
|  $4004  | pAPU Pulse #2 Control Register (W)                         |
|  $4005  | pAPU Pulse #2 Ramp Control Register (W)                    |
|  $4006  | pAPU Pulse #2 Fine Tune Register (W)                       |
|  $4007  | pAPU Pulse #2 Coarse Tune Register (W)                     |
|  $4008  | pAPU Triangle Control Register #1 (W)                      |
|  $4009  | pAPU Triangle Control Register #2 (?)                      |
|  $400A  | pAPU Triangle Frequency Register #1 (W)                    |
|  $400B  | pAPU Triangle Frequency Register #2 (W)                    |
|  $400C  | pAPU Noise Control Register #1 (W)                         |
|  $400D  | Unused (???)                                               |
|  $400E  | pAPU Noise Frequency Register #1 (W)                       |
|  $400F  | pAPU Noise Frequency Register #2 (W)                       |
|  $4010  | pAPU Delta Modulation Control Register (W)                 |
|  $4011  | pAPU Delta Modulation D/A Register (W)                     |
|  $4012  | pAPU Delta Modulation Address Register (W)                 |
|  $4013  | pAPU Delta Modulation Data Length Register (W)             |
+---------+------------------------------------------------------------+
|  $4014  | Sprite DMA Register (W)                                    |
|         |                                                            |
|         |  Transfers 256 bytes of memory into SPR-RAM. The address  |
|         |  read from is $100*N, where N is the value written.        |
+---------+------------------------------------------------------------+
|  $4015  | pAPU Sound/Vertical Clock Signal Register (R)              |
|         |                                                            |
|         |    D6: Vertical Clock Signal IRQ Availability              |
|         |            0 = One (1) frame occuring, hence IRQ cannot    |
|         |                occur                                       |
|         |            1 = One (1) frame is being interrupted via IRQ  |
|         |    D4: Delta Modulation                                    |
|         |    D3: Noise                                               |
|         |    D2: Triangle                                            |
|         |    D1: Pulse #2                                            |
|         |    D0: Pulse #1                                            |
|         |            0 = Not in use                                  |
|         |            1 = In use                                      |
|         +------------------------------------------------------------+
|         | pAPU Channel Control (W)                                   |
|         |                                                            |
|         |    D4: Delta Modulation                                    |
|         |    D3: Noise                                               |
|         |    D2: Triangle                                            |
|         |    D1: Pulse #2                                            |
|         |    D0: Pulse #1                                            |
|         |            0 = Channel disabled                            |
|         |            1 = Channel enabled                             |
+---------+------------------------------------------------------------+
|  $4016  | Joypad #1 (RW)                                             |
```

```
|         |                                                          |
|         | READING:                                                 |
|         |    D4: Zapper Trigger                                    |
|         |           0 = Pulled                                     |
|         |           1 = Released (not held)                        |
|         |    D3: Zapper Sprite Detection                           |
|         |           0 = Sprite not in position                     |
|         |           1 = Sprite in front of cross-hair              |
|         |    D0: Joypad Data                                       |
|         +----------------------------------------------------------+
|         | WRITING:                                                 |
|         | Joypad Strobe (W)                                        |
|         |                                                          |
|         |    D0: Joypad Strobe                                     |
|         |           0 = Clear joypad strobe                        |
|         |           1 = Reset joypad strobe                        |
|         +----------------------------------------------------------+
|         | WRITING:                                                 |
|         | Expansion Port Latch (W)                                 |
|         |                                                          |
|         |    D0: Expansion Port Method                             |
|         |           0 = Write                                      |
|         |           1 = Read                                       |
+---------+----------------------------------------------------------+
|  $4017  | Joypad #2/SOFTCLK (RW)                                   |
|         |                                                          |
|         | READING:                                                 |
|         |    D7: Vertical Clock Signal (External)                  |
|         |           0 = Not occuring                               |
|         |           1 = Occuring                                   |
|         |    D6: Vertical Clock Signal (Internal)                  |
|         |           0 = Occuring     (D6 of $4016 affected)        |
|         |           1 = Not occuring (D6 of $4016 untouchable)     |
|         |    D4: Zapper Trigger                                    |
|         |           0 = Pulled                                     |
|         |           1 = Released (not held)                        |
|         |    D3: Zapper Sprite Detection                           |
|         |           0 = Sprite not in position                     |
|         |           1 = Sprite in front of cross-hair              |
|         |    D0: Joypad Data                                       |
|         +----------------------------------------------------------+
|         | WRITING:                                                 |
|         | Expansion Port Latch (W)                                 |
|         |                                                          |
|         |    D0: Expansion Port Method                             |
|         |           0 = ???                                        |
|         |           1 = Read                                       |
+---------+----------------------------------------------------------+
```

9 File Formats

A iNES Format (.NES)

```
+--------+------+-----------------------------------------+
| Offset | Size | Content(s)                              |
+--------+------+-----------------------------------------+
|   0    |  3   | 'NES'                                   |
|   3    |  1   | $1A                                     |
|   4    |  1   | 16K PRG-ROM page count                  |
|   5    |  1   | 8K CHR-ROM page count                   |
|   6    |  1   | ROM Control Byte #1                     |
|        |      |   %####vTsM                             |
|        |      |    |  ||||+- 0=Horizontal mirroring     |
|        |      |    |  ||||   1=Vertical mirroring       |
|        |      |    |  |||+-- 1=SRAM enabled             |
|        |      |    |  ||+--- 1=512-byte trainer present |
|        |      |    |  |+---- 1=Four-screen mirroring    |
|        |      |    |  |                                 |
|        |      |    +--+----- Mapper # (lower 4-bits)    |
|   7    |  1   | ROM Control Byte #2                     |
|        |      |   %####0000                             |
|        |      |    |  |                                 |
|        |      |    +--+----- Mapper # (upper 4-bits)    |
|  8-15  |  8   | $00                                     |
| 16-..  |      | Actual 16K PRG-ROM pages (in linear     |
|  ...   |      | order). If a trainer exists, it precedes |
|  ...   |      | the first PRG-ROM page.                 |
| ..-EOF |      | CHR-ROM pages (in ascending order).     |
+--------+------+-----------------------------------------+
```

10 Programming the NES

A General Information

None.

B CPU Notes

None. See Section 11, Subsection B for more possible information.

C PPU Notes

Reading and writing to VRAM consists of a multi-step process:

Writing to VRAM	Reading from VRAM
1) Wait for VBlank	1) Wait for VBlank
2) Write upper VRAM address byte into $2006	2) Write upper VRAM address byte into $2006
3) Write lower VRAM address byte into $2006	3) Write lower VRAM address byte into $2006
4) Write data to $2007	4) Read $2007 (invalid data once)
	5) Read data from $2007

NOTE: Step #4 when reading VRAM is only necessary when reading VRAM data not in the $3F00-3FFF range.

NOTE: Accessing VRAM should only be performed during VBlank. Attempts to access VRAM outside of VBlank will usually result in garbage showing up on the screen. See Section 4, Subsection N for more information regarding why this occurs.

Waiting for VBlank is quite simple:

```
8000: LDA $2002
      BPL $8000
```

Reading $2002 will result in all bits being returned; however, D7 will be reset to 0 after the read is performed.

The actual on-screen palette used by the NES, as stated prior, is not RGB. However, a near-exact replica can be found in common NES emulators today. Contact the appropriate authors of these emulators to obtain a valid RGB palette.

Be sure to clear the internal VRAM address via $2006 semi-often. You will often encounter a situation where a palette fade or a VRAM update will cause the screen "to be trashed" (squares on the screen, or what seem to be graphical "glitches"). The reason for this is that your code took longer than a VBlank. When the VBlank goes to refresh the screen with the data in the PPU, it takes whatever value is in the internal VRAM address and uses that as the starting base for Name Table #0. The solution is to fix your code by re-assigning the VRAM address to $0000 (or $2000), so that the refresh may occur successfully. Such code would be:

```
LDA #$00
STA $2006
```

```
STA $2006
```

You will find code like this in commercial games quite often.

11 Emulation

A General Information

If you're going to be programming an emulator in C or C++, please be familiar with pointers. Being familiar with pointers will help you out severely when it comes to handling mirroring and VRAM addressing. For you assembly buffs out there, obviously pointers are nothing more than indirect addressing -- it's easier to change a 32-bit value than to swap in and out an entire 64K of data.

When SRAM ($6000-7FFF) is disabled, writes to the memory area should be ignored. Reads will possibly return data previously left on the bus, and therefore when emulated should return 0 (or should be trapped).

RAM-based memory areas ($0000-07FF) should *NOT* be zeroed on RESET; they should be zeroed on power on/off. (Technically, the RAM is not zeroed on power on/off either: the RAM will slowly dissapate over time when the unit it off. However, for emulation purposes, please make sure that a cold boot and a warm boot do different things).

See Section 12, Subsection E for Mailing List information.

B CPU Notes

The NES does not use a 65c02 (CMOS) CPU as rumored.

Ignore opcodes which are bad (or support the option of trapping them). Some ROM images out there, such as "Adventures of Lolo" contain bad opcodes, due to dirty connectors on the cartridge during the extraction process (or other reasons).

There are 154 valid opcodes (out of 256 total) on the NES.

C PPU Notes

The formulae to calculate the base address of a Name Table tile number is:

```
  (TILENUM * 16) + PATTERNTABLE
```

Where TILENUM is the tile number in the Name Table, and PATTERNTABLE is the Pattern Table Address defined via register $2000.

It's recommended that DOS programmers use what is known as "MODE-Q," a 256x256x256 "tweaked" video mode, for writing their emulator. Try to avoid Mode-X modes, as they are non-chained, and result in painfully slow graphics. Chained modes (like MODE-13h) are linear, and work best for speedy graphics. Since the NES's resolution is 256x240, the aforementioned "MODE-Q" should meet all necessary requirements.

Most emulators do not limit the number of sprites which can be displayed per scanline, while the actual NES will show flicker as a result of more than eight (8). {Put some more garbage here; it's early...}

Emulators should _NOT_ mask out unused bits within registers; doing so may result in a cart not working.

D APU Notes

To be written.

12 Reference Material

A CPU Information

None.

B PPU Information

None.

C APU Information

None.

D MMC Information

None.

E Mailing Lists

There is a NES Development Mailing List in existence. Contact Mark Knibbs for more information. This list is for anyone who wishes to discuss technical issues about the NES; it is not a list for picking up the latest and greatest information about NES emulators or what not.

F WWW Sites

The following are a list of WWW sites which contain NES-oriented material. If you encounter errors, bad links, or other anomolies while visiting these sites, contact the site authors/owners, NOT me. Thanks.

http://nesdev.parodius.com/

  Contains a verbose amount of documentation regarding anything NES-oriented, including hard-to-find mapper documentation. Seems to be a decent NES information depository.

http://www.ameth.org/~veilleux/NES_info.html

  Currently only contains hardware-oriented material, such as overviews of cart and unit ASICs, mappers, and MMCs. Many pinout diagrams for mappers and NES units are available here. Also provides documentation on NES repair, modifying your NES to give stereo output, applying stereo mixing to your NES, and much much more.

G Hardware Information

The following security bits may be purchased from MCM Electronics (http://www.mcmelectronics.com/):

  For NES carts: 22-1145 (3.8mm security bit)
  For NES units: 22-1150 (4.5mm security bit)

The 4.5mm security screw is also used for the Super Nintendo Entertainment System (SNES), and Nintendo 64.