



<b>INTRODUCTION .....</b>	<b>3</b>
<b>PATCH FILE .....</b>	<b>4</b>
LAYOUT.....	4
IPS DEFINITION SECTION. ....	5
<i>Memory Mode</i> .....	5
<i>Super Magicom (SMC) Header</i> .....	5
<i>Create Patching Object (IPO)</i> . ....	5
USER SECTION.....	5
<b>FUNCTIONS AVAILABLE .....</b>	<b>6</b>
OPEN A 'PATCH' SEGMENT.....	6
<i>Description</i> .....	6
<i>Remarks</i> .....	6
<i>Syntax</i> .....	6
<i>Example</i> .....	6
CLOSE A 'PATCH' SEGMENT.....	7
<i>Description</i> .....	7
<i>Syntax</i> .....	7
<i>Example</i> .....	7
FILL A ROM AREA. ....	8
<i>Description</i> .....	8
<i>Syntax</i> .....	8
<i>Remarks</i> .....	8
<i>Example</i> .....	8
CONVERT A HiROM TO LoROM ADDRESS. ....	9
<i>Description</i> .....	9
<i>Syntax</i> .....	9
<i>Remarks</i> .....	9
<i>Example</i> .....	9
PATCH THE SMC HEADER.....	10
<i>Description</i> .....	10
<i>Syntax</i> .....	10
<i>Remarks</i> .....	10
<i>Example</i> .....	10
<b>ADDITIONAL POINTS.....</b>	<b>11</b>
PATCHING COLOURS.....	11
MAKING PATCHING OBJECTS (IPOS).....	11

## **Introduction**

This is the manual for writing S-NES Assembly (ASM) Language to be outputted as an International Patching System (IPS) File. It will cover the layout of '.PAT' file to be assembled; the initial settings needed to describe what type of file you are patching to; and what patching functions are available to you, and their format.

I developed this method in late 1997. Its use has ranged from modifying data/code in SMCs, repairing faulty headers; to being the basis for an ASM environment in all the translations I've worked on (and used by other translation groups, too), due to it's ease of maintaining/inserting code.

This is the Psy-Q/SNSYS version of the method; if you use the x816 freeware assembler, then be sure to check my homesite.

Please let me know if you have any problems. It'd also be nice to know if you actually get some use out of it! ☺

Thanks again.

-- FH 23/05/04

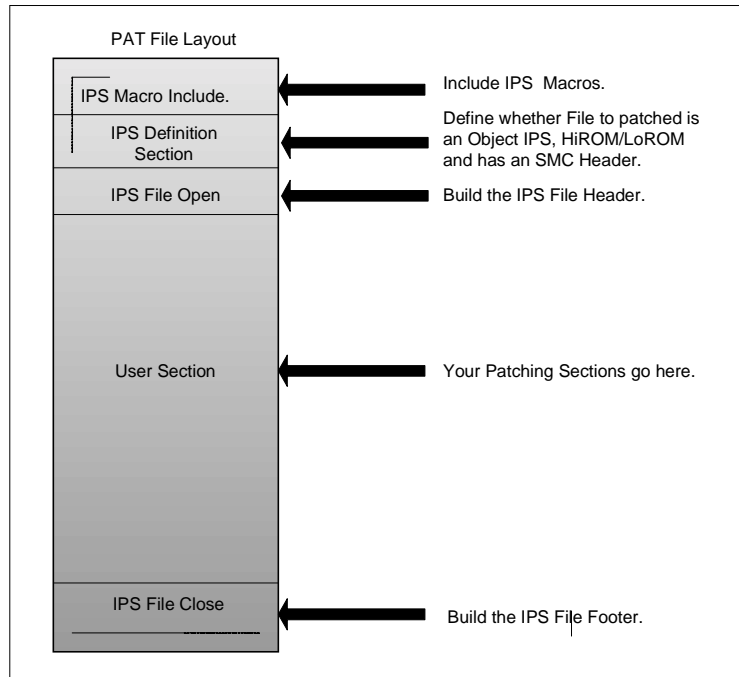
[fh512@yahoo.co.uk](mailto:fh512@yahoo.co.uk)

<http://travel.to/fh>

## Patch File

### Layout

The following diagram gives a visual representation of the PAT file.



And here is the 'PAT' template of the above diagram.

```

; _____ IPS MACRO INCLUDE _____
      INCLUDE "IPS_PSYQ.INC"      ; Patching Macros for PsyQ/SNSYS.
; _____IPS DEFINITION SECTION _____
IPS_SMCHDR = IPS_SMCHDR_YES      ; Either 'IPS_SMCHDR_YES' or 'IPS_SMCHDR_NO'.
IPS_ROMMAP = $20                  ; Either '$20' or '$21'.
IPS_OBJ   = IPS_OBJ_YES          ; Either 'IPS_OBJ_YES' or 'IPS_OBJ_NO'.
; _____IPS OPEN SECTION _____
      IPSOpen                      ; Open IPS Output File.
; _____ USER SECTION _____

; _____IPS CLOSE SECTION _____
      IPSClose                      ; Close IPS Output File.
; _____
; _____ END OF PATCH FILE _____
; _____

```

The only sections that will be modified will be the 'IPS DEFINITION SECTION' and the 'USER SECTION'. Those will be covered now.

## **IPS Definition Section.**

### **Memory Mode**

The S-NES has two main memory maps: Mode20 (LoROM) and Mode21 (HiROM). These group binary information in 32 and 64kbytes respectively. Use this setting to define which Memory Type you are patching to.

Set to either:

**IPS\_ROMMODE = \$20** ; For Mode 20 (LoROM) Patching.

Or

**IPS\_ROMMODE = \$21** ; For Mode 21 (HiROM) Patching.

Currently, patching Memory Maps 20 and 21 are supported.

### **Super Magicom (SMC) Header.**

SMC Files begin with a 512-byte header, which describes how the File should be loaded into a backup unit. If you are patching to a file that has no SMC header, then set **IPS\_SMCHDR = IPS\_SMCHDR\_NO**.

If you are patching to a file that has a SMC header, then set **IPS\_SMCHDR = IPS\_SMCHDR\_YES**.

### **Create Patching Object (IPO).**

This setting defines whether the IPS Header and Footer are added to the output file. If you set **IPS\_OBJ = IPS\_OBJ\_YES**, it will be unable to patch to a file by itself, but can be Binary included into another PAT file. This topic will be covered later in 'Making Patching Objects'. For the moment, set **IPS\_OBJ = IPS\_OBJ\_NO**.

## **User Section**

This section allows you to build your Assembly Language code. It functions in the normal way one would enter code; with the exception that any data is placed in 'Patch Sections'. We shall now cover what functions are available to you in the 'USER SECTION'.

## Functions available

### Open a 'Patch' Segment.

#### Description

This Macro will set up the virtual address to assemble your code. The physical address produced will be dependant on the LoROM/HiROM and the SMC Header Definitions.

#### Remarks

Any code entered within the Patch Macro will be assembled with the passed Patch Address as the starting address for the code. This allows flexible relocating and writing of your code to any ROM address. All patches must close with the 'Eop' Macro (see next section).

#### Syntax

##### Patch 24-bit Address

#### Example

```

; --- My patch is set to LoROM, and I am patching
; --- to a File with an SMC Header.
NEW_CODE_AREA EQU $009f1c ; Define Our New Code Area.

; --- Change Reset Vector to Jump to Our New Code.
    Patch $00fffc ; Patch Reset Vector
    DW NEW_CODE & $FFFF ; Patch Reset Vector with New Code Address.
    Eop ; Close Patch

; --- Place Code In New Area.
    Patch NEW_CODE_AREA ; Currently patching to $007f00.
    php ; Preserve Status Flags.
    rep #$21 ; Accum 16-bit and Clear Carry.
    Eop ; Close Patch

; --- Change the ROM Title
    Patch $00ffc0 ; Address for ROM Title.
    DB "BARNEY VS. THE BEAR " ; 21-Byte Title.
    Eop ; Close Patch

```

The above code would patch to the following hex Addresses in the SMC file:

```

+-----+
| ADDR dec | ADDR hex | Bytes | RLE byte |
+-----+-----+-----+-----+
| 33276 | 0081FC | 2 | |
| 8476 | 00211C | 3 | |
| 33216 | 0081C0 | 21 | |
+-----+

```

## Close a 'Patch' Segment.

### Description

This 'End of Patch' Macro will close the current patch segment.

### Syntax

**Eop**

### Example

```
; ---
    Patch $8900                ; ← This will expand as $008900.
    bit    ProcFlg              ; Check if we need to process
    bpl    @noProc
    jmp    ExitCode            ; ← This will be assembled to as 'jmp $890a'
@noMult
    inc    Value
    rts
ExitCode
    INCLUDE "exit.asm"         ; Include external source code into patch Module.
NewCharacterSet:
    INCBIN "chrset.bin"        ; Include external Binary Data into patch Module.
    Eop                        ; ← Always close the Patch.

    INCBIN "vwf.ipo"          ; Add in a IPO file.
```

## Fill a ROM Area.

### Description

This macro defines a Run Length Encoding Segment.

### Syntax

**RLE 24-bit Address, 16-bit Length, 8-bit FillValue**

### Remarks

This Macro is quite useful for Expanding a ROM to a bigger size or blanking selected areas.

### Example

```

; --- My Patch is set to LoROM.
TRAINER_CODE      EQU   $8200

        Patch TRAINER_CODE                ; ← Our new code area.
        lda      $1234
        sta      $00
        rts

TRAINER_CODE_END = *
        Eop                                ; ← Close Patch Segment
TRAINER_CODE_BANKFILL = $10000-(TRAINER_CODE_END&$FFFF) ; Remaining area in bank.
        RLE TRAINER_CODE_END,TRAINER_CODE_BANKFILL,$00 ; Fill Remaining Bank with 0's

```

The IPS was patched as follows (as shown, using IPS.exe's '/l' command):

```

+-----+
| ADDR dec | ADDR hex | Bytes | RLE byte |
+-----+-----+-----+-----+
|    1024  | 000400  |     6  |          |
|    33286 | 008206  | 32250  |     0    |
+-----+-----+-----+-----+

```

## Convert a HiROM to LoROM Address.

### Description

This Macro Converts a HiROM Address into a loROM Address

### Syntax

**H2L 24-bit Address**

'Our\_Label' = @l\_addr

**ENDH2L**

### Remarks

This was borne out from finding potential free space, by viewing in a hex editor, and the need to quickly add that space into the source.

### Example

```

SMC_HEADER EQU 512 ; SMC Header is 512 bytes in size.
ROM_FSPACE EQU $112000-SMC_HEADER ; Convert the File Address, $112000, to a
; HiROM address by subtracting the SMC Header Size.

; -- Create LoROM Address from HiROM Address.
H2L ROM_FSPACE ; Open HiROM to LoROM conversion Macro.
FREESPACE = @l_addr ; Pass the local Macro Address value to your label.
ENDH2L ; Close the conversion Macro.

Patch FREESPACE
rep #20
stx $1234
Eop

```

And, from the listing, we can see that the coded is assembled at the converted LoROM address, \$229e00.

```

229E00 C2 20 REP #20
229E02 8E 34 12 STX $1234

```

## Patch the SMC Header

### Description

This Macro creates a Patch module for use within the 512-byte SMC address range, and allows modifications to the SMC File settings.

### Syntax

**Patch\_SmcHdr** *10-bit Address*

### Remarks

As with the Patch Macro, any code entered will be assembled with the passed Patch Address as the starting address for the code. It must also close with the 'Eop' Macro

### Example

```

; --- My Patch is set to LoROM.

; --- Rebuild SMC Header.
; Set New 8Kbyte Page Count.
; Set game to work in EmuMode2 (Game Emulation with some back-up unit registers accessible); No Multi-File,
; MapMode20, No Static RAM, and the Cartridge DSP Registers visible.
; Set as a S-NES Game File.
ROM_SIZE EQU      $180000
INCLUDE "smc.inc"           ; Include file for SMC definitions.
Patch_SmcHdr $000
DW ROM_SIZE >>14
DB SMC_EMU_GMODE_2 | SMC_EMU_MULTI_N | SMC_EMU_DMODE20 | SMC_EMU_SRAMOFF | SMC_EMU_EXT_ON

DS 5                        ; Filler bytes.
DB SMC_ID1
DB SMC_ID2
DB SMC_FTYPE_GFILE_SNS
Eop

```

For game translations, it would allow you to reconfigure the size, Save RAM, etc. (though it's more preferable to use my C-Lib solution to header fixing). It also is very useful in fixing incorrect Headers for DSP-1 Mapping and Save RAM files.

## Additional Points.

### Patching Colours

Sometimes it may be necessary to alter or create colours in a game. There are two Macros which cater to this: 'COLWRD' AND 'TLPCOL'. These two macros will allow you to convert, either the B,G,R or Tile Layer Pro representation of a colour, to the S-NES colour-word format. Here is an example of both at use.

```

GAME_INTRO_PALAREA EQU $108290 ; Location of Palette for Game's Intro.

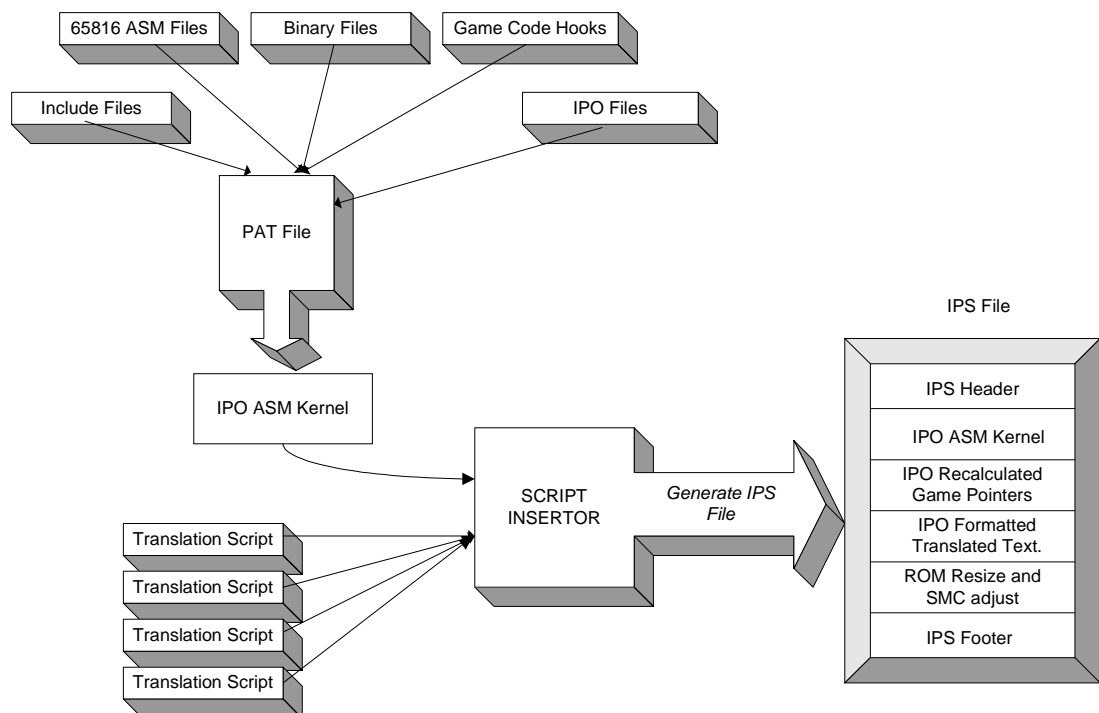
Patch GAME_INTRO_PALAREA ; Patch Section to rewrite colours.
COLWRD $1f,$1f,$1f ; BGR Full White.
COLWRD $00,$00,$1f ; BGR Full Red.
COLWRD $00,$1f,$00 ; BGR Full Red.
COLWRD $1f,$00,$00 ; BGR Full Blue.

TLPCOL 255,255,255 ; Tile Layer Pro Full White.
TLPCOL 000,000,255 ; Tile Layer Pro Full Blue.
TLPCOL 000,255,000 ; Tile Layer Pro Full Green.
TLPCOL 255,000,000 ; Tile Layer Pro Full Red.
Eop ; Close Patch Segment

```

### Making Patching Objects (IPOs).

As stated in the directive section of this manual, you can create a IPS file with no header or footer to create a 'IPO'. These are extremely useful in translation projects, in the fact your testers, key script/grammar editors can create IPS translations through your customer Script Inserter; which will read in the latest build of your ASM Kernel and combine it with the Script Inserter's output IPS. Here is a diagram showing the translation process we use (note how it is not dependent on a ROM to overwrite).



For some information on the Inserter code, please refer to the homepage (unfortunately, I will not release full source code/descriptions due to unofficial Retranslations of our translation projects).

=== END OF FILE ===