任天堂产品系统文件

（Pinokio@163.com译、编 2002/10/10）

目录

· 1. 系统简介
· 2. 缩写表
· 3. 中央处理器
· 4. 图形处理器
· 5. 声音处理器
· 6. 手柄、控制器和扩展口
· 7. 内存映像硬件
· 8. I/O端口
· 9. NES文件格式
· 10. 任天堂磁碟机系统

---

1. 系统简介

任天堂主机由 6502处理器和一个特制的图形处理器组成。CPU是 6502，而不是传言中的 65C02（CMOS）。PPU的显存是和 CPU的内存是分离的，可以通过对特殊端口的读/写来操作。卡带可能包含的内容有位于处理器地址 $8000-$FFFF的 ROM，和位于 PPU地址 $0000-$1FFF的 VROM，由于 NES只有 2K的 RAM，因此变量的可用的变量空间只有从 $0000到 $07FF共 8个页面。在开机之后 RAM和 VRAM中的内容是 0，但是注意：复位并不改变其中的内容。在更小的卡带，比如只有 16KB的 ROM，它占有 $C000-$FFFF，而 $8000-$BFFF的空间是不用的。那些大于 32KB的卡带，它被特殊的电路分页到一定的地址空间。一些卡带在 $6000-$7FFF有 SRAM，那是电池存储的位置。卡带 VROM被用来做图案表（例如 Tile表，角色发生器等等）。通常的数量是 8KB，包含两个图案表。大于 8KB的 VROM被特殊的电路分页到一定的地址空间。内部的 VRAM在 PPU内存里定位于 $2000-$3FFF，它用来存储命名表（例如屏幕缓冲）。虽然 PPU支持 4个命名表，但只能支持两个的存放空间。另外的两个是开始两个的镜像。NES共有 154条指令。

在本文本里，你将遇到如下形式的符号：“Dn”（5位，3位，等等）。位是按从最低位（0位）到最高位（7位）。所有的十六进制都在前面加上一个美圆符号（$）（$2002,$4026,等等）是在 6502处理器汇编里常用的符号，二进制前面加上一个百分号%。

2. 缩写表

NES	任天堂娱乐系统

Famicom	任天堂家用计算机，即 FC
FDS	任天堂磁碟机系统
CPU	中央处理器，NES使用一个定制的 6502（NMOS）芯片，有些型号为 6527
PPU	图像处理器，用来处理背景，精灵和其他图像特性，通常为 6538
APU	声音处理器，集合在 CPU内部，包含 4个模拟通道和 1个数字通道
MMC	ROM和 VROM的扩容控制，用来控制访问超过 6502限制的 64K地址，同样，也可以扩容 VROM
VRAM	图像 RAM，PPU专用，2K字节
VROM	图像 ROM，储存图像数据的地方，可以由 MMC切换到 VRAM里
ROM	程序 ROM，实际程序储存的地方，扩容部分可以通过 MMC切换到 PRG－RAM里
RAM	程序 RAM，和 ROM同义，不同的是它是 RAM
SPR－RAM	精灵 RAM，RAM中的 256字节，专用于储存精灵，它不属于 VRAM或 ROM
SRAM	电池 RAM，卡带上用来保存游戏记录的 EPROM－电擦写 ROM
DMC	三角波调制通道，APU用来处理数字声音的，也写作 PCM通道
EX－RAM	扩展 VRAM，用在 MMC5里，可以扩展 VRAM容量

3. 中央处理器

NES定制的 6502内部特别加上了声音处理单元。NTSC制式的 NES使用 1.7897725MHz主频，PAL制式使用 1.773447MHz主频。

CPU内存映像：

开始地址	用途	结束地址
$0000	2K字节 RAM，做 4次镜象（即 $0000-$07FF可用）	$1FFF
$2000	寄存器	$2007
$2008	寄存器（$2000-$2008的镜像，每 8个字节镜像一次）	$3FFF
$4000	寄存器	$401F
$4020	扩展 ROM	$5FFF
$6000	卡带的 SRAM（需要有电池支持）	$7FFF
$8000	卡带的下层 ROM	$BFFF
$C000	卡带的上层 ROM	$FFFF

中断：

6502有 3个中断 IRQ/BRK、NMI和 RESET，每个中断都有一个 16位的向量，即指针，用来存放该中断发生时中断服务函数的地址。中断发生时 CPU都会把状态标志和返回地址压栈，然后调用中断服务程序。

IRQ/BRK中断由一下两种情况产生：一是软件通过 BRK指令产生，一是硬件通过 IRQ引脚产生。
RESET在开机的时候触发，这是 ROM被装入， 6502跳到 RESET向量指向的地址没有寄存器被修改，没有内存被清空，这些都只在开机是发生。
NMI指不可屏蔽中断，它在 VBlank即屏幕刷新时发生，持续时间根据系统（NTSC/PAL）不同而不同。 NTSC是每秒 60次，而 PAL是每秒 50次。 6502的中断延时是 7个时钟周期，也就是说，进入和离开中断都需要 7个时钟周期。它产生于 PPU的每一帧结束， NMI中断可以由 $2000的第 7位的 1/0控制允许/禁止。
大部分中断应该使用 RTI指令返回，但是有些游戏不用，例如《最终幻想 1》。它用一个很奇怪的方式：手工修改堆栈指针，然后执行 RTS指令。这种方法在技术上是可行的，但是应该尽量避免。
以上中断在 ROM内有以下对应的地址：

中断地址	中断	优先权
$FFFA	NMI	中
$FFFC	RESET	高
$FFFE	IRQ/BRK	低

特别说明：
NES的 6502不支持 10进制。虽然 CLD和 SED指令都正常工作，但是 ADC和 SBC都不使用 CPU状态标志的“ D”位。由于复位后“ D”位的状态是不确定的，所以游戏通常在程序开始时使用一个 CLD指令。
声音寄存器映射到 CPU内部，所有波形发生的工作都在 CPU内部完成。
注意那两个分开的 16K ROM段，它们可能是连续的，但是它们根据卡带的大小扮演不同的角色。有的卡带只有一个 16K ROM，那么它就同时被装入 $8000和 $C000。
所有游戏都将它们自己装入 $8000，使用 32K RAM，但是它们都能够通过内存映射把多于一个 16K ROM装入 $8000。 VROM也是同样的道理。
当 BRK中断发生的时候， CPU把状态标志压入堆栈，同时设置“ B”标志。而 IRQ中断发生时， CPU把状态标志压入堆栈，同时清除“ B”标志。这是因为 6502使用同一个向量来处理两种中断，用“ B”标志来区分它们。你可以用以下程序来区别两种中断：

```
C134: PLA ;                 拷贝 CPU状态标志到 A
C135: PHA ;                 把状态标志还回给堆栈
C136: AND #$10 ;            检查“ B”标志
C138: BNE is_BRK_opcode;    如果设置了，就是软件中断 (BRK)
```

在 NMI里指向 BRK会导致已经被压栈的“ B”标志被设置。
6502的 $6C指令（间接绝对跳转）有一个 BUG，当低位字节是 $FF时 CPU将不能正确计算有效地址。例如：

```
C100: 4F
C1FF: 00
C200: 23
```

..
D000: 6C FF C1 - JMP ($C1FF)
本来它是应该跳到 $2300的，但是在计算高位字节的时候，在页面边界处地址是不能再增加的，所以实际将跳转到 $4F00。
需要注意的是，页面越界不会在变址间接寻址模式发生。由于 0页面的限制，由于 0页面的限制，所有变址间接寻址的读写都应该在计算有效地址之后和 #$FF进行逻辑与操作。例如：

```
C000: LDX #3 ;          从 $0002+$0003 读变址地址，
C002: LDA ($FF,X) ;     不是 $0102+$0103.
```

---

4. 图形处理器

PPU时序：

	NTSC制式	PAL制式
基频（Base clock）	21477270.0Hz	21281364.0Hz
CPU主频（Cpu clock）	1789772.5Hz	1773447.0Hz
总扫描线数（Total scanlines）	262	312
扫描线总周期（Scanline total cycles）	1364(15.75KHz)	1362(15.625KHz)
水平扫描周期（H-Draw cycles）	1024	1024
水平空白周期（H-Blank cycles）	340	338
结束周期（End cycles）	4	2
帧周期（Frame cycles）	1364*262	1362*312
帧 IRQ周期（FrameIRQ cycles）	29830	35469
帧率（Frame rate）	60(59.94Hz)	50Hz
帧时间（Frame period）	1000.0/60.0(ms)	1000.0/50.0(ms)

镜像是指通过硬件映射特殊的内存地址或范围的一个过程。
PPU内存映像：

开始地址	用途	结束地址
$0000	图案表 0（256x2x8，可能是 VROM）	$0FFF
$1000	图案表 1（256x2x8，可能是 VROM）	$1FFF
$2000	命名表 0（32x30块）（镜像，见命名表镜像）	$23BF
$23C0	属性表 0（镜像，见命名表镜像）	$23FF
$2400	命名表 1（32x30块）（镜像，见命名表镜像）	$27BF
$27C0	属性表 1（镜像，见命名表镜像）	$27FF

$2800	命名表 2（32x30块）（镜像，见命名表镜像）	$2BBF
$2BC0	属性表 2（镜像，见命名表镜像）	$2BFF
$2C00	命名表 3（32x30块）（镜像，见命名表镜像）	$2FBF
$2FC0	属性表 3（镜像，见命名表镜像）	$2FFF
$3000	$2000-$2EFF的镜像	$3EFF
$3F00	背景调色板 #1	$3F0F
$3F10	精灵调色板 #1	$3F1F
$3F20	镜像，（见调色板镜像）	$3FFF
$4000	$0000-$3FFF的镜像	$7FFF

命名表：

NES的图像通过 Tile矩阵来显示，这个网格就叫命名表。一个命名表和字符模式下的屏幕缓冲比较相象，它包含字符的代码，也就是 30列的 32Byte长度。每个 Tile有 8x8个象素，每个命名表有 32x30个 Tile，也就是 256x240象素。 PPU支持 4个命名表，他们在 $2000,$2400,$2800,$2C00。在 NTSC制式下，上面和下面的 8象素通常不显示出来，只有 256x224象素；在 PAL制式下，屏幕有 256x240象素。

需要说的是，虽然 PPU支持 4个命名表，任天堂主机只支持 2个命名表。另外两个被做了镜像。命名表保存了 Tile的编号，而 Tile存在图案表里。计算命名表里 Tile号对应的实际地址的公式是：

（Tile号×16）＋由 $2000端口指定的图案表地址

命名表镜像：

NES只有 2048字节（$800）的 VRAM给命名表使用，但是如前表所示， NES有能力寻址到 4个命名表。缺省情况下， NES卡带都带有水平和垂直镜像，允许你改变命名表指向 PPU的 VRAM位置。这种方式同时影响两个命名表，你不能单独改变其中的一个。每个卡带都控制着 PPU地址线的 A10和 A11。它可能将他们设置成以下 4种可能的方式的 1种。下面这个图表有助于理解 NES里的各种镜像，指向 PPU VRAM中命名表的 12位地址相当于

“ $2xxxx”：

名字	命名表 #0	命名表 #1	命名表 #2	命名表 #3	说明	地址线 A11	地址线 A10
水平	$000	$000	$400	$400		1	0
垂直	$000	$800	$000	$800		0	1
4屏幕镜像	$000	$400	$800	$C00	卡带里有 2K VRAM，4个命名表物理上独立的	1	1
单屏幕	$X00	$X00	$X00	$X00	所有的命名表指相同的 VRAM区域，X=0、4、8、C	0	0
VROM镜像	Mapper 68#游戏映射 VROM到 PPU VRAM的命名表，这使得命名表是基于 VROM的，你不能写它但却可以通过						

	mapper自己来控制是否使用这种特性		

图案表：

图案表储存了实际 8x8象素的 Tile，同时也储存了用来指向 NES调色板全部 16种颜色的 4位元矩阵的低两位。 PPU支持两个图案表在 $0000和 $1000。他们有以下格式：

VRAM地址	图案表内容		颜色效果
$0000 .. .. $0007	%00010000 = $10 %00000000 = $00 %01000100 = $44 %00000000 = $00 %11111110 = $FE %00000000 = $00 %10000010 = $82 %00000000 = $00	组 0	...1.... ..2.2... .3...3.. 2.....2. 1111111. 2.....2. 3.....3.
$0008 .. .. $000F	%00000000 = $00 %00101000 = $28 %01000100 = $44 %10000010 = $82 %00000000 = $00 %10000010 = $82 %10000010 = $82 %00000000 = $00	组 1	点表示 0号颜色，数字表示实际调色板颜色代号

注意在图案表里存储的是每个点的 2个位。其他两个由属性表得到。所以，在屏幕上总体出现的颜色数是 16，而每个块里只有 4种颜色。

属性表：

每个命名表有它自己的属性表。属性表的每一个字节代表了屏幕上的一组 4x4的 Tile，一共有 8x8个字节。有几种方法来描述属性表里一个字节的功能：

- 保存 32x32象素方格的高 2位颜色，每 16x16象素用 2位；
- 保存 16个 8x8 Tile的高 2位颜色；
- 保存 4个 4x4 Tile格子的高 2位颜色。

看以下两个图表帮助理解：

1、一个 16x16 象素的格子：#0-F 代表了一个 8x8 Tile,方块 [x] 代表了 4个 8x8 Tile，

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

2. 属性表一个字节的实际格式定义如下（对应于上面的例子）：

位	描述
0, 1	方块 0的高两位颜色 (Tile #0,1,2,3)
2, 3	方块 1的高两位颜色 (Tile #4,5,6,7)
4, 5	方块 2的高两位颜色 (Tile #8,9,A,B)
6, 7	方块 3的高两位颜色 (Tile #C,D,E,F)

调色板：

NES有两个调色板，背景调色板和精灵调色板。调色板不包含实际的 RGB值，它们更象一个索引表。写到 $3F00-$3FFF的 D6– D7字节被忽略。

30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F

调色板镜像：

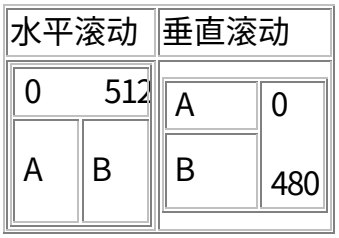
镜像发生在背景调色板和精灵调色板之间，例如所有写到 $3F00的数据会被镜像到 $3F10, $3F04镜像到 $3F14. 背景和精灵的最高 3个调色板的 0号色盘定义为透明，存在那里的颜色不会被画出来。 PPU使用放在 $3F00里的颜色作为背景色，详细如下：

*$0D被写到 $3F00 (镜像到 $3F10)；
*$03被写到 $3F08 (镜像到 $3F18)；
*$1A被写到 $3F18；
*$3F08被读到累加器。

PPU使用 $0D作为背景颜色，尽管 $3F08有一个颜色 $03（因为 0号颜色在所有的调色板里都定义为透明）。最后，累加器上有一个值 $1A, 这是从 $3F18映像过来的。又一次，这个 $1A值没有被画出，因为所有的调色板的 0号颜色被定义为透明。整个背景和精灵调色板同时也映像到 VRAM的其他区域， $3F20-$3FFF全部都是这两个调色板分别的映像。写到 $3F00-$3FFF的 D6– D7字节被忽略。

背景滚动：

NES可以通过预提取命名表，图案表和属性表来使背景滚动，背景是独立于精灵而位于最下层的。可以水平和垂直滚动。



命名表 A通过 $2000的 D1-D0指定， B是跟在后面的一个命名表，根据镜像不同 B是动态的。这个不能工作在水平和垂直同时滚动的游戏里。背景会跨越多个命名表，如下所示：

在 $2005里写到水平滚动的值可以从 0— 256，写到垂直滚动的值从 0— 239， 239是考虑了负值的结果，例如 248代表— 8。

屏幕和精灵分层：

下面是 NES画图的循序：

CI代表颜色亮度，相当于 $2001的 D7-D5；BG是背景；EXT是扩展口的图像信号。 BGPRI代表 VRAM里背景‘优先权’位，每个精灵都有的，即第二字节的 D5位。 OBJ数代表了实际精灵的号码，不是 Tile索引值。前台高于任何其他层，最后被画上，后台低于任何其他层，最先被画上。

精灵和精灵 RAM:

NES用一个页面（256字节）来存放动画，每个精灵 4个字节，一共可以有 64个动画 /精灵，它们可以是 8x8或 8x16象素。动画 /精灵图案被存储在 VRAM的图案表其中一个里面。精灵属性，例如翻转和优先权被储存在一个特殊的 256字节的精灵 RAM，它不是 CPU或 PPU的地址的一部分。整个精灵 RAM可以通过 $4014的 DMA方式来写，写一个 8位的数到 $4014就将这个 8位数所指定的内存页面整个拷贝到精灵 RAM上。也可以通过把开始地址放在 $2003然后读 /写于 $2004（每次存取地址自动加一），它是一个一个字节存取的。动画 /精灵的属

性格式是：
动画 /精灵属性 RAM:
｜精灵 #0 ｜精灵 #1 ｜ ... ｜精灵 #62 ｜精灵 #63 ｜

字节	位	描述
0	YYYYYYYY	精灵左上角的 Y坐标－ 1
1	IIIIIIII	Tile索引号
2	vhp000cc	v=垂直翻转 (1=翻转)
		h=水平翻转 (1=翻转)
		p=背景优先权 （ 0=前台 1=后台）
		c=颜色的高 2位
3	XXXXXXXX	精灵左上角的 X坐标

Tile索引号就和命名表里的一样。精灵图案可以象 Tile图案对于背景图片一样抓取。唯一的不同是在 8x16的精灵时，上半部分（偶数号码）的 Tile索引由精灵图案表的 $0000开始，而下半部分（奇数号码）由 $1000开始。 $2000寄存器对 8x16精灵无效。全部 64个精灵有自己内部的优先权， 0号最高（最后被画）， 63号最低（最先被画）。每条扫描线只能显示 8个精灵，每个精灵 RAM条目都会被检查是否处于其他精灵的水平范围内。注意，这是基于扫描线的，不是基于精灵的，也就是说，会进行 256次检查，而不是 256/8或 256/16次。在一个真实的 NES芯片里，如果精灵被禁止（ $2001的 D4是 0）很长一段时间，精灵数据会渐渐消失。可以理解为精灵 RAM是一个 DRAM， D4控制了 DRAM的周期刷新信号。

碰撞标志：
碰撞标志是 PPU状态寄存器 ($2002)的第 6位（ D6） 。 PPU能够找出 0号精灵的位置，然后设置 D6，它是这样工作的： PPU扫描出背景图案象素和精灵象素同时不是透明的第一个地方。比如，屏幕的背景是一个非透明颜色 (颜色号 >0)， 0号精灵的坐标是（ 12， 34） ，它只是在第 4行的开始才有象素，那么，碰撞标志在屏幕刷新到（ 12， 37） 时才被设置。要记住， 0号颜色被定义为透明。引起 D6被设置的象素必须是已经被画出来的。下面的例子能帮助理解，这是两个 Tile，下划线表示透明的 0号颜色， *号表示碰撞标志被设置。

精灵		背景		结果	
__1111__ _111111_ 11222211 112__211 112__211 11222211 _111111_ __1111__	+	________ _____2__ _____21_ ____211_ ____2111 ___21111 __211111 _2111111	=	__1111__ _1111112_ 11222211 112__*11 112_2211 11222211 _1111111 _2111111	‘ *’会用 2号颜色画出

这个例子说的是背景＋精灵，但是对于靠近背景的精灵也通用，即精灵＋精灵，通过设置背景优先权来实现。 D6不可以通过读 PPU的状态来复位，只在每次 VBlank之后被清零。 Hit标志可以用在水平或者垂直屏幕分割的时候，还有许多好玩的效果。

水平和垂直消隐：

所有的游戏机都有一个刷新动作，用来重新定位电子枪显示可见的数据。最通用的显示设备是电视机，它分为每秒刷新 60次的 NTSC制式和 50次的 PAL制式。电子枪从左到右画出象素，它每次只能画一条扫描线，画下一条之前要先回到左面并且做好准备，这之间有一段时间叫做水平消隐（HBlank）。在画完全部 256条扫描线之后它又回到屏幕左上角准备下一次画屏幕（帧），这之间的一段时间就是垂直消隐（VBlank）。电子枪就是一个不断的走‘之’字形的过程。 VBlank标志就是 $2002的 D7，它表明 PPU是否在 VBlank期间，当 VBlank标志存在时，你就可以通过 $2006和 $2007访问 PPU内存。一个程序可以通过读 $2002来使 D7复位。在屏幕刷新期间，我们不能访问 PPU，而 PPU会在 CPU背后修改 VRAM指针，这样我们很容易在写入 VRAM时出错，为了让 PPU停下来，可以对 $2000和 $2001写 00。

访问 PPU RAM:

在一个任天堂主机，访问 PPU内存只可以在 VBlank期间。当在屏幕刷新时访问会破坏刷新地址寄存器，一般它经常用来做隐含的“分割屏幕”效果（见“在屏幕刷新的时候访问 VRAM”）。很多小些的 ROM用只读存储体 (VROM)用做图案表。在这种情况下，你不可以写 PPU地址，只可以读。

写 PPU记忆体：

a) 写高位地址字节到 $2006

b) 写低位地址字节到 $2006

c) 写数据到 $2007。每一个写操作后，地址会增加 1($2000的第二位是 0)或增加 32($2000的第二位是 1)。

读 PPU记忆体：

a) 写高位地址字节到 $2006

b) 写低位地址字节到 $2006

c) 从 $2007读数据。从 $2007读到的第一个字节是无效的 (见下例)。然后每读一次地址就增加一 ($2000的第二位是 0)或增加
32($2000的第二位是 1)。

例子： VRAM的 $2000里有 $AA $BB $CC $DD。 VRAM的增量是 1。执行效果如下：

```
LDA #$20
STA $2006
LDA #$00
STA $2006 ; VRAM 地址设为 $2000
LDA $2007 ; A=?? VRAM 缓冲 =$AA
LDA $2007 ; A=$AA VRAM 缓冲 =$BB
LDA $2007 ; A=$BB VRAM 缓冲 =$CC
LDA #$20
STA $2006
LDA #$00
```

```
STA $2006 ; VRAM地址设为 $2000
LDA $2007 ; A=$CC VRAM缓冲=$AA
LDA $2007 ; A=$AA VRAM缓冲=$BB
```

这个只适用于$0000-$FEFF，访问调色板数据没有这种现象。

在屏幕刷新的时候访问VRAM:

前面说过，在屏幕刷新的时候存取VRAM的地址和数据是不合法的。许多程序存取这些寄存器来制造不同的滚动效果。比如，一些游戏从屏幕底部开始滚动，那么它可能向$2006写第一行的状态来复位屏幕滚动。更好的诡计是PPU在屏幕刷新时用VRAM的地址寄存器来储存当前地址。通过向$2006修改地址以及让PPU从一个不同的地方接着刷新。关于$2007如何影响屏幕刷新仍不了解。

当不知道向$2006中写什么数据时，看下面的图表。

写到$2006的地址

位	描述
0—4	Tiles的水平偏移量(例如 1 = 8象素)；水平位置＝水平偏移量×8
5—9	Tiles的垂直偏移量(例如 1 = 8象素)；垂直位置＝垂直偏移量×8
A B	命名表的号码($2000,$2400,$2800,$2C00)；
C、D	附加的垂直偏移量，单位：象素(0..3)；扫描线＝垂直偏移量＋附加的垂直偏移量
E、F	00

5、声音处理器

NES的集成CPU里包含了一个4声道的声音处理器（2A03），为播放音乐提供准模拟的声音。4个声道分别是：2个方波，一个三角波和一个噪声发生通道，每个声道的特性都不一样。

声道：

方波声道能够产生54.6Hz到12.4KHz的方波，它的主要特性是频率扫略功能和调整输出的占空周期；三角波声道可产生27.3Hz到55.9KHz分辨率是4位（16级）的三角波，它的主要特性是输出模拟三角波和线性计数器（可用来定时关闭声道）；噪声通道能产生29.3Hz到44.7KHz的随机噪声，它的主要特性是产生伪随机数，在声道输出随机频率。

帧计数器：

2A03有一个内含的帧计数器，它能够产生60Hz（1/1帧率）、120Hz（1/2帧

率）和 240Hz（1/4帧率）的信号，提供给其他一些声音硬件使用。1/4帧率的信号是由 CPU频率乘以 2即 3579545Hz再除以 14915得到的。

声音硬件延迟：
在 2A03复位之后，到各个声音硬件开始接收时钟信号之间有 2048个时钟周期的延迟。当这 2K个时钟周期之后，声音开始正常工作，输出使能有效。这种现象只发生在系统复位之后的 2048个时钟周期内。这个资料的准确性有待验证。

控制寄存器：
2A03的声音控制寄存器在 CPU内分配的特殊地址如下：

寄存器地址	描述
$4000-$400	方波 1
$4004-$400	方波 2
$4008-$400	三角波
$400C-$400	噪声
$4015	声道使能、音长计数状态

只有 $4015是可读可写的，其他都是只能够写的，如果你读这些寄存器最可能得到的是 $40（因为 NES数据线的重负荷）。读这些只写寄存器对寄存器本身和声道都没有任何影响。每个声音通道都有 4个寄存器，具体描述如下：

寄存器 \位	0	1	2	3	4	5	6	7
$4000/$4004/$400C	音量衰减率				无衰减	音长计数中止 /衰减循环使能		占空周期类型
$4001/$4005	右移量			1减少 /0增加波长	扫描更新率			扫描使能
$4002/$4006/$400A	波长低八位							
$4003/$4007/$400B/$400	波长高三位			音长计数值				
$4008	线性计数值							音长计数中止 /线性计数开始
$4009/$400D	未使用							
$400E	播放采样率				未使用			随机数类型发生
$4015:音长状态（读）	方1	方2	三角	噪声	DMC	未使用		DMC的 IRQ状态
$4015:通道使能（写）	方1	方2	三角	噪声	DMC	未使用		

声道结构体系：
总表：

设备	三角波	噪声	方波
三角阶梯发生器	√		
线性计数器	√		
可编程定时器	√	√	√
音长计数器	√	√	√
4位 DAC	√	√	√
音量衰减单元		√	√
扫频单元			√
占空周期发生器			√
波长转换器		√	
随机数发生器		√	

三角阶梯发生器：
这是一个 5位的单向的计数器，只存在于三角波通道，计数器输出的低 4位分别通到一个 XNOR（异或非）门。 4个 XNOR门被同一个选通信号控制，当选通是 0时，计数器输出的 4个低位在门电路的输出端反相；当选通是 1时，不反相。选通信号连到计数器的最高位，这样就可以在 XNOR门的输出端产生一个计数序列，就造出了一个比较理想的三角阶梯信号发生器。 XNOR门的输出反馈到一个 4位的 DAC输入端。当三角波的音长计数器或线性计数器是 0时，这个 5位的计算器会暂停，同时锁定，计数器不会被重设置到某一个特定的状态。在系统复位的时候，计数器装入 0。计数器的时钟输入直接连接到三角波通道内一个 11位的可编程定时器的输出端，这样，这个 5位的三角波阶梯信号发生器的输出频率就是通道内可编程定时器频率的 1/32。

线性计数器：（$4008）
这是一个 7位的递减计数器，只存在于三角波通道，计数器减到 0时输出。计数器的时钟是 240Hz（1/4帧率），用帧来计算长度就是 0.25× N，其中 N是预先装入的 7位的值。计数器会保持计数状态直到输出 0，这时线性计数器和三角阶梯发生器的时钟信号被禁止，导致两个计数器被锁定，线性计数器的输出保持是 0，声道因此而出现静音。线性计算器有两种工作模式：装入和计数。在装入模式，所有的输入（写到 $4008）的值都会在传送到输出，也就没有计数；在计数模式，计数器会从当前值向下减，在 $4008的数会被忽略。下表表示了如何通过 $4008的最高位设置计数器：

旧值	新值	模式
0	X	计数
1	0	在写周期没有变化，计数
1	1	没有变化

可编程定时器（$4002/3、$4006/7、$400A/B、$400F）：
这是一个 11位可预设的递减计数器，每个声音通道都有，但是注意在噪声通道不是直接可用的，详见“波长转换器”。这个计数器在计数完成的时候自动重载，因此它在减到 0前会计算 N+ 1个时钟周期，其中 N是预装入的 11位值。这个计数器直接使用 6502的频率（1.79MHz）作为时钟，输出频率是 1.79MHz/(N+ 1)。当到达计数终点的时候会产生一个 1时钟周期的脉冲，而输出频率会被反馈到对应于声音通道的特殊设备。对于方波，这个计数终点脉冲会被送到占空周期发生器；对于三角波，会被送到三角阶梯发生器；对于噪声，会被送到随机数发生器。

音长计数器（$4003、$4007、$400B、$400F）：
这是一个 7位的递减计数器，时钟是 60Hz，每个声音通道都有。当计数器减到 0，计数停止，相应的通道出现静音。可以通过设置音长计数时钟的禁止位来暂停计数，写 1则中止计数，0则允许计数。在不同的寄存器里，这个位的位置不同，见寄存器表。计数值通过 $4003、$4007、$400B和 $400F的 3– 7位来写入。输入的 5位值通过下面 2个表转换到计数器的 7位值：

第 3位＝ 0		
4– 6位的值	帧	
	第 7位＝ 0	第 7位＝ 1
0	05	06
1	0A	0C
2	14	18
3	28	30
4	50	60
5	1E	24
6	07	08
7	0E	10

第 3位＝ 1	
4– 7位	帧
0	7F
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
A	0A
B	0B

C	0C
D	0C
E	0E
F	0F

音长计数器的状态可以通过 $4015来读取， 0表示计数器处于 0状态（停止在 0） ， 1表示处于非 0状态。通过写 0到 $4015可以令音长计数器永远处于 0状态，使得声道被禁止；写 1可以取消禁止状态，但它不会改变计数器里面的值。在系统复位时这个寄存器全部 5个使用的位都被设置为 0。

4位 DAC：
这是一个标准的 4位 DAC（数模转换）单元，有 16级输出电压， 4个声道共用。在 2A03里，方波 1和方波 2混合通过引脚 1输入，三角波和噪音通过引脚 2输入。这些模拟量输出需要一个负的电流源，以使得电压的级输出变成平滑的声音信号。因为 NES只外置了一个 100欧姆的下拉电阻，使得输出波形的幅值很小，但是也获得最小的不平滑度。这个不平滑度小到难以发现，就算用示波器也不明显。

音量衰减单元（ $4000、 $4004、 $4000） ：
音量衰减器只存在于方波和噪声通道。当衰减禁止位（第 4位）被设置为 1，当前的音量值（ 0— 3位）被直接送到 DAC。在下列情况中，音量值会被忽略，而直接送 0到 DAC。
· 声道硬件要停止声音，例如音长计数器或扫频单元；
· 处于占空周期或随机数发生器输出频率信号的负值部分，也即在声音使能的时候送到 DAC的或者是音量值或者是 0（无声）。
当衰减禁止位被清零， 0— 3位当前的值就是衰减率。衰减率是指音量从最大值向下线性衰减的速度，衰减速度是 240/(N+1)， N的值从 $0— $F。当衰减到 0之后，根据寄存器第 5位的状态将发生两种情况：

第 5位	行为
0	音量停在 0，声道静音
1	音量循环不断从最大值减到 0

只有通过对寄存器进行写操作才能重新设置硬件的衰减状态，否则就一直按当前状态进行衰减甚至循环，即使寄存器的第 4位被设置成禁止衰减也没有用。

扫频单元：
这个东西只存在在方波通道中。当寄存器的第 7位被设为 1，扫频被激活，波长会实时地增加或减少（对应频率降低或提高）。波长的值通过 $4002/3和 $4006/7被不断的读入和更新，写到那里的值会立刻反应到声音里。第 4— 6位代表了频率扫略速度，或者说 $4002/3和 $4006/7里波长值被改变的频率。刷新率是 120Hz/(N+1)， N是写入的值，从 0— 7。第 3位控制扫频的模式， 1表示波长减少， 0表示波长增加。第 0— 2位是右移量，用来控制波长增量的右移量，见下表：

第 3位	表达式，N是右移量从 0— 7
0	波长＝波长＋（波长 >>N)
1	波长＝波长－（波长 >>N)

当满足以下 3个条件，波长寄存器每个周期都会被更新：
· 第 7位被设置，即扫频使能；
· 右移量不等于 0;
· 音量计数器的值非 0。
以下两种情况会暂停扫描时钟，使声道静音，而不管扫描刷新率是多少或扫描使能位是否设置：
· 11位的波长值小于 $008;
· 扫频处于增加模式，而进位标志＝ 1。硬件在波长寄存器被更新之前总要检查进位标志的。

占空周期发生器（ $4000、 $4004) ：
占空周期发生器只存在于方波通道，它使用可编程定时器的输出频率作为时钟，使用 4位计数器来产生 4种类型的占空周期，因此它的输出频率是可编程定时器的 1/16，输出信号反馈到音量衰减器。占空类型见下表：

6、 7位	一个时钟周期里的占空周期（正 /负）
00	2/14
01	4/12
10	8/8
11	12/4

波长转换器（ $400B) ：
这个东西只存在于噪声通道，用来将一个给定的 4位值转换成 11位的波长，然后送到可编程定时器。下表是转换关系：

0— 3位	音度	音阶	CPU时钟周期（ 11位波长＋ 1)
0	15	A	002
1	14	A	004
2	13	A	008
3	12	A	010
4	11	A	020
5	11	D	030
6	10	A	040
7	10	F	050
8	10	C	065
9	9	A	07F
A	9	D	0BE
B	8	A	0FE

C	8	D	17D
D	7	A	1FC
E	6	A	3F9
F	5	A	7F2

随机数发生器（$400E）：
噪声通道有一个 1位的伪随机数发生器，它是由一个 15位的移位寄存器和一个异或门组成，它能够提供‘长’和‘短’两种类型的伪随机数序列：当寄存器第 7位是 0时，产生 32767位的长序列（32K模式）；是 1时，产生 93位的短序列（93位模式）。93位模式通常在声道中产生较高频率的声音回放。下表说明了在移位寄存器的那些位里进行异或：

模式	EDCBA9876543210 <----------<
32K	**
93位	* *

当前的异或结果在下一个移位周期之前送到移位寄存器的 0位，1个位的随机数从 E引脚输出，反相然后送到音量衰减器。移位寄存器每接收到两个可编程定时器的时钟脉冲就移位一次，所以它的频率是可编程定时器的一半，也就是低一个音度。系统复位的时候移位寄存器被装入值 1。

6. 手柄、控制器和扩展口

NES支持很多种输入设备，主要有手柄，光枪、4适配器和摇杆。两个手柄分别通过 $4016和 $4017访问，用选通脉冲对手柄复位，即对 $4016或 $4017先写 1后写 0，这将在游戏手柄的电路里发生个脉冲，然后可以在 $4016或 $4017读取了。每次读取将返回一个单一的按钮的状态，在第 0位（1表示按下，0表没有）。第 1位表示了游戏棒是否和端口连接。如果连接就置 0，否则是 1。第 6和 7位的 $4016/$4017好像有相同的重要性，不知是什么，有些游戏期望当一个按钮按下的时候在 $4016/$4017得到 $41。复位的时候返回 0。

读取#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
按键	A键	B键	选择	开始	上	下	左	右	忽略								信号				0	0	0	0

信号用来检测是 4端口的接入状态：
%0000:没有连接设备
%0001:只有 $4016手柄
%0010:只有 $4017手柄
光枪：

只是简单地使用了 $4016或 $4017的 D0、 D3和 D4位，完全有可能同时接两支光枪。

4适配器：

将手柄扩展到 4个，支持 4手柄的游戏有： Tengen的《Gauntlet II》和 Nintendo的《RC Pro Am 2》。全部 4个控制器使用 $4016或 $4017的 D0位。对于 $4016端口，读 1— 8字节是手柄 1， 9— 16字节是手柄 3， $4017则对应手柄 2和 4。

读取	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
按键	1PA	1PB	1P选择	1P开始	1P上	1P下	1P左	1P右	2PA	2PB	2P选择	2P开始	2P上	2P下	2P左	2P右	信号				0	0	0	0

摇杆：

有的游戏使用摇杆，例如 Taito的《Arkanoid》。摇杆的位置通过 $4017的 D1来读取，第一个读的是 MSB，第 8个读的是 LSB，有效值的范围是 98— 242， 98代表了，摇杆完全是被逆时针转的。读到的数据要取反，例如读到%01101011，取反是%10010100，即 146。摇杆还带有一个按钮，通过 $4016的 D1来读取， 1代表按钮被按下。

扩展口：

NES都有一个扩展口，用来接手柄或双手柄。手柄初始化需要写 1和 0，如果初始化没有完成，或者循序错了，扩展口就可以用来通讯，例如：

```
LDA #%00000001
STA $4016
STA $4017        ; 开始扩展口的读模式
LDA #%00000011   ; 写%110到扩展口
STA $4016
```

确实有些卡带用这种方式来通讯。

任天堂主机内部输入/输出端口映射到 $2000-$2007和 $4000-$4017。一些端口的用处是未明的或未知的。任何有关信息将受到感激。

7、内存映射硬件

这里只提供一些硬件资料，软件方面请参考扩容资料。

Mapper其实就是硬件上的地址映射。用了切换 ROM和 VROM，更有的用于动态切换 VROM，动态切换 VROM通常有 2种办法：一是用 Latches，所谓 Latch就是切换器，当 PPU访问命名表遇到特定的数据时，自动切换去指定的 VROM，例如圣火徽章外传就用了这种方法，它的特定数据是 FE和 FD，就是对话框的两个边 ^_^；二是用 IRQ中断，算好 PPU的时序和 CPU的指令周期后设置 IRQ计数器，等待到特定的时候就触发 IRQ中断，调用中断服务切换 VROM，例如三国志 2—

霸王之大陆就是这样，有的 IRQ计数器是每个 CPU周期增加 1，有的是每条扫描线增加 1，视硬件电路而定。

图例：
R：连接到 ROM部分；
S：由 ROM、MMC和 NES共用；
N：连接到卡带接口；
W：连接到电池 RAM；
Pxxx：对应于 CPU的；
Vxxx：对应于 PPU的；
M2：时钟。

+5V	M2	PA13	PA14	P-CE	PD7	PD0	R/W	VA10	VA12	VA11	V10
24	23	22 N	21 N	20 N	19 S	18 S	17	16 N	15 N	14 S	13 S
)	MMC1										
1 R	2 R	3 R	4 R	5 R	6 W	7 R	8 R	9 R	10 R	11 R/W	12
PA14	PA15	PA16	PA17	P-CE	B-CE	VA12	VA13	VA14	VA15	VA16/B-CE	GND

RD387-001：

+5V	+5V	GND	VA10	VA15	VA12	VA14	VA12	VA13	VA16	VA8	VA5	VA9	VA4	VA11	VA3	VA7	VA2	VA10	VA1	V-CE
XX	40	39	38 N	37 R	36 R	35 R	34 N	33 R	32 R	31 S	30 S	29 S	28 S	27 S	26 S	25 S	24 S	23 S	22 S	21 S
)	MMC2																			
XX	01	02 N	03 N	04 R	05 R	06 S	07 R	08 R	09 R	10 S	11 S	12 S	13 S	14 S	15 N	16 N	17 S	18 S	19 S	20
GND	M2	PA14	PA13	PA15	PA14	PA12	PA13	PA16	P-CE	PD4	PD3	PD0	PD1	PD2	R/W	P-CE	V-RD	VA0	VA6	GND

		PD6	PA0	PD7	B-CE	M2	GND	+5V	P-CE	PA18	PA13	PA17		
		33 S	32 S	31 S	30 W	29	28	27	26 R	25 R	24 N	23 R		
PD0	34 S	MMC3											22 R	PA14
PD5	35 S												21 R	PA16
PD1	36 S												20 N	PA14
PD4	37 S												19 R	PA13
PD2	38 S												18 R	PA15
PD3	39 S												17 N	R/W
GND	40												16	?
B-CE	41 W												15	GND
B-WE	42 W												14 N	P-CE

PA11	43 P												13	IRQ
PA16	44 P												12 P	VA17
		1	2 R	3 N	4 N	5 N	6	7 R	8 R	9 R	10 N	11 P		
		?	VA10	VA12	VA11	VA10	GND	VA13	VA14	VA12	VA10	VA15		

MMC5:

引脚	功能	引脚	功能	引脚	功能	引脚	功能	引脚	功能
1	RC0	21 P	VA16	41 S	PD5	61 P	PA14	81	NC
2	RC1	22 P	VA17	42 S	PD6	62 P	PA15	82	NC
3	RC2	23 P	VA18	43 S	PD7	63 P	PA16	83 V	B-CE
4	+5V	24 P	VA19	44	+5V	64 P	PA17	84 S	VD0
5 S	VA0	25 N	VA10	45 S	PA0	65 P	PA18	85 S	VD1
6 S	VA1	26 N	VA11	46 S	PA1	66 P	PA19	86 S	VD2
7 S	VA2	27 N	VA12	47 S	PA2	67 N	PA13	87 S	VD3
8 S	VA3	28 N	VA13	48 S	PA3	68 N	PA14	88 S	VD4
9 S	VA4	29	NC	49 S	PA4	69 V	A13	89 S	VD5
10 S	VA5	30	NC	50 S	PA5	70 V	A14	90 S	VD6
11 S	VA6	31 N	V-CE	51 S	PA6	71 V	CE-(	91 S	VD7
12 S	VA7	32 N	VA10	52 S	PA7	72 V	CE-1	92	NC
13 S	VA8	33 N	V-WA	53 S	PA8	73	NC	93	NC
14 S	VA9	34 S	V-RD	54 S	PA9	74 P	P-CE	94 P	VA0
15 P	VA10	35 N	IRQ	55 S	PA10	75	NC	95 P	VA1
16 P	VA11	36 S	PD0	56 V	VCC	76 V	B-WE	96 P	VA2
17 P	VA12	37 S	PD1	57	+BAT	77	NC	97	CL3
18 P	VA13	38 S	PD2	58 S	PA11	78	NC	98	SL3
19 P	VA14	39 S	PD3	59 S	PA12	79 N	M2	99	GND
20 P	VA15	40 S	PD4	60 P	PA13	80	GND	100	RC3

MMC6:

引脚	功能	引脚	功能	引脚	功能	引脚	功能	引脚	功能	引脚	功能	引脚	功能
1 N	PA13	11	GND	21 P	VA13	31	IRQ	41 S	PD5	51 S	PA5	61 P	PA14
2	M2	12	+5V	22 P	VA12	32 N	P-CE	42 S	PA0	52 S	PA6	62 P	PA18
3	GND	13	HLD	23 P	VA14	33	R/W	43 S	PD6	53 S	PA9	63 N	PA14
4	GND	14 N	VA10	24	GND	34 S	PD2	44 S	PA1	54 S	PA7	64 P	PA17
5	GND	15 N	VA11	25	+BAT	35 S	PD3	45 S	PD7	55 S	PA8		
6	NC	16	NC	26 P	VA18	36 S	PD1	46 S	PA2	56	+BAT		
7	GND	17 P	VA10	27 N	VA10	37	GND	47 P	P-CE	57	GND		
8	+5V	18 P	VA16	28 S	V-RD	38	+BAT	48 S	PA3	58 S	PA12		
9	+5V	19 P	VA11	29 S	V-CE	39 S	PD4	49 S	PA4	59 P	PA13		

10	+BAT	20	N	VA12	30	F	VA17	40	S	PD0	50	F	PA16	60	F	PA15		

NES-CN-ROM-256-[02,05]:

+5V	NC	VA14	VA13	VA12	VA11	GND	R/W
16	15	14	F	13	F	12	11	10	09
)	74LS161						
01	02	03	04	05	06	07	08
+5V	P-CE	PD0	PD1	PD4	PD5	GND	GND

NES-AOROM-03 NES-BN-ROM-01:

+5V	NC	PA15	PA16	PA17	VA10	GND	R/W
16	15	14	13	12	11	10	09
)	74LS161						
01	02	03	04	05	06	07	08
+5V	P-CE	PD0	PD1	PD2	PD4	GND	GND

NES-CN-ROM-256-[02,05]:

+5V	NC	LS32-13	LS32-2	LS32-9	NC	GND	R/W
16	15	14	R	13	R	12	11	10	09
)	74LS161						
01	02	03	04	05	06	07	08
+5V	P-CE	PD0	PD1	PD2	GND	GND	GND

+5V	LS161-14	A14	A14	A14	LS161-12	A16
14	13	12	11	F	10	09	08	F
)	LS32					
01	02	03	F	04	05	06	07
A14	LS161-13	A15	GND	GND	NC	GND

NES-ANROM-03:

+5V	NC	PA15	PA16	NC	VA10	GND	R/W
16	15	14	13	12	11	10	09
)	74LS161						
01	02	03	04	05	06	07	08
+5V	P-CE	PD0	PD1	+5V	PD4	GND	GND

+5V	-8-9	-1	P-CE	CE	-13	-13
14	13	12	11	F	10	F	09	08
)	LS02					

01	02	03 F	04	05	06	07
-12	R/W	R/W	NC	GND	GND	GND

CPU:

+5V	Load	Q1	Q2	INP0	INP1	NMI	R/W	IRQ	M2	GND	CLK	D0	D1	D2	D3	D4	D5	D6	D7
40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21
)	6502/6527																		
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
Audio 1	Audio 2	RESET	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	GND

PPU:

+5V	ALE	AD0	AD1	AD2	AD3	AD4	AD5	AD6	AD7	A8	A9	A10	A11	A12	A13	RD	WR	RST	VIDEO OUT
40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21
)	6528/6538																		
01P	02P	03P	04P	05P	06P	07P	08P	09P	10P	11P	12P	13P	14P	15P	16P	17P	18P	19P	20
R/W	D0	D1	D2	D3	D4	D5	D6	D7	A2	A1	A0	CE	GND	GND	GND	GND	CLK	INT	GND

8. I/O端口

端口是预分配好的可访问地址，程序通过端口和 PPU或 APU交换信息。每个端口都是 16位的

端口	读写/位	功能描述
$2000	读写	PPU控制寄存器 1
	0-1	命名表地址： %10 ($2800) %11 ($2C00) %00 ($2000) %01 ($2400) 请记住，因为镜像，只有 2个真实的命名表，而不是 4个； 当滚动超出当前命名表时 PPU会自动切换到另一个命名表

	2	垂直写，1= VRAM地址以 32递增： 命名表，W= 0 命名表，W= 1 ——>写 | 写 | V
	3	精灵图案表地址，1= $1000，0= $0000
	4	屏幕图案表地址，1= $1000，0= $0000
	5	精灵尺寸，1= 8x16，0= 8x8
	6	PPU 主 /从模式，没有在 NES里使用
	7	Vblank使能，1=在 Vblank时发生中断
$2001	读写	PPU控制寄存器 2
	0	显示模式，0=彩色，1=单色
	1	背景掩码，0=不显示屏幕的左 8列
	2	精灵掩码，0=不在左 8列显示精灵
	3	屏幕使能，1=显示图像，0=黑屏
	4	精灵使能，1=显示精灵，0=隐藏精灵
	5-7	当 D0= 0：背景颜色，% 000=黑，%001=蓝，%010=绿，%100=红；不用混合使用，因为可能会损坏 PPU硬件 当 D0= 1：颜色亮度 ,% 000没效果，%00 1强化蓝色，%010强化绿色，%100强化红色，不要混合使用
$2002	读	PPU状态寄存器：读取后 $2005和 $2006被复位，下一个写到 $2005的数据是水平的，写到 $2006的数据是高位
	0-3	未知 (???)
	4	VRAM写标志：0=写有效，1=写忽略
	5	扫描线精灵计数：0=当前扫描线精灵个数小于等于 8个，1=大于 8个
	6	命中标志：1=精灵刷新碰到了 #0精灵。在屏幕刷新状态，这个标志被复位为 0
	7	Vblank标志：1= PPU在 Vblank状态。当 Vblank结束或 CPU读 $2002时，该标志被复位为 0
$2003	写	精灵 RAM地址：用来设置通过 $2004访问的 256字节精灵 RAM地址。每次访问 $2004后该地址增加 1
$2004	读写	精灵 RAM数据：用来读 /写精灵内存。地址通过 $2003来设置，每次访问后地址增加 1
$2005	写	屏幕滚动偏移：第一个写的值会进入垂直滚动寄存器（若 >239，被忽略）。第二个值出现在水平滚动寄存器
$2006	写	VRAM地址：设置 $2007访问的 VRAM地址。第一个写地址的高 6位。第二个写低 8位。每次访问 $2007后地址增加

$2007	读写	VRAM数据：用来访问 VRAM数据，通过 $2006设置的地址在每次访问之后会增加 1或 32
$4000	写	APU方波 #1控制端口
$4001	写	APU方波 #1音量控制端口
$4002	写	APU方波 #1细音调（FT）端口
$4003	写	APU方波 #1粗音调（CT）端口
$4004	写	APU方波 #2控制端口
$4005	写	APU方波 #2音量控制端口
$4006	写	APU方波 #2细音调（FT）端口
$4007	写	APU方波 #2粗音调（CT）端口
$4008	写	APU三角波控制端口 #1
$4009	?	APU三角波控制端口 #2
$400A	写	APU三角波频率端口 #1
$400B	写	APU三角波频率端口 #2
$400C	写	APU噪声控制端口 #1
$400D		未使用 (???)
$400E	写	APU噪声频率端口 #1
$400F	写	APU噪声频率端口 #2
$4010	写	APU增量调制控制端口
$4011	写	APU增量调制 D/A端口
$4012	写	APU增量调制地址端口
$4013	写	APU增量调制数据长度
$4014	写	DMA访问精灵 RAM：通过写一个值 xx到这个端口，引起 CPU内存地址为 $xx00－ $xxFF的区域传送到精灵内存
$4015	读写	声音通道切换
	0	方波 #1通道： 1＝声音使能，
	1	方波 #2通道： 1＝声音使能，
	2	三角波通道： 1＝声音使能，
	3	噪声通道： 1＝声音使能，
	4	增量调制： 1＝声音使能，
	5、7	未使用 (???)
	6(只读)	垂直时钟信号 IRQ可用： 0＝ 1帧存在，因为没有 IRQ；1＝ 1帧被 IRQ中断
$4016	读写	手柄 1＋选通
	0	读：手柄 1数据；写：选通， 1＝复位， 0＝清除；扩展端口： 0＝写， 1＝读

	1	手柄 1 存在， 0=连接
	3	光枪精灵检测： 1=瞄准
	4	光枪扳机： 0=按下， 1=松开
	2、5—7	未知， 6、 7设为%10(???)
$4017	读写	手柄 2+ 选通
	0	读：手柄 2数据；写：选通， 1=复位， 0=清除；扩展端口： 0=??, 1=读
	1	手柄 2 存在， 0=连接
	2、5	未使用 (???)
	3	光枪精灵检测： 1=瞄准
	4	光枪扳机： 0=按下， 1=松开
	6	垂直时钟信号（内部）， 0=存在， $4016的 D6受影响； 1=不存在 $4016的 D6不可触及
	7	垂直时钟信号（外部）， 0=存在， 1=不存在

9、NES文件格式

.NES文件为模拟用来储存 NES卡带的映像。下面是一个 .NES文件的结构。

偏移	字节数	内容
0—3	4	字符串“ NES^Z”用来识别 .NES文件
4	1	16kB ROM的数目
5	1	8kB VROM的数目
6	1	D0：1=垂直镜像， 0=水平镜像
		D1：1=有电池记忆， SRAM地址 $6000-$7FFF
		D2：1=在 $7000-$71FF有一个 512字节的 trainer
		D3：1= 4屏幕 VRAM布局
		D4—D7：ROM Mapper的低 4位
7	1	D0—D3：保留，必须是 0（准备作为副 Mapper号 ^_^）
		D4—D7：ROM Mapper的高 4位
8—F	8	保留，必须是 0
16-	16KxM	ROM段升序排列，如果存在 trainer，它的 512字节摆在 ROM段之前
-	8KxN	VROM段，升序排列

EOF		

10. 任天堂磁碟机系统

任天堂家庭机磁碟系统 (FDS)是它的扩展单位，它只是由任天堂公司生产以及只在亚洲国家销售。它有一个软盘驱动器可以插 2.5寸软盘，可以以 32KB的 RAM（代替了 ROM）来读进程序。 8KB的 VRAM（代替了 VROM），其他的硬件描述于下。

内存映像：
FDS的地址空间的描述：

$0000	2kB InternalRAM, mirrored 4 times
$2000	Input/Output
$5000	Expansion Modules
$6000	32kB Program RAM
$E000	8kB FDS BIOS ROM

FDS还增加了一些 I/O端口于 $4000-$40FF来控制磁盘系统，声音系统， IRQ记数器。

端口	读写 / 位	功能描述
$4020	W	Lower Byte of IRQ Counter
$4021	W	Upper Byte of IRQ Counter
$4022	W	Enable/Disable IRQs
	2	\ = Stop IRQ counter and reset its interrupt request / = Load IRQ counter with a value from $4020-$4021 and start it
$4024	W	Data Write Register:To write data to the disk, output it into this register
$4025	W	Control Register
	0	Drive Motor, 0 = on, 1 = off
	1	\ = Set drive head to the start of the first track
	2	Disk Write, 0 = enabled, 1 = disabled
	3	Screen Mirroring, 0 = vertical, 1 = horizontal
	4-6	Unknown (???)
	7	Disk IRQs, 0 = disabled, 1 = enabled
$4026	W	ExPort Output (???)

$4030	R	Disk Status Register 0
	4	Unknown (???)
	6	Unknown (???)
$4031	R	Data Read Register:To read data from the disk, input it from this register
$4032	R	Disk Status Register 1
	0	Disk Presence, 0 = inserted, 1 = not inserted
	1	Disk Ready, 0 = ready, 1 = not ready
	2	Write Protection, 0 = unprotected, 1 = protected
$4033	R	ExPort Input
	7	Battery Status, 0 = ok, 1 = low

可屏蔽中断计数器：

FDS提供了一个 16位的 IRQ记数器连接到 CPU时钟发生器。计数器开始于你写在 $4021的一个预设在 $4020-$4021的值。它在每个 CPU时钟周期后减一。当计数器为零时，它从预设的值开始继续计数。这时就发生了一个 IRQ。当垂直空白时，屏幕刷新时也会发生。 $4022的第 2个位用来控制 IRQ计数器。当它是 0，那么计数停止，要到达的 IRQ被复位（如果有的话）。设置了 1后，记数可以被恢复。

磁碟格式：

每个磁盘有两面， A和 B，每面可以有 65000字节的数据。要反转一个当前面，必须先拿出来，反转，重新插回去。数据在每个面上被储存于一系列的不同类型的块 [1,2,3,4,3,4,...,3,4]。

Block	Byte	Description
1. Side Header Block(56 bytes)	0	$01
	1-14	"*NINTENDO-HVC*"
	15	Maker ID
	16-19	Side Name
	20	Version Number
	21	Side Number, $00 = Side A, $01 = Side B
	22-24	Additional Disk Data
	25	$08
	26-56	Reserved Space (not used by BIOS)
2. File Number Block (2 bytes)	0	$02

	1	Number of Files on this side
3. File Header Block (16 bytes)	0	$03
	1-2	File Number (not used by BIOS?)
	3-10	File Name (not used by BIOS?)
	11-12	Starting Address in memory
	13-14	File Size
	15	File Type:$00= Program Data,$01 = Character Data,$02 = Screen Data
4. File Data Block (variable length)	0	$04
	1-...	Data (see File Header Block for size)

所以两字节地方都以最重要的字节开始。面的名称可能和相同游戏软盘的不同面不同，但它们是一样的。当文件被读取，开始的地址在 RAM和 VRAM都是实在的地址。

磁碟访问：
下面的图表显示了 FDS如何从软盘中读 /写。
READ:

$4025	A	B	C	D	E	A) Initialization
7bit	_	__	_	-	--_	B) Motor on
6bit	_	__	-	-	--_	C) Read start mark
5bit	-	--	-	-	---	D) Enable IRQs
4bit	_	__	_	_	_-_	E) Read end mark
2bit	-	--	-	-	---	1. Read data, ($4030)=[xx0xxxxx]
1bit	-	-_	_	_	___	2. Read data, ($4030)=[xxx0xxxx]
0bit	_	--	-	-	--_	
Note					1 2	

WRITE:

$4025	A	B	C	D	E	A) Initialization
7bit	_	__	__	-	--_	B) Motor on
6bit	_	__	_-	-	--_	C) Write start mark
5bit	-	--	--	-	---	D) Enable IRQs
4bit	_	__	__	_	_-_	E) Write end mark
2bit	-	--	__	_	__-	1. Delay, write [00000000]

1bit	-	-	_	_	_	2. Write [10000000]
0bit	_	-	-	-	-	3. Write data, ($4030)=[xx0xxxxx]
Note			1	2	34	4. Delay

对于读写是如何完成的并不清楚。虽然知道 FDS每当一个字节从软盘里读出的时候就发生一个 IRQ。 FDS的 IRQ处理器（是 FDS的 BIOS一部分）在 $4031读取这个字节，然后指针指向下个字节。

磁碟错误：
下面是软盘错误的信息：

ERR	Meaning	注释
01	No disk	没有软盘
02	No power	没有电源
03	Broken prong on the disk	??
04	Wrong maker ID	??
05	Wrong game name	不正确的游戏名字
06	Wrong game version	不正确的游戏版本
07	Wrong side number (flip the disk)	不正确的盘面（反转之）
08	Wrong disk #1	一号软盘错
09	Wrong disk #2	二号软盘错
10	Wrong disk #3	三号软盘错
11		
12		
13		
14		
15		
16		
17		
18		
19		
20	Allows it to recognize screen data differs (???)	??
21	Wrong Side Header Block ("*NINTENDO-HVC*")	不正确的盘面柱头块
22	Wrong Side Header Block ID ($01)	不正确的盘面柱头块号
23	Wrong File Number Block ID ($02)	不正确的文件块号
24	Wrong File Header Block ID ($03)	不正确的文件头块号
25	Wrong File Data Block ID ($04)	不正确的文件数据块号
26	Error writing data to the disk	写文件错

27	Block ends prematurely	块过早结束
28	The disk unit and the same period can't take it(???)	??
29	The disk unit and the same period can't take it (???)	??
30	Disk full	软盘满
31	Data number of a disk doesn't match up (???)	??