Romancing Saga1 'Quick Notes'

Forward

Hello there! Thanks for taking the time to check out our document, 'Romancing Saga 1 Quick notes'. This document covers some of the processes involved to understand and alter code, thus bypassing constraints sometimes met when translating games from Japanese to English. It's built from my own notes, so it may not be the most comforting read! ☺ It is also far from finished/structured/coherent. I felt, what with the minimal feedback from my last document, I'll wait to see if there's any response before turning the notes into something that's more of a tutorial/learning resource.

I have a few reasons for releasing this information. Primarily, I hope it encourages others to venture into this field or, at least, get people to think or ask questions about this stuff.
Other reasons would be in the general decline in 'creativity'(?) in translations, and opting for 'off-the-shelf' tools. I'm sure a lot of tool writers/ASM programmers out there will agree with me in that there's no greater feelin' when you write your first Script Dumper!! ☺ Kudos to J3d! for his document all those years ago which taught me! ;)
Another reason is that it justifies why Translation projects such as RS1 go quiet for periods!! This is the kind of thing that is being worked on in those silent times.
The not-so-great point in releasing this information is the rise in Retranslations, using off-the-shelf editors to write over other people's translation projects. I promised myself I'd keep this sore point out of the document, so… I hope you get something from it too (though not as a guide to retranslating RS1 =D).

If you have any questions about anything covered in the document, you're welcome to drop me an email at fh512@yahoo.co.uk, Anyways, I hope you do get something useful out of it. Thanks again for reading!!!

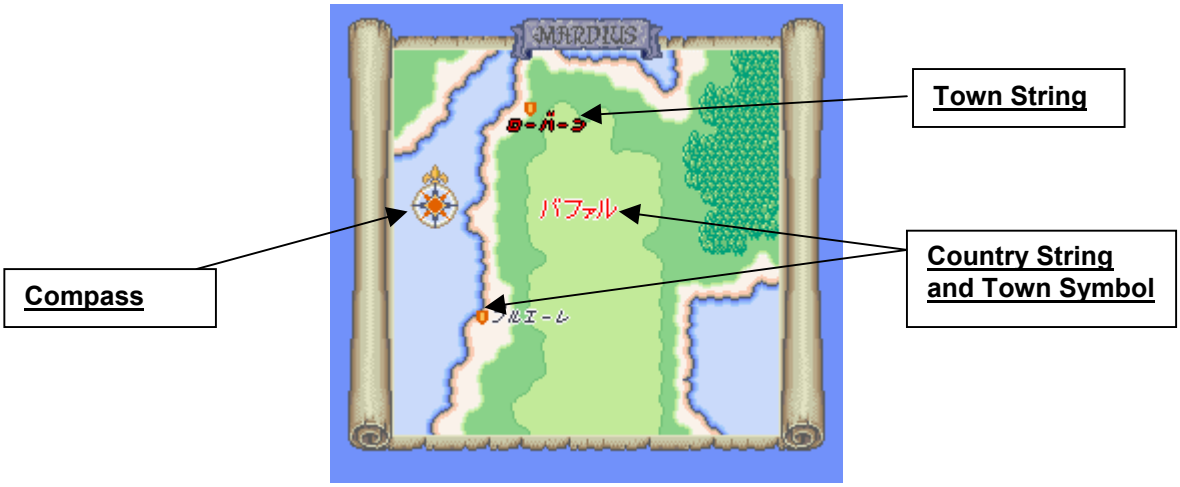
-- FH: RS1 English Translation Project member: http://get.to/rsaga1

Sincere Greetings and Thanks to:

David Mullen, Jason Li, Nocomply, Darkforce, all members of the Translation Projects I've been involved with, Spinner8, Skeud, and all my regular website visitors! ☺

Overview

A point was reached in the Romancing Saga1 (RS1) Translation, David pointed out we had not translated the Map Text in the game. The Map Screen is show below:



Using snes9x to locate how the S-NES was displaying the text (Keys 1~5), it was found that the Town String, Compass, Country String and Town Symbol were being displayed as Objects.

This document will go into detail on the following game code routines, which are used to create the above screen:

BuildCompass()	// Place the Compass on the Map.
BuildCountryStr()	// Place Country string and add the Town Symbol.
BuildTownString()	// Place the Town String.
RS1E_BuildTownString()	// Our code to place the Town String.

Discussion of Objects

Objects.

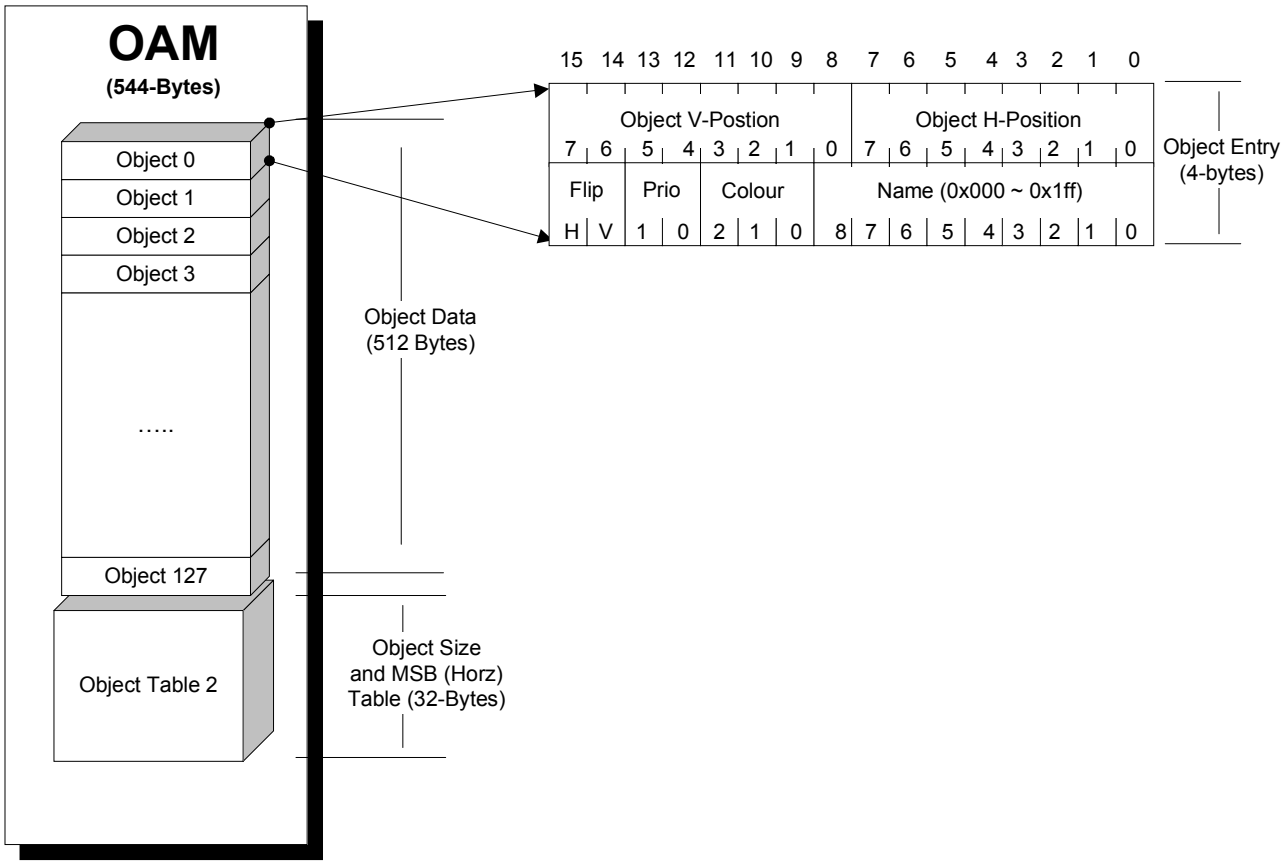
As RS1 is using Objects (or 'Sprites') to display text, it is important to cover the rudiments of Objects, how the S-NES displays them and how RS1 updates them.

Description of an Object.

An Object could be described as a 'movable graphic', such as the character 'Mario', or a bullet from a space ship. An individual Object has properties such as Horizontal and Vertical position on the screen, orientation (say, upside down or mirror flipped) or the number ('name') of the object, which indicates whether is, say, a turtle shell or Mario.

S-NES OAM ('Object Attribute Memory').

The S-NES hardware can render up to 128 Objects. Object properties are held in S-NES' Graphics Memory known as the OAM. The format of the OAM looks like this:



OAM Buffer.

The OAM can only be accessed and updated when the S-NES is not writing a picture to the television screen. The period when most games change OAM data is known as the 'Vertical Blank'.

Since the Vertical Blank is a short space of time to alter positions, Object data is built whilst the screen is displaying, and stored in an 'OAM Buffer' in Work RAM. This buffer is a mirror image of the 544-Byte OAM layout; this means, when the Vertical Blank occurs, we transfer a 'carbon copy' of our OAM Buffer to the OAM (thus only taking a short time to transfer).

What the following text routines will show, is the manipulation of the OAM Buffer for text display.

Map Routines

Compass

Description

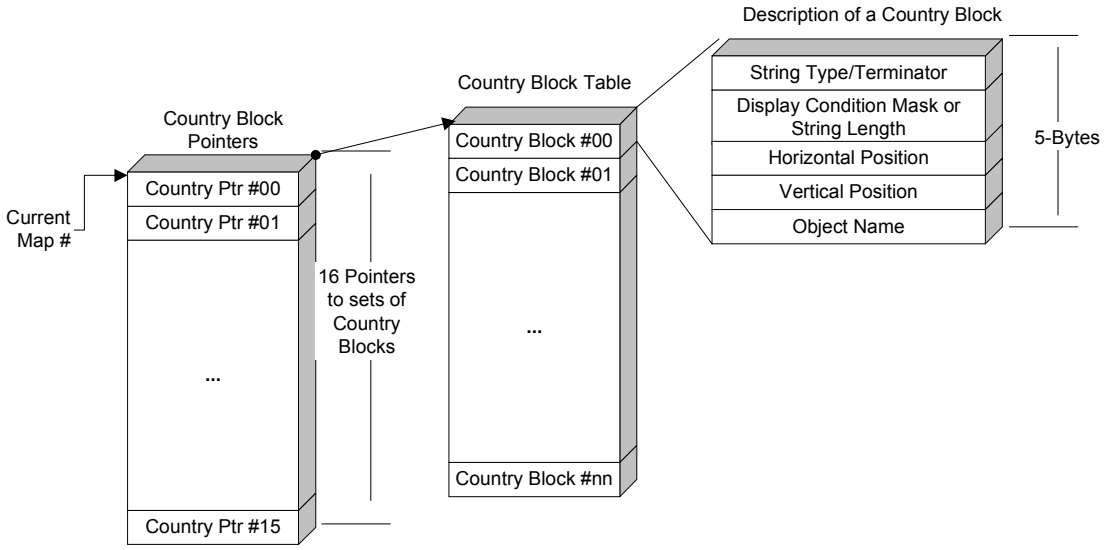
This routine Fetches Horizontal and Vertical components for the Compass. It will then construct a 3*4 Object grid which will be the compass.

Disassembly

```
; === Subroutine: Generate Compass =====
; This function starts from fixed Object
; name $01f3, and will build a 3*4 tile
; grid to display the compass at the
; Designated H/V Position.

00ebc6 ad 81 16      LDA $1681        ; Fetch Map number.
00ebc9 c2 20         REP #$20         ; Accum (16 bit)
00ebcb 29 ff 00      AND #$00ff       ; Mask extraneous data.
00ebce 0a            ASL              ; Mult by 4.
00ebcf 0a            ASL              ; ...
00ebd0 aa            TAX              ; Use as index.
00ebd1 bf 02 80 19   LDA $198002,X    ; Fetch H/V tile pos (th=lowbyte, tv=highbyte).
00ebd5 20 b8 ec      JSR $ecb8        ; TilePos2Pixelpos.
00ebd8 e2 20         SEP #$20         ; Accum (8 bit)
00ebda 20 de eb      JSR $ebde        ; SetupFixedOamEntry.
00ebdd 60            RTS              ; Return.
; ======= End of GenerateCompass =======

; === Function:  Tile2PixelPos ==========
00ecb8 18            CLC              ; Clear Carry for addition.
00ecb9 69 04         ADC #$0304       ; Offset 3 Vertical tiles, 4 Horizontal tiles.
                     AND #$1f1f       ; Mask to 0~31 tile range.
                     ASL              ; Mult by 8 to give pixel position
                     ASL              ; ...
00ecc1 0a            ASL              ; ...
00ecc2 85 1c         STA $1c          ; Store in this temp.
00ecc4 60            RTS              ; Return.
;======= End of Tile2PixelPos ==========

; === Function: SetupFixedOamEntry =====
00ebde a9 f3         LDA #$f3         ; Fixed OAM name??
00ebe0 85 22         STA $22          ; Store in temp.
00ebe2 a9 04         LDA #$04         ; Setup a count of 4.
00ebe4 85 20         STA $20          ; Store in V-Row Counter.
00ebe6 a6 31         LDX $31          ; Load X with OAM index.

00ebe8 a9 03         LDA #$03         ; Setup count of 3.
00ebea 85 1f         STA $1f          ; Store in H-Row counter.
00ebec a5 1c         LDA $1c          ; Load H-Pos of Object.
00ebee 48            PHA              ; Push onto stack.
00ebef 20 a1 ec      JSR $eca1        ; AddObj2Buffer
00ebf2 68            PLA              ; Restore H-Pos.
00ebf3 18            CLC              ; Clear carry for addition.
00ebf4 69 08         ADC #$08         ; Add 8-pixels to horizontal position of Obj.
00ebf6 e6 22         INC $22          ; Increment OAM Name.
00ebf8 c6 1f         DEC $1f          ; Decrement loop count.
00ebfa d0 f2         BNE $ebee        ; Branch back if not done for all.

00ebfc a5 1d         LDA $1d          ; Load V-Pos.
00ebfe 18            CLC              ; Clear carry for addition.
00ebff 69 08         ADC #$08         ; offset by 8 pixels.
00ec01 85 1d         STA $1d          ; Update V-Pos.
00ec03 c6 20         DEC $20          ; Decrement V-Row counter.
00ec05 d0 e1         BNE $ebe8        ; Branch back if not done for all.

00ec07 86 31         STX $31          ; Done for all: Update current OamBuffer index.
00ec09 60            RTS              ; Return.
; ======= End of SetupFixedOamentry ====

; === Function: AddObj2Buffer ==========
; Accum has ObjHPos value upon entry.
00eca1 9d 00 00      STA $0000,X      ; Store ObjHPos.
00eca4 a5 1d         LDA $1d          ; Fetch.
00eca6 9d 01 00      STA $0001,X      ; Store ObjVPos.
00eca9 a5 22         LDA $22          ; Fetch.
00ecab 9d 02 00      STA $0002,X      ; Store ObjName.
00ecae a9 25         LDA #$25         ; %00-10-010-1
00ecb0 9d 03 00      STA $0003,X      ; Store ObjAttrib (Flip,Prio,Pal,TNameMSB)
00ecb3 e8            INX              ; Adjust Index to next ObjEntry.
00ecb4 e8            INX                    ; ...
00ecb5 e8            INX              ; ...
00ecb6 e8            INX              ; ...
00ecb7 60            RTS              ; Return.
;===== End of AddObj2Buffer ============
```

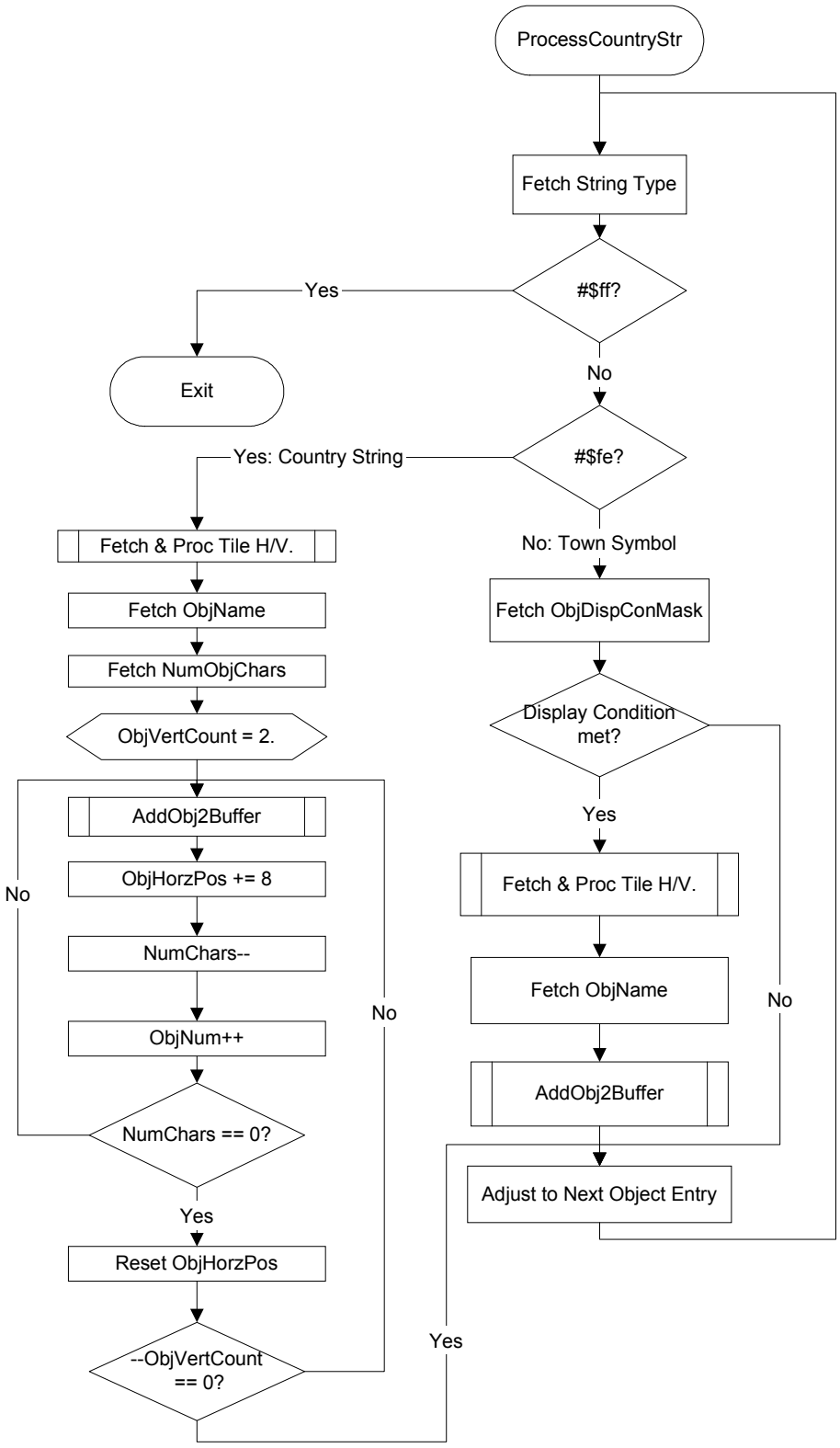
Country/Town Symbol

Description



Memory Structures

Flowcharts


ProcessCountryStr
Fetch String Type
#$ff?
Yes
Exit
No
#$fe?
Yes: Country String
Fetch & Proc Tile H/V.
Fetch ObjName
Fetch NumObjChars
ObjVertCount = 2.
AddObj2Buffer
ObjHorzPos += 8
NumChars--
ObjNum++
NumChars == 0?
No
Yes
Reset ObjHorzPos
--ObjVertCount == 0?
No
Yes
No: Town Symbol
Fetch ObjDispConMask
Display Condition met?
Yes
Fetch & Proc Tile H/V.
Fetch ObjName
AddObj2Buffer
No
Adjust to Next Object Entry
Yes

Disassembly

```
; === Subroutine: WriteCountryString =========
00ec0a ad 81 16      LDA $1681        ; Fetch Map Number.
00ec0d c2 20         REP #$20         ; Accum (16 bit)
00ec0f 29 ff 00      AND #$00ff       ; Mask off extraneous data.
00ec12 0a            ASL              ; Mult by 2 for word access.
00ec13 aa            TAX              ; Use as index.
00ec14 bf 60 80 19   LDA $198060,X    ; Fetch this word value.
00ec18 aa            TAX              ; Use as index.
00ec19 e2 20         SEP #$20         ; Accum (8 bit)

00ec1b bf 00 00 19   LDA $190000,X    ; Fetch this value.
00ec1f c9 ff         CMP #$ff         ; Check if it's '#$ff'. (EOT)
00ec21 f0 35         BEQ $ec58        ; Branch out if 'yes'.
00ec23 c9 fe         CMP #$fe         ; Check if '#$fe'.     (16*16 String)
00ec25 f0 32         BEQ $ec59        ; Branch if 'yes'.

;--- Town Symbol: Check whether condition met to display symbol.
00ec27 20 a4 c7      JSR $c7a4        ; Default:Fetch Nibble from RAM Table (%iiiiiiin)
00ec2a 3f 01 00 19   AND $190001,X    ; Mask fetched nibble with this table value.
00ec2e f0 1c         BEQ $ec4c        ; Branch if not matched (skip display).

;--- Matched: display town text:
00ec30 c2 20         REP #$20         ; Accum (16 bit)
00ec32 bf 02 00 19   LDA $190002,X    ; Fetch Tile H/V Offsets.
00ec36 20 b8 ec      JSR $ecb8        ; Tile2PixelPos.
00ec39 e2 20         SEP #$20         ; Accum (8 bit)
00ec3b bf 04 00 19   LDA $190004,X    ; Fetch ObjName.
00ec3f 85 22         STA $22          ; Store ObjName.
00ec41 da            PHX              ; Preserve X Index.
00ec42 a6 31         LDX $31          ; Load X with OAMBuffer Index.
00ec44 a5 1c         LDA $1c          ; Load ObjHPos.
00ec46 20 a1 ec      JSR $eca1        ; AddObj2Buffer
00ec49 86 31         STX $31          ; Update OAMBuffer Index.
00ec4b fa            PLX              ; Restore X Index.
; Here if condition not met in RAM-Table.

; --- Index adjustment and loop-back.
00ec4c c2 20         REP #$20         ; Accum (16 bit)
00ec4e 8a            TXA              ; Transfer Index to Accum.
00ec4f 18            CLC              ; Clear carry for addition.
00ec50 69 05 00      ADC #$0005       ; Offset Index by +5 (next entry in table).
00ec53 aa            TAX              ; Pass back to X Index.
00ec54 e2 20         SEP #$20         ; Accum (8 bit)
00ec56 80 c3         BRA $ec1b        ; Loop back into fetch loop.

; Here if fetch value '#$ff' (EOT).
00ec58 60            RTS              ; Return.

; Here if fetched value '#$fe'. (16*16 Country String)
00ec59 c2 20         REP #$20         ; Accum (16 bit)
00ec5b bf 02 00 19   LDA $190002,X    ; Fetch H/V tile position (th=lowbyte,
tv=highbyte).
00ec5f 20 b8 ec      JSR $ecb8        ; TilePos2Pixelpos.
00ec62 e2 20         SEP #$20         ; Accum (8 bit)
00ec64 bf 04 00 19   LDA $190004,X    ; Fetch ObjName.
00ec68 85 22         STA $22          ; Store here.
00ec6a bf 01 00 19   LDA $190001,X    ; H-Pos Counter (number of chars).
00ec6e 85 1e         STA $1e          ; Store here.
00ec70 da            PHX              ; Preserve X Index.
00ec71 a6 31         LDX $31          ; Load OAMBuffer Index.
00ec73 a9 02         LDA #$02         ; V-Pos Counter.
00ec75 48            PHA              ; Push onto stack.
00ec76 a5 1c         LDA $1c          ; ObjHPos
00ec78 48            PHA              ; Push onto stack.
00ec79 a5 1e         LDA $1e          ; H-Pos Counter.
00ec7b 48            PHA              ; Push onto stack.
00ec7c a5 1c         LDA $1c          ; ObjHPos.
00ec7e 20 a1 ec      JSR $eca1        ; AddObj2Buffer
00ec81 a5 1c         LDA $1c          ; ObjHPos.
00ec83 18            CLC              ; Clear Carry for Addition.
00ec84 69 08         ADC #$08         ; Offset HPos by +8 Pixels.
00ec86 85 1c         STA $1c          ; Update ObjHPos.
```

```
00ec88 e6 22         INC $22          ; Increment ObjName.
00ec8a 68            PLA              ; Restore H-Pos Counter.
00ec8b 3a            DEC              ; Decrement count.
00ec8c d0 ed         BNE $ec7b        ; Branch back if not done for all.

00ec8e 68            PLA              ; Restore ObjHPos Base position.
00ec8f 85 1c         STA $1c          ; Update ObjHPos.
00ec91 a5 1d         LDA $1d          ; Load ObjVPos.
00ec93 18            CLC              ; Clear Carry for Addition.
00ec94 69 08         ADC #$08         ; Offset VPos by +8 Pixels.
00ec96 85 1d         STA $1d          ; Update ObjVPos.
00ec98 68            PLA              ; Restore V-Pos counter.
00ec99 3a            DEC              ; Decrement count.
00ec9a d0 d9         BNE $ec75        ; Branch back if not done for all.

00ec9c 86 31         STX $31          ; Done for all: Update current OamBuffer index.
00ec9e fa            PLX              ; Restore index.
00ec9f 80 ab         BRA $ec4c        ; Adjust index to next entry and loop back.
; ========= End of WriteMapText ========
```

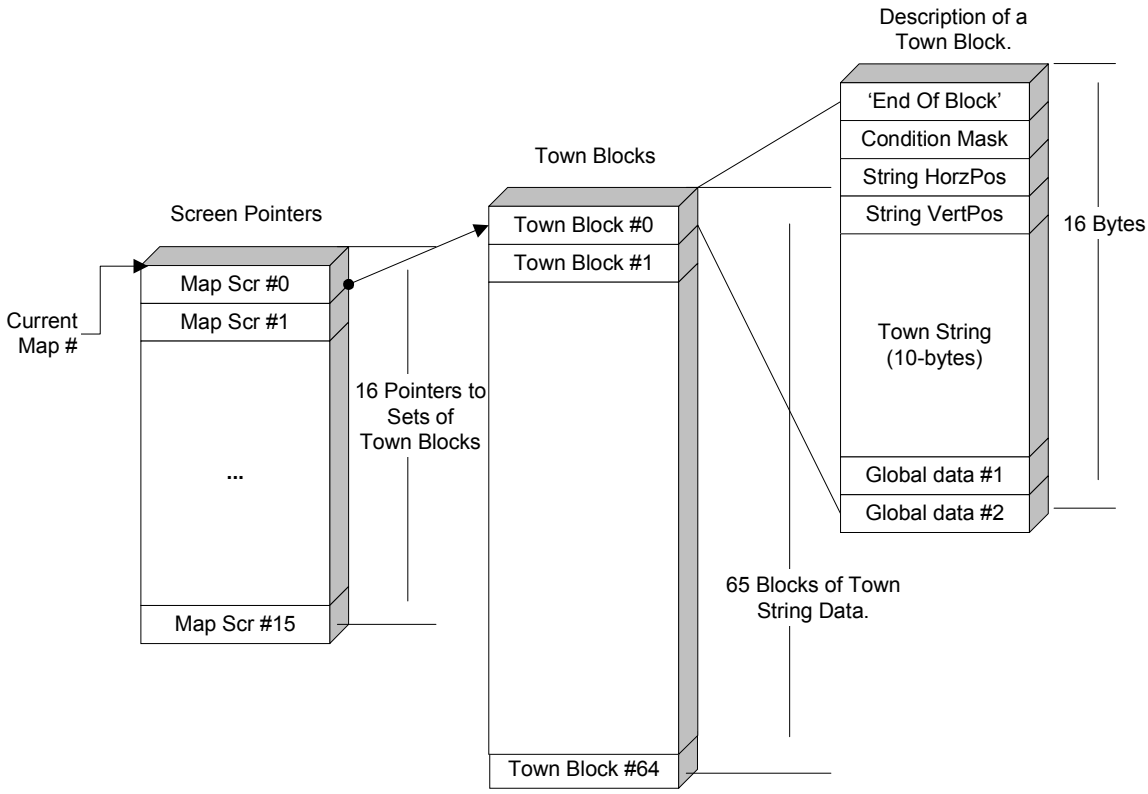
Town Text

Description

Memory Structures

Description

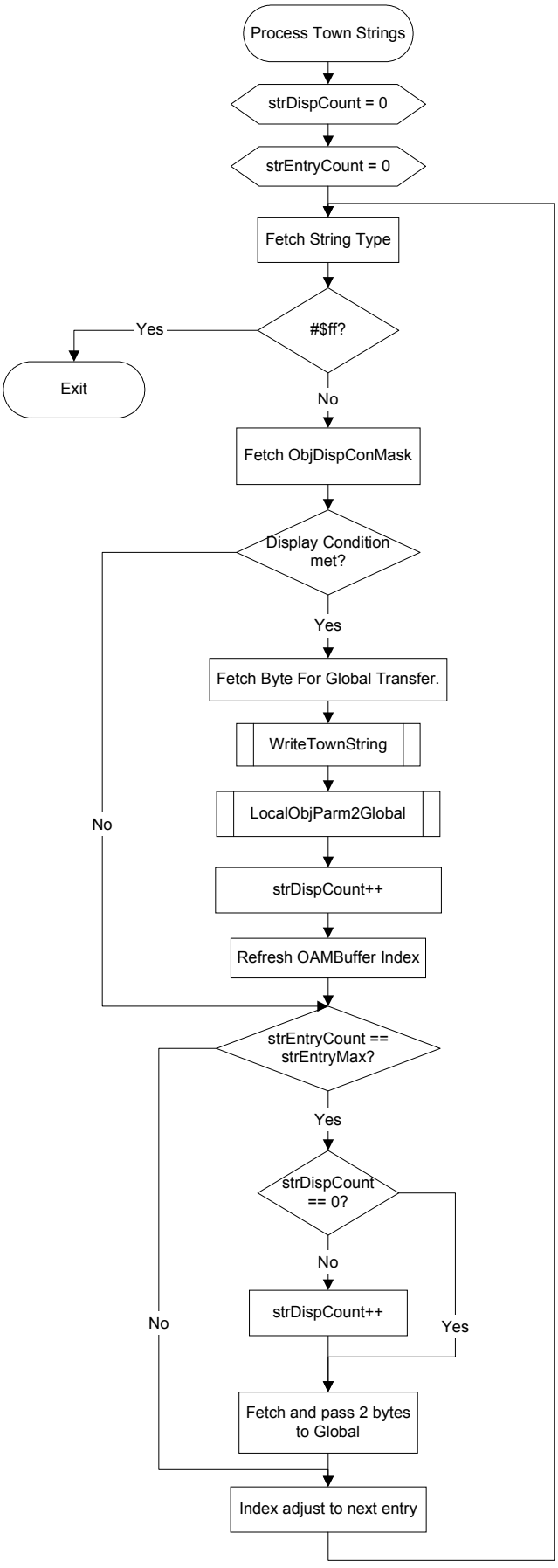
There are 16 display maps used in Romancing Saga1. On each of those, several town strings can be displayed. The RS1 code uses a screen Pointer to point to a set of Town Blocks (these contain the string, Horizontal and Vertical positions, etc). The last block in the list is denoted by the 'End of Block' element, contained in the Town Block, being equal to 0xff. Below are C Structures and visual representations of the Memory.

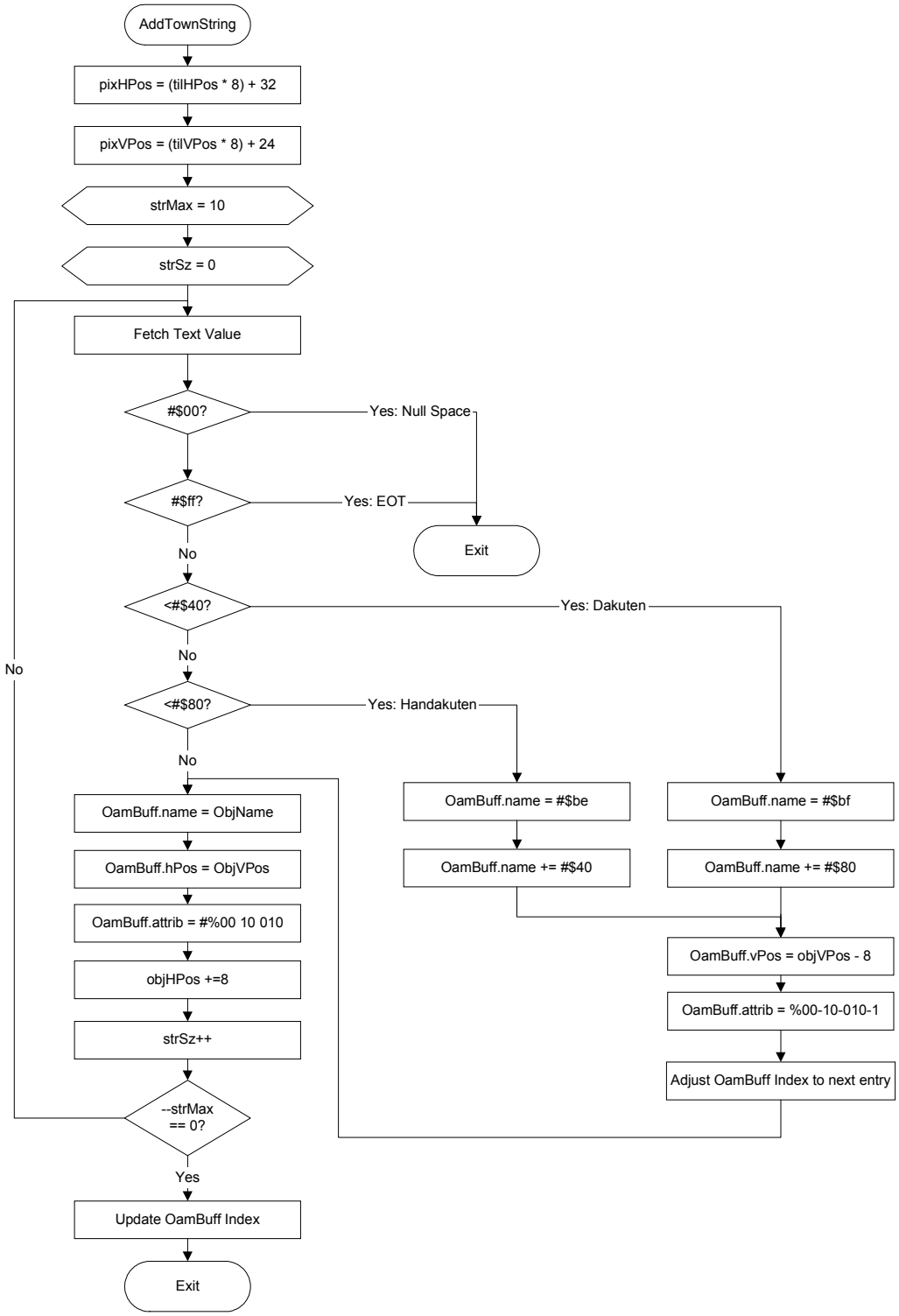


Here are some C-Language representations of the above structures:

```
struct scrPTab_s {              // --- Screen Pointer Table ---
      WORD ptr[16];             // 16 pointers to Town Text Blocks.
} scrPTab;

struct townBlock_s {            // --- 16-byte Town Block ---
/* 0x00 */     BYTE eob;       // 'End Of Block' marker.
/* 0x01 */     BYTE cond;      // Display Condition mask (Whether text is displayed or not).
/* 0x02 */     BYTE tHPos;     // Tile Horizontal Position.
/* 0x03 */     BYTE tVPos;     // Tile Vertical Position.
/* 0x04 */     BYTE str[10];   // Town String (max of 10 chars).
/* 0x0e */     BYTE glob1;     // Global data #1.
/* 0x0f */    BYTE glob2;      // Global data #2.
} townBlock[65];
```

Flowcharts


Process Town Strings
strDispCount = 0
strEntryCount = 0
Fetch String Type
#$ff?
Yes
Exit
No
Fetch ObjDispConMask
Display Condition met?
Yes
Fetch Byte For Global Transfer.
WriteTownString
LocalObjParm2Global
No
strDispCount++
Refresh OAMBuffer Index
strEntryCount == strEntryMax?
Yes
strDispCount == 0?
No
strDispCount++
No
Yes
Fetch and pass 2 bytes to Global
Index adjust to next entry


AddTownString
pixHPos = (tilHPos * 8) + 32
pixVPos = (tilVPos * 8) + 24
strMax = 10
strSz = 0
Fetch Text Value
#$00?
Yes: Null Space
#$ff?
Yes: EOT
Exit
No
<#$40?
Yes: Dakuten
No
<#$80?
Yes: Handakuten
No
No
OamBuff.name = ObjName
OamBuff.hPos = ObjVPos
OamBuff.attrib = #%00 10 010
objHPos +=8
strSz++
--strMax
== 0?
Yes
Update OamBuff Index
Exit
OamBuff.name = #$be
OamBuff.name += #$40
OamBuff.name = #$bf
OamBuff.name += #$80
OamBuff.vPos = objVPos - 8
OamBuff.attrib = %00-10-010-1
Adjust OamBuff Index to next entry

Disassembly

```
; === Subroutine: ProcessMapText =========
00ecc5 ad 81 16      LDA $1681        ; Fetch Map number.
00ecc8 c2 20         REP #$20         ; Accum (16 bit)
00ecca 29 ff 00      AND #$00ff       ; Mask off extraneous data.
00eccd 0a            ASL              ; Mult by 2 for word access.
00ecce aa            TAX              ; Use as index.
00eccf bf 40 80 19   LDA $198040,X    ; Fetch this value.
00ecd3 aa            TAX              ; Use as index.
00ecd4 e2 20         SEP #$20         ; Accum (8 bit)
00ecd6 64 34         STZ $34          ; Init this loc. (clear 'strDispCount'?)
00ecd8 64 35         STZ $35          ; Init this loc. (clear 'strEntryCount'?)

00ecda bf 00 00 19   LDA $190000,X    ; Fetch Text value.
00ecde c9 ff         CMP #$ff         ; EOT?
00ece0 f0 4d         BEQ $ed2f        ; Branch if 'yes'.
00ece2 20 a4 c7      JSR $c7a4        ; Fetch Nibble from RAM Table.

00ece5 3f 01 00 19   AND $190001,X    ; Mask with this Table value.
00ece9 f0 19         BEQ $ed04        ; Branch if not matched (skip display).
; --- Condition met: Display The Text:
00eceb c2 20         REP #$20         ; Accum (16 bit)
00eced bf 0e 00 19   LDA $19000e,X    ; Fetch this value.
00ecf1 a8            TAY              ; Xfer to Y (passed to Global data in func:$ed33)
00ecf2 e2 20         SEP #$20         ; Accum (8 bit)
00ecf4 da            PHX              ; Preserve X Index.
00ecf5 5a            PHY              ; Preserve Y.
00ecf6 20 5a ed      JSR $ed5a        ; WriteTownString
00ecf9 7a            PLY              ; Restore Y.
00ecfa 20 33 ed      JSR $ed33        ; LocalObjParm2Global
00ecfd e6 34         INC $34          ; Increment 'strDispCount', maybe?
00ecff a6 3c         LDX $3c          ; Temp OAMBuffer Index (set by
func:WriteTownString).
00ed01 86 31         STX $31          ; Update OAMBuffer Index.
00ed03 fa            PLX              ; Restore X Index.
; ---
00ed04 a5 35         LDA $35          ; Load strEntryCount?
00ed06 c5 36         CMP $36          ; Compare to this. (strEntryMax?)
00ed08 d0 17         BNE $ed21        ; Branch if not a match.

; -- end of table reach: pass stuff to global.... err, I'm not sure!?
00ed0a a5 34         LDA $34          ; $35==$36: Fetch this value. (load strDispCount)
00ed0c f0 01         BEQ $ed0f        ; Skip if no Strings displayed.
00ed0e 3a            DEC              ; !=0: Decrement value.
00ed0f 85 36         STA $36          ; Store here.
00ed11 bf 0e 00 19   LDA $19000e,X    ; Fetch this value.
00ed15 8f 50 35 7f   STA $7f3550      ; Store in Global loc.
00ed19 bf 0f 00 19   LDA $19000f,X    ; Fetch this value.
00ed1d 8f 51 35 7f   STA $7f3551      ; Store in Global loc.
; --- strEntry Index adjust and loop back.
00ed21 c2 20         REP #$20         ; Accum (16 bit)
00ed23 8a            TXA              ; Transfer X Index to accum.
00ed24 18            CLC              ; Clear carry for addition.
00ed25 69 10 00      ADC #$0010       ; Offset index by +16 bytes to next entry.
00ed28 aa            TAX              ; Update X Index.
00ed29 e2 20         SEP #$20         ; Accum (8 bit)
00ed2b e6 35         INC $35          ; Increment running count?
00ed2d 80 ab         BRA $ecda        ; Loop back.

;--- here on $190000,x = #$ff
00ed2f 20 33 ed      JSR $ed33        ; Parameter passing to memory.
00ed32 60            RTS              ; Return.
; ======== End of WriteMapText =========




; ===== Function: WriteTownString ========
; --- Generate string offset.
00ed5a bf 02 00 19   LDA $190002,X    ; Fetch TilHPos.
00ed5e 85 38         STA $38          ; Store here.
00ed60 0a            ASL              ; Mult by 8.
00ed61 0a            ASL              ; ...
00ed62 0a            ASL              ; ...
00ed63 18            CLC              ; Clear carry for addition.
```

```
00ed64 69 20         ADC #$20        ; Offset intermediate ObjHPos by +32 pixels.
00ed66 85 1c         STA $1c         ; Store as ObjHPos.
00ed68 bf 03 00 19   LDA $190003,X   ; Fetch this value.
00ed6c 85 39         STA $39         ; Store here.
00ed6e 0a            ASL             ; Mult by 8.
00ed6f 0a            ASL             ; ...
00ed70 0a            ASL             ; ...
00ed71 18            CLC             ; Clear carry for addition.
00ed72 69 18         ADC #$18        ; Add 24.
00ed74 85 1d         STA $1d         ; Store here.
; --- Adjust TownString Index.
00ed76 e8            INX             ; Adjust X index by 4 to access next entry.
00ed77 e8            INX             ; ...
00ed78 e8            INX             ; ...
00ed79 e8            INX             ; ...
; --- Setup Parms.
00ed7a ac 31 12      LDY $1231       ; Load OAMBuffer index.
00ed7d a9 0a         LDA #$0a        ; Setup count of 10.
00ed7f 8d 1f 12      STA $121f       ; Store in count hold.
00ed82 64 3a         STZ $3a         ; Zero this value.
; --- Parse Fetched value.
00ed84 bf 00 00 19   LDA $190000,X   ; Fetch this value.
00ed88 f0 31         BEQ $edbb       ; 0     : Branch out if 'yes'.
00ed8a c9 ff         CMP #$ff        ; Check against #$ff.
00ed8c f0 2d         BEQ $edbb       ; #$ff  : Branch out if 'yes'.
00ed8e c9 40         CMP #$40        ; Check against #$40.
00ed90 90 2c         BCC $edbe       ; <#$40 : Branch if 'yes'.
00ed92 c9 80         CMP #$80        ; Check against #$80.
00ed94 90 4c         BCC $ede2       ; <#$80 : Branch if 'yes'.
; --- Standard character (>#$80 && != #$ff):
00ed96 99 02 00      STA $0002,Y     ; Default: Store as ObjName.
00ed99 a5 1c         LDA $1c         ; Fetch.
00ed9b 99 00 00      STA $0000,Y     ; Store ObjHPos.
00ed9e a5 1d         LDA $1d         ; Fetch.
00eda0 99 01 00      STA $0001,Y     ; Store ObjVPos.
00eda3 a9 25         LDA #$25        ; %00-10-010-1
00eda5 99 03 00      STA $0003,Y     ; Store ObjAttrib (Flip,Prio,Pal,TNameMSB)
00eda8 c8            INY             ; Adjust by 4 to next OAM entry.
00eda9 c8            INY             ; ...
00edaa c8            INY             ; ...
00edab c8            INY             ; ...
00edac a5 1c         LDA $1c         ; Load ObjHPos.
00edae 18            CLC             ; Clear Carry for addition.
00edaf 69 08         ADC #$08        ; Offset HPos by +8 Pixels.
00edb1 8d 1c 12      STA $121c       ; Update ObjHPos.
00edb4 e8            INX             ; Increment X Index.
00edb5 e6 3a         INC $3a         ; Increment this running counter?
00edb7 c6 1f         DEC $1f         ; Decrement max entry count?
00edb9 d0 c9         BNE $ed84       ; Loop back if not done for all.
00edbb 84 3c         STY $3c         ; Store in temp OAMBuffer index.
; --- EOT character (val == #$ff):
00edbd 60            RTS             ; Return.

;--- Place Dakuten Character (val <#$40):
00edbe 48            PHA             ; Preserve current ObjName.
00edbf a9 bf         LDA #$bf        ; Load fixed ObjName for Dakuten Character.
00edc1 99 02 00      STA $0002,Y     ; Store ObjName.
00edc4 68            PLA             ; Restore fetched ObjName.
00edc5 18            CLC             ; Clear carry for addition.
00edc6 69 80         ADC #$80        ; Offset ObjName value by -#$80 bytes.

00edc8 48            PHA             ; Preserve adjusted ObjName.
00edc9 a5 1c         LDA $1c         ; Load ObjHPos.
00edcb 99 00 00      STA $0000,Y     ; Store ObjHPos.
00edce a5 1d         LDA $1d         ; Load ObjVPos.
00edd0 38            SEC             ; Set Carry for subtraction.
00edd1 e9 08         SBC #$08        ; Offset by -8 Pixels.
00edd3 99 01 00      STA $0001,Y     ; Store ObjVPos.
00edd6 a9 25         LDA #$25        ; %00-10-010-1
00edd8 99 03 00      STA $0003,Y     ; Store ObjAttrib (Flip,Prio,Pal,TNameMSB)
00eddb 68            PLA             ; Restore accum.
00eddc c8            INY             ; Adjust OamBuffer index by 4 bytes to next
entry.
00eddd c8            INY             ; ...
00edde c8            INY             ; ...
00eddf c8            INY             ; ...
00ede0 80 b4         BRA $ed96       ; Loop back.
```

```
;--- Place Handakuten Character (val >#$40 && <#$80):
00ede2 48            PHA              ; Preserve fetched ObjName.
00ede3 a9 be         LDA #$be         ; Load fixed ObjName for Handakuten Character.
00ede5 99 02 00      STA $0002,Y      ; Store ObjName.
00ede8 68            PLA              ; Restore fetched ObjName.
00ede9 18            CLC              ; Clear carry for addition.
00edea 69 40         ADC #$40         ; Offset ObjName value by -#$40 bytes.
00edec 80 da         BRA $edc8        ; Rejoin code to build rest of OamData.
; ======== End of WriteTownString ======
```

Modification to Town Text

General Functions

```
; ==== Function: LocalObjParm2Global ===
; $34 used:  Y,$31,$38,$3a
00ed33 da            PHX              ; Preserve X index.
00ed34 a5 34         LDA $34          ; Fetch this value.
00ed36 c2 20         REP #$20         ; Accum (16 bit)
00ed38 29 ff 00      AND #$00ff       ; Mask off extraneous data.
00ed3b 0a            ASL              ; Mult by 8 for table access.
00ed3c 0a            ASL              ; ...
00ed3d 0a            ASL              ; ...
00ed3e aa            TAX              ; Use as index.
00ed3f 98            TYA              ; Pass Y index to accum.
00ed40 9f 52 2b 7f   STA $7f2b52,X    ; Store here.
00ed44 a5 31         LDA $31          ; Fetch this value. (OamBuffer Index).
00ed46 9f 50 2b 7f   STA $7f2b50,X    ; Store here.
00ed4a a5 38         LDA $38          ; Fetch this value.  (TilHPos/TilVPos).
00ed4c 9f 54 2b 7f   STA $7f2b54,X    ; Store here.
00ed50 e2 20         SEP #$20         ; Accum (8 bit)
00ed52 a5 3a         LDA $3a          ; Fetch this value. (ObjEntryCount).
00ed54 9f 56 2b 7f   STA $7f2b56,X    ; Store here.
00ed58 fa            PLX              ; Restore X index.
00ed59 60            RTS              ; Return.
; ====== End of LocalObjParm2Global ====
```

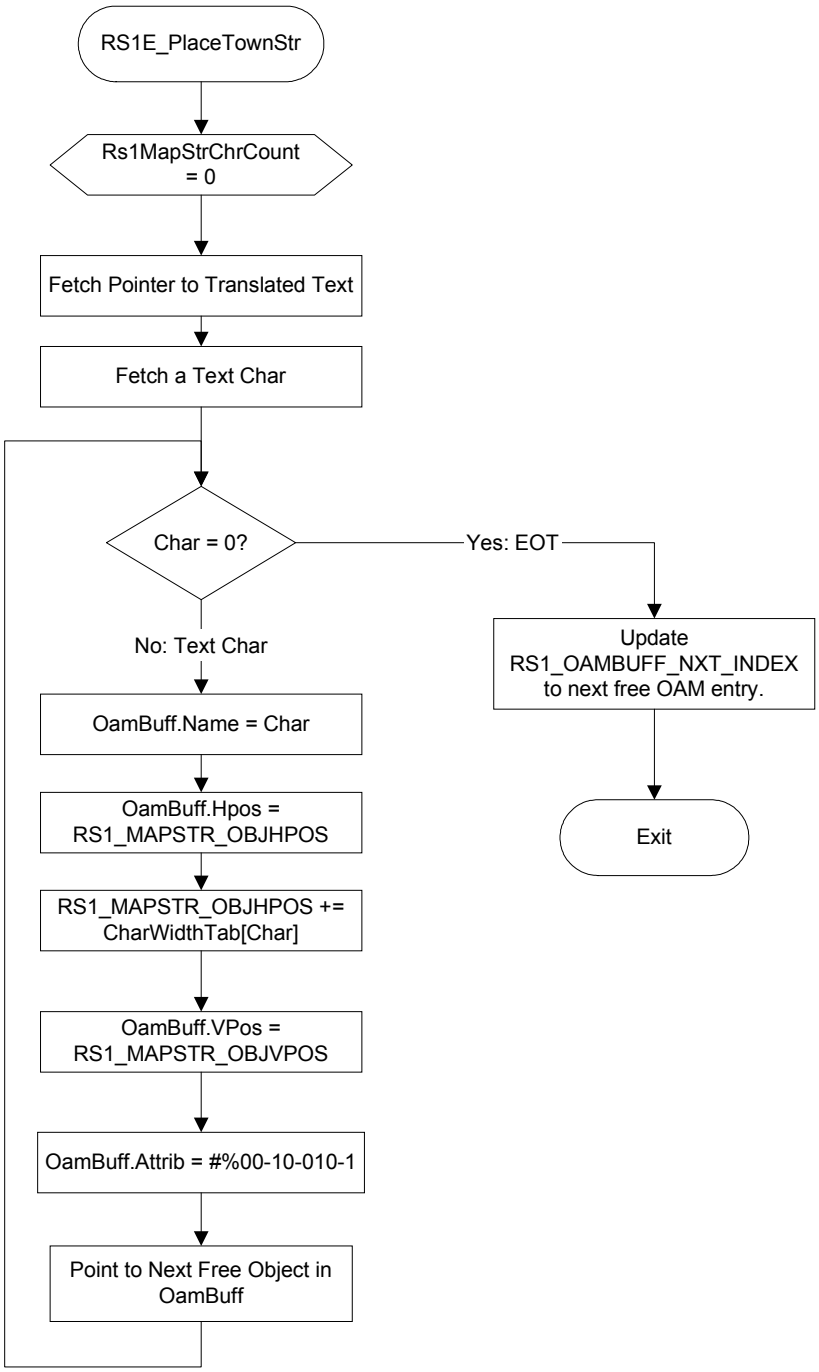
```
; ===== Subroutine:  InitOAMBuffer =====
00edee a0 80 00      LDY #$0080       ; Load Y with 128 Obj Count.
00edf1 a2 00 00      LDX #$0000       ; Init X Index with 0.
00edf4 9e 00 1a      STZ $1a00,X      ; ObjHPos = 0.
00edf7 e8            INX              ; Increment X Index.
00edf8 a9 f8         LDA #$f8         ; Set YPos to offscreen (refer to S-NES Dev'
Manual).
00edfa 9d 00 1a      STA $1a00,X      ; ObjVPos = #$f8.
00edfd e8            INX              ; Increment X Index.
00edfe 9e 00 1a      STZ $1a00,X      ; ObjName = 0.
00ee01 e8            INX              ; Increment X Index.
00ee02 9e 00 1a      STZ $1a00,X      ; ObjAttrib = 00-00-000-0
00ee05 e8            INX              ; Increment X Index.
00ee06 88            DEY              ; Decrement Obj Entry count.
00ee07 d0 eb         BNE $edf4        ; Branch back if not done for all.
;--- Init OAM2Tab (Sz/H-MSB)
00ee09 a9 20         LDA #$20         ; Setup count of 32.
00ee0b 9e 00 1a      STZ $1a00,X      ; Sz=small, H-MSB=0.
00ee0e e8            INX              ; Adjust to Next entry.
00ee0f 3a            DEC              ; Decrment count
00ee10 d0 f9         BNE $ee0b        ; Loop back if not done for all.
00ee12 60            RTS              ; Return.
;======= End of InitOAMBuffer ==========
```

Replacement Map Routine

Commented Assembly

```
;RS1E_BuildTownString
; Note: my code is written in Western Design's syntax:
; |$xxxx   (16-bit address)
; >$xxxxxx (24-bit address)
; <$xx     ( 8-bit address)
; ================= Equates ==========================
RS1_DP_OBJVAR                 EQU           $1200
RS1_OAMBUFF_INDEX             EQU           $1231
RS1_MAPSTR_CHRCOUNT           EQU           $123A
RS1_OAMBUFF_NXT_INDEX         EQU           $123C
RS1_MAPSTR_MAXSZ              EQU           $121F
RS1_MAPSTR_OBJHPOS            EQU           $121C
RS1_MAPSTR_OBJVPOS            EQU           $121D
RS1E_MTOWN_STRMAX             EQU           $18
RS1E_MTOWN_TEXTBASE           EQU           $210000

; ============== Assumes ============================
; DP: RS1_DP_OBJVAR
; DB: $00
; M: 8-bit
; X: 16-bit
; ORG: $00ed76
; ==================================================

        ldy     RS1_OAMBUFF_INDEX             ; Load Y with Object Index.
        stz     RS1_MAPSTR_CHRCOUNT           ; Init Character Cout to 0.
@l1
        rep     #$21                          ; Accum to 16-bit
        lda     >$190004,x                    ; Fetch RS1E's Pointer to Text.
        Tax                                   ; Transfer to X Index.
        sep     #$20                          ; Accum to 8-bit.
@l2
        lda     >RS1E_MTOWN_TEXTBASE,x        ; Load Accum with text char.
        beq     @l3                           ; Branch if 'End ot Text'.
; --- Here if we have a text character
        sta     |$0002,y                      ; Store as current Object Name.
        phx                                   ; Preserve X.
        rep     #$21                          ; Accum to 16-bit.
        and     #$00ff                        ; Mask of extraneous data.
        tax                                   ; Use as index to char width table.
        sep     #$20                          ; Accum back to 8-bit.
        lda     RS1_MAPSTR_OBJHPOS            ; Load accum with Object HPosition.
        sta     |$0000,y                      ; Store as current Object Hpos.
        adc     RS1E_MTown_ChrWidthTab-$80,x  ; Offset Hpos by current Character Width.
        sta     RS1_MAPSTR_OBJHPOS            ; Update Object Hposition.
        plx                                   ; Restore X Index.

        lda     RS1_MAPSTR_OBJVPOS            ; Load Accum with Object Vposition.
        sta     |$0001,y                      ; Store as current Objen Vpos.
        lda     #%00100101                    ; 00-10-010-1 (Flip,Prio,Pal,TNameMSB)
        sta     |$0003,y                      ; Store as current Object Attrib
        iny                                   ; Adjust Y Index to next Object Element.
        iny                                   ; ...
        iny                                   ; ...
        iny                                   ; ...
        inx                                   ; Adjust X index to next text character.
        inc     RS1_MAPSTR_CHRCOUNT           ; Increment string's Character Count.
        bra     @l2                           ; Loop back.
; --- Here upon 'End of Text'.
@l3

        sty     RS1_OAMBUFF_NXT_INDEX         ; Store Index to next free Object.
        rts                                   ; Return.
RS1E_MTown_ChrWidthTab
        INCBIN "widthout.bin"                 ; Include our Character Width Table.
```

Flowchart


RS1E_PlaceTownStr
Rs1MapStrChrCount = 0
Fetch Pointer to Translated Text
Fetch a Text Char
Char = 0?
Yes: EOT
No: Text Char
Update RS1_OAMBUFF_NXT_INDEX to next free OAM entry.
Exit
OamBuff.Name = Char
OamBuff.Hpos = RS1_MAPSTR_OBJHPOS
RS1_MAPSTR_OBJHPOS += CharWidthTab[Char]
OamBuff.VPos = RS1_MAPSTR_OBJVPOS
OamBuff.Attrib = #%00-10-010-1
Point to Next Free Object in OamBuff