SNES Kart

The most complete guide to a SNES cartridge worldwide

None of the information contained in this text comes from any confidential source. It was obtained from various sources on the Internet, but also the product of my own investigation. Refer to the Acknowledgements section at the end of this text.

Use this information for your own use, I will not take any responsibility for your actions. All

I wrote this because all of this information is scattered in small files everywhere, if existing at all, most of it outdated. This is an attempt to conveniently bring all of the information to one place, and as up-to-date as possible. If you find this useful, tell me! I love positive feedback.

Contents

Pin Layouts

What is the cartridge pin layout?

If the SNES doesn't detect the CIC while power is on, then it will not continue to read the cartridge. Further details of this are not known to me.

```
                Super FX   01   32
                           02   33
                           03   34
                           04   35

                     GND   05   36   GND
F                    A11   06   37   A12
r                    A10   07   38   A13
o                     A9   08   39   A14
n                     A8   09   40   A15
t                     A7   10   41   BA0
                      A6   11   42   BA1
o                     A5   12   43   BA2
f                     A4   13   44   BA3
                      A3   14   45   BA4
c                     A2   15   46   BA5
a                     A1   16   47   BA6
r                     A0   17   48   BA7
t                   /IRQ   18   49   /CS
                      D0   19   50   D4
                      D1   20   51   D5
                      D2   21   52   D6
                      D3   22   53   D7
                     /RD   23   54   /WR
      CIC out data (p1)   24   55   CIC out data (p2)
       CIC in data (p7)   25   56   CIC in clock (p6)
                   RESET   26   57   nc
                     Vcc   27   58   Vcc

                           28   59
                           29   60
                           30   61
              Left audio   31   62   Right audio

LoROM:    32kbyte pages/banks (A15 not used - assumed high)
HiROM:    64kbyte pages/banks

BA0-BA7 switch between a possible 256 banks/pages.
```

LoROM data is stored in the upper 32kbytes of the possible 64kbyte bank/page (A15 is assumed high). Using 64kbyte pages, the SNES can address a huge 16Mbytes or 128Mbits!

According to a SNES memory map, LoROM games can be as large as 16Mbit while HiROM games are limited to 32Mbit... what about the 48Mbit game floating around?

What is the ROM pin layout?

This pin layout was taken from a Donkey Kong Country 2 cartridge and seems to be consistent with all their mask ROMs (some are 32pin, others 36pin).

```
               A20             Vcc
               A21             A22
```

```
A17   01   32   Vcc
A18   02   31   /OE
A15   03   30   A19
A12   04   29   A14
 A7   05   28   A13
 A6   06   27   A8
 A5   07   26   A9
 A4   08   25   A11
 A3   09   24   A16
 A2   10   23   A10
 A1   11   22   /CS
 A0   12   21   D7
 D0   13   20   D6
 D1   14   19   D5
 D2   16   18   D4
Vss   16   17   D3
```

What is the DSP1 pin layout?

This was taken from a hacked Pilotwings cartridge with a switch on it - possibly to select between HiROM and LoROM DSP1 games. I'm not 100% sure that the following is correct or complete though.

```
Vcc   01   28   Vcc
Vcc   02   27   A14 (A12 - used for HiROM?)
 nc   03   26   /CS
 nc   04   25   /RD
 nc   05   24   /WR
 D0   06   23   ?
 D1   07   22   ?
 D2   08   21   Vcc
 D3   09   20   Vcc
 D4   10   19   Vcc
 D5   11   18   Vcc
 D6   12   17   GND
 D7   13   16   /RESET (inverted RESET- SNES slot)
 D8   14   15   CLOCK?
```

If you can verify/correct this, it would be greatly appreciated.

What is the MAD-1 and its pin layout?

The MAD-1 stands for Memory Address Decoder revision 1. It is used on the Donkey Kong Country (1 and 2) cartridge and possibly other cartridges in order to address one or two ROMs and a static RAM.

```
/HI   01   16   /LO
/SE   02   15   A13
      03   14   A14
/RE   04   13   BA5
Vcc   05   12   A15
Vcc   06   11   /CS (p49 SNES slot)
Vcc   07   10   Vcc
GND   08   09   RESET (p26 SNES slot)
```

```
/RE - /CS on a 32Mbit ROM (possibly for MAD-1a only)
/LO - /CS on ROM1 (lower 16mbit)
/HI - /CS on ROM2 (upper 16mbit)
/SE - /CS on Static RAM
```

What is the pin layout of the 16kbit SRAM most commonly used by Nintendo?

It seems that Nintendo uses this SRAM in many of their games, mainly because it is very cheap, only $A5 (retail) - much cheaper for Nintendo who buys millions of them. It can address up to 2048 bytes or 16kbits.

```
 A7   01   24   Vcc
 A6   02   23   A8
 A5   03   22   A9
 A4   04   21   /WE
 A3   05   20   /OE
 A2   06   19   A10
 A1   07   18   /CS
 A0   08   17   D7
 D0   09   16   D6
 D1   10   15   D5
 D2   11   14   D4
Vss   12   13   D3
```

Cartridge Addressing Schemes

LoROM cartridges:

```
read ROM   /RD, /CS, RESET low
           /WR high
read SRAM  /CS, /RD low
           RESET, /WR high
           A15, BA4, BA5 high
write SRAM /CS, /WR low
           RESET, /RD high
           A15, BA4, BA5 high
```

HiROM cartridges:

```
read ROM   /CS, /RD, RESET low
           /WR high
read SRAM  /RD low
           RESET, /WR, /CS high
           A13, A14, BA5 high
write SRAM /WR low
           RESET, /RD, /CS high
           A13, A14, BA5 high
```

Would anyone like to verify this?

Embedded Cartridge Information

Most of the information in this section was obtained from Mindrape's SNES ROM document, but also a result, of my own investigation.

All values are in decimal unless specified with a trailing 'h', indicating a hexadecimal value.

The starting offset for this information is located at the end of the first page:

LoROM: offset 32704
HiROM: offset 65472

Game title (21 bytes)

The title is in upper case on most games.

ROM makeup (1 byte)

Upper nibble (4 bits):

```
Value ROM speed
  0   SlowROM (200ns)
  3   FastROM (120ns)
```

Lower nibble (4 bits):

```
Value Bank size
  0   LoROM (32kb banks)
  1   HiROM (64kb banks)
```

ROM type (1 byte)

```
Byte ROM type
 0   ROM only
 1   ROM and RAM
 2   ROM and Save RAM
 3   ROM and DSP1 chip
 4   ROM, RAM and DSP1 chip
 5   ROM, Save RAM and DSP1 chip
 19  ROM and Super FX chip
227  ROM, RAM and GameBoy data
246  ROM and DSP2 chip
```

ROM size (1 byte)

```
Byte ROM size
 8     2 MegaBits
 9     4 MegaBits
 10    8 MegaBits
 11   16 MegaBits
 12   32 MegaBits
```

At the time of writing, the largest SNES game is 48Mbit, while 8Mbit cartridges are the most common. There are cartridge sizes of 10Mbit, 12Mbit, 20Mbit and 24Mbit, which are reported as 16Mbit, 16Mbit, 16Mbit and 32Mbit respectively.

Another way of calculating the ROM size is: 1 shl (ROMbyte-7) MegaBits

SRAM size (1 byte)

```
Byte SRAM size
 0   (none)
 1   16 KiloBits
 2   32 KiloBits
 3   64 KiloBits
```

64 KiloBit SRAM's are the largest Nintendo uses (except DOOM?), while most copiers have 256 kiloBits on-board.

Another way of calculating the SRAM size is: 1 shl (SRAMbyte+3) KiloBits

Country (1 byte)

```
Byte Country                                  Video system
 0   Japan                                    NTSC
```

1	USA	NTSC
2	Australia, Europe, Oceania and Asia	PAL
3	Sweden	PAL
4	Finland	PAL
5	Denmark	PAL
6	France	PAL
7	Holland	PAL
8	Spain	PAL
9	Germany, Austria and Switzerland	PAL
10	Italy	PAL
11	Hong Kong and China	PAL
12	Indonesia	PAL
13	Korea	PAL

License (1 byte)

Byte	Company	Byte	Company
1	Nintendo	131	Lozc
3	Imagineer-Zoom	132	Koei
5	Zamuse	134	Tokuma Shoten Intermedia
6	Falcom	136	DATAM-Polystar
8	Capcom	139	Bullet-Proof Software
9	HOT-B	140	Vic Tokai
10	Jaleco	142	Character Soft
11	Coconuts	143	I''Max
12	Rage Software	144	Takara
14	Technos	145	CHUN Soft
15	Mebio Software	146	Video System Co., Ltd.
18	Gremlin Graphics	147	BEC
19	Electronic Arts	149	Varie
21	COBRA Team	151	Kaneco
22	Human/Field	153	Pack in Video
23	KOEI	154	Nichibutsu
24	Hudson Soft	155	TECMO
26	Yanoman	156	Imagineer Co.
28	Tecmo	160	Telenet
30	Open System	164	Konami
31	Virgin Games	165	K.Amusement Leasing Co.
32	KSS	167	Takara
33	Sunsoft	169	Technos Jap.
34	POW	170	JVC
35	Micro World	172	Toei Animation
38	Enix	173	Toho
39	Loriciel/Electro Brain	175	Namco Ltd.
40	Kemco	177	ASCII Co. Activison
41	Seta Co.,Ltd.	178	BanDai America
45	Visit Co.,Ltd.	180	Enix
49	Carrozzeria	182	Halken
50	Dynamic	186	Culture Brain
51	Nintendo	187	Sunsoft
52	Magifact	188	Toshiba EMI
53	Hect	189	Sony Imagesoft
60	Empire Software	191	Sammy
61	Loriciel	192	Taito
64	Seika Corp.	194	Kemco
65	UBI Soft	195	Square
70	System 3	196	Tokuma Soft
71	Spectrum Holobyte	197	Data East
73	Irem	198	Tonkin House
75	Raya Systems/Sculptured Software	200	KOEI
76	Renovation Products	202	Konami USA
77	Malibu Games/Black Pearl	203	NTVIC
79	U.S. Gold	205	Meldac
80	Absolute Entertainment	206	Pony Canyon
81	Acclaim	207	Sotsu Agency/Sunrise

82	Activision	208	Disco/Taito
83	American Sammy	209	Sofel
84	GameTek	210	Quest Corp.
85	Hi Tech Expressions	211	Sigma
86	LJN Toys	214	Naxat
90	Mindscape	216	Capcom Co., Ltd.
93	Tradewest	217	Banpresto
95	American Softworks Corp.	218	Tomy
96	Titus	219	Acclaim
97	Virgin Interactive Entertainment	221	NCS
98	Maxis	222	Human Entertainment
103	Ocean	223	Altron
105	Electronic Arts	224	Jaleco
107	Laser Beam	226	Yutaka
110	Elite	228	T&ESoft
111	Electro Brain	229	EPOCH Co.,Ltd.
112	Infogrames	231	Athena
113	Interplay	232	Asmik
114	LucasArts	233	Natsume
115	Parker Brothers	234	King Records
117	STORM	235	Atlus
120	THQ Software	236	Sony Music Entertainment
121	Accolade Inc.	238	IGS
122	Triffix Entertainment	241	Motown Software
124	Microprose	242	Left Field Entertainment
127	Kemco	243	Beam Software
128	Misawa	244	Tec Magik
129	Teichio	249	Cybersoft
130	Namco Ltd.	255	Hudson Soft

Game Version (1 byte)

The version is stored as version 1.VersionByte and must be less than 128. i.e. Less than 1.128.

Inverse ROM Checksum (2 bytes)

This is the same as XORing the two checksum bytes. i.e. The checksum bits are inversed.

ROM Checksum (2 bytes)

The checksum is a 16bit word with the lower 8bits stored first, followed by the upper 8bits.

The checksum is calculated by dividing the ROM into 4Mbit chunks then adding all the bytes in these chunks together. Once you have the checksum for each chunk, add them together and take the lower 32bits of the result.

With a non-standard image size, you do not get it equally divisible by 4Mbit (excluding 2Mbit images). e.g. 10Mbit = 4Mbit + 4Mbit + 2Mbit chunks.

Therefore, you must create a 4Mbit chunk from what is left over. Using the same example, you would add the checksum of the following chunks to get the ROM checksum:

4Mbit + 4Mbit + (2Mbit + 2Mbit)
or
4Mbit + 4Mbit + (2 x 2Mbit)

Non Maskable Interrupt / VBL Vector (2 bytes)

```
LoROM: at offset 33274
HiROM: at offset 66042
```

Reset Vector (2 bytes)

Where to start the ROM code.

```
LoROM: at offset 33276
HiROM: at offset 66042
```

How do I know if the ROM is HiROM or LoROM?

When you OR the checksum bytes of a disk image and the inverse checksum bytes, the result should be FFFF hex. Therefore, to detect whether an image is HiROM or LoROM, you must read those bytes, OR them, and see if they equal FFFF hex.

The ROM's type depends at which location the OR'd bytes equal FFFF hex. If it isn't found at either location, then the other way of checking is to see at which location the title contains uppercase alphanumeric characters. (But this fails with most Japanese cartridges)

Why don't you use the ROM Makeup Byte? You can, and some utilities do, but some utilities allow you to change this byte, so incorrect results may occur.

For the actual ROM, the embedded cartridge information is stored at the same position for both LoROM and HiROM. In this case, you must use the ROM Makeup Byte or read a 64kb page and see if both 32kb chunks (upper and lower 32kb) are the same. If they are the same, it is LoROM (32kb pages - A15 is not used, the data repeats itself) otherwise it is HiROM.

As a general rule of thumb, if you can't detect which ROM type it is, default to LoROM, as these are the most common of cartridges.

Cheat Device Decoding

We'll start with the easiest first then work our way down. These codes work by replacing a byte at a specific location in the ROM.

E.g. In the game F-Zero, at a particular position in the ROM, there is a number 3 indicating 3 lives to start off with. What a cheat code will do is replace this byte with, let's say, the number 9, so now when the game is run, the player starts off with 9 lives.

Pro Action Replay (hardware)

```
Code format:        AAAAAADD            (8 digits)

A - Address
D - Data
```

These codes are in Hex, the address being a CPU address, not a direct ROM location (more about this later).

Gold Finger (software)

```
Code format:        AAAAADDDDDDCCW      (14 digits)

A - Address
D - Data
C - Checksum
W - What to change (DRAM or SRAM)
```

This code was designed for the copiers, and are straight Hex characters. Therefore the Address is a ROM address, not a CPU address. Data bytes are arranged in 2 characters (2 D's per byte), which allows for 3 bytes. If a byte is not being used, it is denoted by 'XX'. I have never seen a code with three unused bytes - what's the point of one anyhow?

The address (A's) is a base address. The first data byte (D's) is to be placed at this address. The second at address+1, the third at address+2 (if to be used, that is, if they are not 'XX').

To calculate the checksum you must take the A's and D's, add a zero (0) to the front of the shortened code, then divide into block's of 2 hex digits (bytes). Add these hex digits together (2 characters per hex digit) then minus 160 hex (352 decimal). Now AND this number by FF hex (255 decimal) to get the lower 8 bits (byte). Convert this number to hex and you have your checksum (C's).

W tells the copier whether to replace the byte in the DRAM (ROM image) or the SRAM (Saved game static RAM) of the copier.

Value of W	**Where to place byte**
0	DRAM (ROM image)
1	SRAM (Saved game image)

The rec.games.video FAQ specifies that there may be non- standard values of 2, 8, A, C, F for W, which may be converted to 0. I personally have only seen Gold Finger codes with W = 0.

Game Genie (hardware)

```
Code format:        DDAA-AAAA      (8 digits)

A - Address
D - Data
```

This is the most difficult code to decipher out of the lot. It is as follows:

First take the code in the form xxxx-xxxx and take out the dash ('-') to form xxxxxxxx. Convert these characters (Genie Hex) to normal hex characters using the following table:

Genie Hex:	D	F	4	7	0	9	1	5	6	B	C	8	A	2	3	E
Normal Hex:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

The first two characters is the data byte in Hex. Now take the other 6 following characters (encoded address) and put it into it's binary form of 24 bits.

Now take each bit of the encoded address and rearrange to form the real address:

```
24bit encoded address: ijklqrst opabcduv wxefghmn
8bit encoded data:     ABCDEFGH
```

Rearrange as:

```
24bit address              : 8bit data
```

```
abcdefgh ijklmnop qrstuvwx: ABCDEFGH
MSB                    LSB  MSB  LSB
```

Bit 23 of the encoded address (bit 15 of the real address) is always 1. The reason being that the SNES CPU address must be 1 for it to access the ROM.

Converting between CPU addresses and ROM addresses

This is very easy once you understand how it is done. To convert from a CPU address to a ROM address, all you need to do is remove bit 15. By doing this, I don't mean just setting it to 0. I mean by removing it, then moving all bits after it down one.

```
e.g. ROMaddress = (CPUaddress and 7FFFh) or ((CPUaddress and FF0000h) shl 1)
```

Therefore, to convert from a ROM address to a CPU address, you must insert a high bit into position 15 (bit 15).

```
e.g. CPUaddress = (ROMaddress and 7FFFh) or ((ROMaddress and 7F8000h) shr 1) or 8000h
```

Easily converting between codes

I have made available two DOS programs with source code which allow you to convert between Game Genie and Gold Finger codes. These are available freely from the Turtle Group Inc.

Note: Because the Gold Finger can only address upto 8Mbit of game data, while other codes can address upto 64Mbit of game data, some Game Genie and Action Replay codes may not be converted to Gold Finger.

SNES Copiers

What are copiers?

A copier is a device which sits on top of the SNES and allows you to backup your cartridges as well as play your backed up games. It does this by storing the ROM image of a cartridge to floppy disks via a 1.44Mb disk drive. Most copiers also include a parallel PC port interface, allowing your PC to control the unit and store images on your hard drive.

Copier's contain DRAM from 1 Megabyte to 16 Megabytes, 8MegaBits to 128MegaBits respectively. This is the reason why they are so expensive.

It is legal to own and use a copier for your own personal backup of cartridges which you legally own in this point in time, although it is illegal to distribute this copy (only one copy is allowed). This may vary depending on where you live.

If you wish to make your own "home brew" copier for the SNES, and other consoles, more information can be found at the Turtle Group Inc.

Super Wild Card (SWC) header information

The SWC (Super Wild Card) image format consists of a 512 byte header. It's layout is as follows (set unused bytes to 00h):

```
Offset Function
  0    Lower 8 bits of size word
  1    Upper 8 bits of size word
  2    Image information byte
  8    SWC header identifier (set to AAh)
  9    SWC header identifier (set to BBh)
 10    SWC header identifier (set to 04h)
```

The size word is calculated by multiplying the image size, not game size (in MegaBits) by 16. e.g. Image is 4 Mbits, so size word would be 4*16=64.

Image information byte (in the form of 76543210):

```
Bit  Description
7    1 - Run program in Mode 0 (JMP $8000)
     0 - Run program in Mode 1 (JMP RESET Vector)
6    1 - Multi image (there is another split file to follow)
     0 - Not multi image (no more split files to follow)
5    1 - SRAM memory mapping Mode 21 (HiROM)
     0 - SRAM memory mapping Mode 20
4    1 - DRAM memory mapping Mode 21 (HiROM)
     0 - DRAM memory mapping Mode 20
3/2  00: 256kbit SRAM
     01: 65kbit SRAM
     10: 16kbit SRAM
     11: no SRAM
1/0  reserved
```

Pro Fighter (FIG) header format

This format is similar to the SWC. It consists of a 512byte header who's layout is as follows (set unused bytes to 00h):

```
Offset Function
  0    Lower 8 bits of size word
  1    Upper 8 bits of size word
  2    40h - Multi image
       00h - Last image in set (or single image)
  3    80h - if HiROM
       00h - if LoROM
  4    If using DSP1 microchip:
           FDh - If using SRAM (SRAM size>0)
           47h - If no SRAM (SRAM size=0)
       77h - If not using DSP1 and no SRAM (SRAM size=0)
  5    If using DSP1 microchip:
           82h - If using SRAM (SRAM size>0)
           83h - If no SRAM (SRAM size=0)
       83h - If not using DSP1 and no SRAM (SRAM size=0)
```

Game Doctor file name format

The Game Doctor does not use a 512 byte header like the SWC, instead it uses specially designed filenames to distinguish between multi files. I'm not sure if it used the filename for information about the size of the image though.

Usually, the filename is in the format of: SFXXYYYZ.078

Where SF means Super Famicon, XX refers to the size of the image in Mbit. If the size is only one character (i.e. 2, 4 or 8 Mbit) then no leading "0" is inserted.

YYY refers to a catalogue number in Hong Kong shops identifying the game title. (0 is Super Mario World, 1 is F- Zero, etc). I was told that the Game Doctor copier produces a random number when backing up games.

Z indicates a multi file. Like XX, if it isn't used it's ignored.

A would indicate the first file, B the second, etc. I am told 078 is not needed, but is placed on the end of the filename by systems in Asia.

e.g. The first 16Mbit file of Donkey Kong Country (assuming it is cat. no. 475) would look like: SF16475A.078

Super Wild Card parallel port I/O protocol

I was given this information a while ago. It is supposed to be direct from the company which makes SWC's and I have included this information because a few people have been asking for it. If you have similar information for other backup devices, it would be appreciated if you could send it to me.

```
[PROTOCOL USED IN PC]
* BYTE OUTPUT PROCEDURE
    WAIT BUSY BIT = 1            STATUS PORT BIT7   (HEX n79, n7D)
    WRITE ONE BYTE               DATA LATCH         (HEX n78, n7C)
    REVERSE STROBE BIT           CONTROL PORT BIT0  (HEX n7A, n7E)
* BYTE INPUT PROCEDURE
    WAIT BUSY BIT = 0            STATUS PORT BIT7   (HEX n79, n7D)
    READ LOW 4 BITS OF BYTE      STATUS PORT BIT3-6 (HEX n79, n7D)
    REVERSE STROBE BIT           CONTROL PORT BIT0  (HEX n7A, n7E)
    WAIT BUSY BIT = 0            STATUS PORT BIT7   (HEX n79, n7D)
    READ HIGH 4 BITS OF BYTE     STATUS PORT BIT3-6 (HEX n79, n7D)
    REVERSE STROBE BIT           CONTROL PORT BIT0  (HEX n7A, n7E)
* 5 TYPES OF COMMAND
* COMMAND LENGTH = 9 BYTES.
* COMMAND FORMAT
  BYTE 1   D5                ID CODE 1
  BYTE 2   AA                ID CODE 2
  BYTE 3   96                ID CODE 3
  BYTE 4   00|01|04|05|06    COMMAND CODE
  BYTE 5   al                LOW BYTE OF ADDRESS
  BYTE 6   ah                HIGH BYTE OF ADDRESS
  BYTE 7   ll                LOW BYTE OF DATA LENGTH
  BYTE 8   lh                HIGH BYTE OF DATA LENGTH
  BYTE 9   cc                CHECKSUM = 81^BYTE4^BYTE5^BYTE6^BYTE7^BYTE8
* COMMAND [00] : DOWNLOAD DATA
  al, ah = ADDRESS
  ll, lh = DATA LENGTH
  OUTPUT DATAS AFTER COMMAND
* COMMAND [01] : UPLOAD DATA
  al, ah = ADDRESS
  ll, lh = DATA LENGTH
  INPUT DATAS AFTER COMMAND
* COMMAND [04] : FORCE SFC PROGRAM TO JMP
  al, ah = ADDRESS
* COMMAND [05] : SET MEMORY PAGE NUMBER
  al BIT0-1 = PAGE NUMBER
  al BIT2-7 + ah BIT0-1 = BANK NUMBER
* COMMAND [06] : SUB FUNCTION
  al = 0  INITIAL DEVICE
```

```
al = 1  PLAY GAME IN DRAM
al = 2  PLAY CARTRIDGE
```

ROM Protection Schemes

This section details ways of bypassing the FastROM, PAL/NTSC and SRAM size checks implemented in many SNES games in order to stop people backing them up using copiers.

Note: You don't necessarily have to find and replace all strings to remove the check(s).

SlowROM checks

Most cartridges these days use 120ns ROM in order to get the most out of the ageing SNES. However, there are still many copiers around which emulate ROM at speeds of 200ns meaning they cannot backup the newer cartridges correctly.

Changing the ROM code to bypass the SlowROM check, found in many, but not all FastROM games, allows many people with SlowROM copiers to backup FastROM games.

To patch a ROM and bypass the SlowROM check, you must find any of the following strings in the image and replace it with the patch string: (all codes in hex)

```
Search for           Replace with
A9 01 8D 0D 42       A9 00 8D 0D 42
A9 01 8E 0D 42       A9 00 8E 0D 42
A2 01 8D 0D 42       A2 00 8D 0D 42
A2 01 8E 0D 42       A2 00 8E 0D 42
A9 01 00 8D 0D 42    A9 00 00 8D 0D 42
A9 01 8F 0D 42 00    A9 00 8F 0D 42 00
```

PAL/NTSC checks

Most SNES games have code which detects which video system the cartridge is being played on and refuses to run if not in the right mode. This is to stop people from buying games from other countries before they are released locally.

To bypass the PAL/NTSC check the following patterns must be found and replaced with the ones specified: (all codes in hex)

```
Search for                 Replace with
3F 21 29 10 C9 10 F0       3F 21 29 10 C9 10 80
3F 21 89 10 C9 10 F0       3F 21 89 10 C9 10 80
3F 21 29 10 F0             3F 21 29 10 80
3F 21 00 89 10 F0          3F 21 00 89 10 80
3F 21 00 29 10 F0          3F 21 00 29 10 80
3F 21 89 10 00 F0          3F 21 89 10 00 80
3F 21 29 10 00 F0          3F 21 29 10 00 80
AD 3F 21 29 10 00 D0       AD 3F 21 29 10 00 80
AF 3F 21 00 29 10 D0       AF 3F 21 00 29 10 80
AF 3F 21 00 29 10 00 D0    AF 3F 21 00 29 10 00 EA EA
AD 3F 21 29 10 D0          AD 3F 21 29 10 EA EA
AD 3F 21 29 10 F0          AD 3F 21 29 10 80
AD 3F 21 89 10 D0          AD 3F 21 89 10 80
```

```
AD 3F 21 29 10 C9 00 F0   AD 3F 21 29 10 C9 00 80
AF 3F 21 00 29 10 00 F0   AF 3F 21 00 29 10 00 80
AF 3F 21 00 89 10 00 F0   AF 3F 21 00 89 10 00 80
```

SRAM size checks

Some SNES games check to see how much SRAM is connected to the SNES as a form of copy protection. As most copiers have 256kbits standard, the game will know it's running on a backup unit and stop to prevent people copying the games. However, the newer copiers get around this detection somehow.

To disable the SRAM size check in a ROM image, search for the following and replace as appropriate.

Note: All codes are in hex, although 'xx' means anything, while a comma means search for either of the two or more (enclosed in brackets).

```
Search for     (8F, 9F) xx xx 70 (CF, DF) xx xx 70 D0
Replace with   (8F, 9F) xx xx 70 (CF, DF) xx xx 70 EA EA (if SRAM size of game = 64kb
               (8F, 9F) xx xx 70 (CF, DF) xx xx 70 80  (if SRAM size of game <> 64kb:

Search for     (8F, 9F) xx xx (30, 31, 32, 33) (CF, DF) xx xx (30, 31, 32, 33) D0
Replace with   (8F, 9F) xx xx (30, 31, 32, 33) (CF, DF) xx xx (30, 31, 32, 33) 80

Search for     (8F, 9F) xx xx (30, 31, 32, 33) (CF, DF) xx xx (30, 31, 32, 33) F0
Replace with   (8F, 9F) xx xx (30, 31, 32, 33) (CF, DF) xx xx (30, 31, 32, 33) EA EA

Search for     (8F, 9F) xx xx (30, 31, 32, 33) AF xx xx (30, 31, 32, 33) C9 xx xx D0
Replace with   (8F, 9F) xx xx (30, 31, 32, 33) AF xx xx (30, 31, 32, 33) C9 xx xx 80
```

Many thanks to Chp for making his uCON v1.41 source publicly available, from which these patterns came.

IPS Patch Format

This patch format is used a lot for patching SNES ROM images. Therefore I have included it's format in this text.

The format is as follows:

```
   Description                  Size
IPS file identifier            5 bytes (characters PATCH)
Offset in file to place patch  3 bytes
Number of bytes in patch       2 bytes (allows 65535 patch bytes)
Patch byte(s)                  (specified by 'No. of bytes in patch')
        .                              .
        .                              .
Start again, looking           3 bytes (characters EOF)
for new offset, unless
and EOF is found.
```

Sample IPS file contents with 2 offset points:

PATCHHooonn?ooonn?EOF

```
o - Offset in file
n - Number of bytes in patch
? - Data byte(s) (n number of bytes)
```

Acknowledgements

The following people have contributed to this text, whether they know it or not. Many thanks to them for their wonderful contribution(s).

Donald Moore	(moore@futureone.com)
Chp	(ronaldm@netcom.com)
Thomas Rolfes	(Thomas_Rolfes@ms.maus.de)
Jeremy Chadwick	(yoshi@parodius.com)
Nigel Bryant	(nbb@essex.ac.uk)

Also used for the creation of this text was the rec.games.video Frequently Asked Questions (FAQ) file; a FAQ with a huge amount of information on consoles in general.

Back

Questions, comments or complaints can be sent via e-mail.

Last updated 1st January 1997