Chobits Translation Notes:

Todo:

Update reserved ranges in script.json file

Make program to translate text boxes

Consider redoing the DoubleShift-RIS to preserve dakuten and handakuten

Would help with the text box

Or replace them with relevant symbols

Write re-inserter for multiple TIM2 images

Translate text in TIM images

Figure out video extraction

Done:

Create encoding table for DoubleShift-RIS

Create half-width English font

Make a program to create combinations of English font and overwrite the game’s font

Figure out why game crashes (hideki looking at ceiling textbox, second one)

Make a program to extract the text boxes

Make program to re-insert translated text boxes

Make text look better

Last boxes at:

1061803 – 2852664 (1033AB - 2B8738)

May want to review character names and search for them:

<https://ja.wikipedia.org/wiki/%E3%81%A1%E3%82%87%E3%81%B3%E3%81%A3%E3%83%84>

1DFFAB0 – may be a missed ‘yes’ and ‘no’ response just before this

May be some informational text from 1E69B0 - 1E7E40

Debugging:

Write TIM2 inserter, test これ dazzling

Tokyo... Its the big city,\nhuh? Back home was all grass\nand cows, so it was pretty lax.

8cc5 91e9 9354 9746 9841 94ad 9866 90e7 9854 9163 9855 9170 9657 0A

90f7 90ff 97fa 8ecf 91f0 90f1 9289 9869 8edf 9853 9256 9859 93e7 9473 0A

8eda 8fda 8f8f 959b 96d0 946f 985b 94ba 9589 9480 9384 9048 94b3 985e 8ee4 974a?

Input data is missing half its first char (1 byte)

Output data is missing one full byte

Crashing notes:

Hideki is looking at the ceiling, second text box

A screenshot of a video game

AI-generated content may be incorrect.

If going fast enough, nothing bad happens

If not going fast enough, the game slows, then crashes

Associated with these types of errors in PCSX2 log:

[24038.1348] TLB Miss, pc=0x13897c addr=0x6abea194 [load] …

[24038.2832] TLB Miss, pc=0x138658 addr=0x2000000 [store]…

Does it happen in the original game?

Does it happen in an unmodified but recompiled version of the game?

Is A003 still modified?

YES – try fixing then loading from savestate and see if that avoids it

Actually, don’t load from savestate—the bad file would probably still be in mem (do after work, not time rn)

Once fixed it stops crashing

Reverse engineering allowed character ranges from kanjiRomAddrOriginal function:

Characters range [character offset] (calculated end) number of chars:

8140 – 81F1 [-40] (calc:81F1), 177

Save: none

Original offset: 6932F

824f – 82f1 [+71] (calc:8300), 163

Save: 824f – 8258 (numbers), English chars, 829f – 82f1 (hiragana)

Save all for 163

8340 – 83d6 [+123] (calc:83D7), 151

Save: 8340 – 8396 (katakana), 87

8440 – 84be [+1ba] (calc:84bf), 125, Russian A, \_|\_ then + , has an overlap

Save: 8440 – 845d (PS2 buttons and Let me be with you), 30

~~None start with 85 or 86 (agrees with table)~~

~~8740 – 879c [+239] (calc:873E), -3 ???~~

889f – 88ff [+237], 61

Save: none

8940 – 89ff [+2f7] , 192, kanji, dot (I think these have no overlap)

Save: 897E is 円, replace with ¥ and skip, 1

8a40 – 8aff [+3b7] , 192, kanji, dot (all end in 3 dots)

8b40 – 8bff [+477] , 192, kanji, dot

8c40 – 8cff [+537] , 192, kanji, dot

8d40 – 8dff [+5f7] , 192, kanji, dot

8e40 – 8eff [+6b7] , 192, kanji, dot

8f40 – 8fff [+777] , 192, kanji, dot

9040 – 90ff [+837] , 192, kanji, dot

9140 – 91ff [+8f7] , 192, kanji, dot

9240 – 92ff [+9b7] , 192, kanji, dot

9340 – 93ff [+a77] , 192, kanji, dot

9440 – 94ff [+b37] , 192, kanji, dot

9540 – 95ff [+bf7] , 192, kanji, dot

9640 – 96ff [+cb7], 192, kanji, dot

9740 – 97ff [+d77], 192, kanji, dot

9840 – 9872 [+e37] (test:9872), 51 (can go up to 73 before hitting code, but maybe don’t)

~3,600 characters in total

3,249 are needed to host all my char combos

Can keep 351 in total

Keeping 275 in total, figure out ranges to preserve

Figure out how to write to script and output to table at same time

SJIS value range:

Est: 0x8100 – 0x9800 (33024-38912) ~= 5888

Sjis.tbl starts at 8140, ends at EAA2

98FC is the latest starting in 98

Are both blank, need to reign in

Lowest: 8141, A starts at 8260

Highest: 9872

SJIS table file says that’s 3512 rows

That can fit 59 characters and change

26 caps, 26 lower = 52, plus 7 characters: . ! ? , “ \_

Plus another 6 characters: . , ! ? “ \_

3364 characters for all combos, 148 extra chars

Is not continuous, identify breaks and see if the code can display them

Breaks: (before/after)

/8141

81AC/824F

82F1/8340

83D6/8440

Font images are in SLES\_\_\_\_\_.55 ELF executable, set tiles to ~~24 x 24~~ and you’ll see them

I think extracted images are **~24wide X 12tall**

Starts at offset of 693BF for the 、character

Must use GBA 8bpp Tile form to see them

GBA 4pp seems clearer, but the text shows up larger

For either form, the text is a bit cut off, may just have to do with offset

Seems like ~3700 characters in total

26 upper, 26 lower, 10 punctuations, all combinations are ~= 3700

Could be a squeeze for some

May want to omit uncommon characters, or have punctuation take up a whole one.

Starts with PsIIlibmc 2540 on line 68BB0

693BF is the proper offset for starting at A, B, C…

4bpp = 4 bits per pixel?

Seems to produce OK results, but there are some emptyish spaces

Each character is 144 bytes long, use increments of that to select the next one

24 pixels wide, 12 pixels tall

fontReplacer.py can successfully change the letter A into a black box

No spaces by default, I’ll need to include them in my font

Need to draw each character

Programmatically create a combo of all important ones

Insert them into the ELF file

\*\*\*Need to determine exact start and end of the character table

Experiment with the highest and lowest allowed numbers

Then determine how many combos I can make and which are important

Look at game text to see what it uses

After that I’ll need to make a script for translating

Maybe have it dump to a separate text (json?) with the original text, and a length maximum in bytes

<https://www.w3schools.com/python/python_json.asp>

Then re-insert the translated text

Adding (zeroes? Something to produce blank characters) at the end if it goes on for longer

<https://www.deepl.com/en/translator>

Can this be automated as well? Maybe paste the translation in the same JSON and flag it if it is too long, and how many characters too long it is

After enabling texture dumping in PCSX2, I was able to get some text boxes out

Had to modify the .xml files Rainbow produced for another image to get it to re-import it back in, specifically adding \_0 to one of the file name fields and changing the height and width fields

Then I re imported it and saved it as a TIM2 named first\_textbox.tm2

Try to find the following hex string that it contained in memory somewhere:

01 03 06 06 03 01

ls

Doesn’t show up anywhere in memory

Maybe in grep somehow?

grep ./ -ra -e “text\_in\_binary\_not\_hex”

A001, A002, A003, A004, A005, A006, A007, A008, A009, A010, A011, A020, A021

9363 = 田

Folders to check:

A900, A920, A940, A980

Executable file – EE990 has some repetitive text, maybe images?

Find Kanji images/bitmap in game files?

Try zeroing out large chunks of A003 and see if it does anything

File A003 has a very large prefix, may contain more data

It also produces a blank image file

Looks mostly random in the tile viewer, maybe its still compressed?

So far I think I’ve looked through all of them and haven’t found it

Is it in the PS2 itself somehow?

May also be a table with bits, perhaps I could rewrite the bit length for English characters?

A005 extraction is different than the rest

Are binary file matches for TIM in:

A920/A93F – try these, may be original tim files not tim2

A920/A931

A920/A900

A900/A911

SLPM\_652.55 – is just code

MODULES/IOPRP254.IMG – both are substrings of TIME

seems like coincidence, they’re just big files, murphy’s law and all that

Ghidra decompiler

font\_vramClear

font\_vramTrans

font\_vramWrite

font\_vramWriteOne

**fontInit()**

\_name\_data\_addr = LoadFile(0x10,0,0,0); - from a different function

1E7470 = 0x10?

uVar1 = LoadFile(0,0,0,0);

??? = 0 – possible font location

LoadFile(0,0,0,0) – search for other load files and see if there’s a pattern

LoadFile is also referenced in joint\_tm2\_cps\_tarns and staff\_disp\_init

In both, only the first parameter is used (e.g., a variable or 0x10)

Called param\_1 in function def.

Passed to:

file\_name\_set as first parameter

First parameter is param\_1

Is &-ed to ffffffffffffffe0, so only last 5 bits are read

Passed to sprint as third parameter, along with @492/3/4

cps2bin as first parameter

memcpy as second parameter, but voided

sceGsExecLoadImage(0x412f40, uVar1) – search for other load images and see if there’s a pattern

Is ‘uVar1’ global or local?

fontsys()

fontsys\_readold()

strncasecmp

strlen

\_nptr

atoi

fontsysExe()

\_kanji\_f

Memset(&fontDisp,0,0x3600)

font\_vramWrite(\_kana\_adr)

FontseigyoExecf

Gofont()

convPacketPrio()

kanjiRomAdrOriginal() – function that may be related to text locations

Has several switch case statements matching hex values like 0x81, 0x82, … 0x98

Corresponds with the first byte of the Kanji (and probably Roman) mappings in the sjis.tbl file

Is mentioned in the stock\_font\_one() function

Three functions seeming to refer to shift-jis:

sjis\_cursor\_move

sjis\_enter

Has variables:

input\_sjis\_code

select\_moji\_index

moji\_command – moji means letter or character in Jap.

ten\_maru\_data

ten maru could be relevant to fixed-length encodings: <https://japanese.stackexchange.com/questions/17390/on-%E5%8E%9F%E7%A8%BF%E7%94%A8%E7%B4%99-when-are-%E3%82%9B-and-%E3%82%9C-ten-ten-and-maru-supposed-to-occupy-a-square-of-thei>

ten-ten is the name of “, maru is the name of **˚**

sjis\_rolling\_check

select\_pos\_set

select\_moji\_set

includes strcpy() command (string copy?), is fed &input\_sjis\_code parameter

& is a reference to the object

Memcpy(…NAMEIN\_MOJI\_OBJ…)

font\_vramWriteOne() is called in a do-while loop

obj\_selmoji\_back\_on()

font\_vramTrans()

**scenario\_printf()**

vsprintf()

**\_kanji\_f = 5; (kanji file?) – file A005 had no TIMs, this may be it**

**\_kana\_adr = &DispStr; (kana address?)**

scenario\_printf\_chk

return \_kanji\_f != 0

scenario\_seq – seems to play ‘scenarios’ (scenes?)

May be related to the ‘S\_01\_0\_0\_\_1…’ style things in the executable

obj\_speech\_window\_status()

get\_hideki\_abcde() – grabs player’s name?

cname\_play\_status()

obj\_speech\_window\_on()

name\_line\_on

speech\_window\_pos

**scenario\_printf(\_kana\_adr,…)**

**\_kana\_addr = (char \*)byte2sjis(0x412fa0);**

**\_kanji\_f = \_scenario\_disp\_mode**

scenario\_chk\_kanji

obj\_speech\_window

convPacketPrio()

obj\_speech\_window\_on

obj\_speech\_window\_status

comobj\_sets

obj\_selmoji\_back\_on

obj\_selmoji\_back

convPacketPrio

ng\_word\_check

name\_moji\_convert

ignore\_moji\_check

get\_ng\_wordn()

strlen

strncmp

mprintf

vsprintf

mp\_matrix

Have program to extract images

After looking through all images files I have not found a Kanji Table

(A001?) may be a roman table but it’s the wrong font for ingame

~~A005 is in the same folder as all the images but doesn’t contain images—maybe the text is in there?~~

Not obvious if so

It has images they just were organized differently, still not the font files

Maybe try to GREP in another folder?

Is it built in to the PS2? If so, why wouldn’t it have support for the half-width characters?

May want to make it a function to make it more automated to scan all files in folder

A003 is mostly prefix and shows up blank in rainbow

Decompressor errors on the following files: A002, A00F, A00A, A001, A020

A020 gets most of the way through and errors out on #324

First mod I made to decompressor borked the output

Could be related to file extractor

Try to make a sanity check that re-inserts the files, see if it’s the same as the original file data

I put in an if statement to avoid this, but it will drop the last byte in the event of that occurrence, maybe look into keeping that

Use the following to find TIM files: grep ./ -r -e "TIM2"

Created bad compressor for testing

Sanity check with quick recompressing succeeds

Game boots even with poorly recompressed images

Game still boots with edited images from rainbow

Still need the 3- or 4-byte prefixes for the images afaik

Created decompressor, works on Broccoli logo

Didn’t work at first I think because there are bits at the end that were necessary, which I didn’t copy from the memory of the game

Reverse engineered compression

It uses a strange encoding scheme—the 2-byte instructions should be read as binary, where the first 6 bits are 29 more than the number of bytes to copy, and the second 10 bits are one less than the number of bytes to count back

Details in the hexExploration.txt file

Haven’t verified if its byte-for-byte yet but eyeballing it it looks pretty close

Current thread:

Replace all Kanji with pairs of 2 english letters, (e.g., ab, ba, ac, ca, etc.) and reference those by address

Would fix the spacing problem and memory size problem

Would need to find the sprites for the kanji and replace them

Probably would have to automate them

~4000 combinations exist (including capital and lowercase, and punctuation)

This would not fix the FMV cutscenes, those would still need to be decompressed and have subtitles added, then re-compressed and re-inserted

Have a script to grab TIM2 files, testing to see if I can load any in Rainbow

I found some that Rainbow tries to open, but it calls them ‘mipmaps’

The header looks more like what Racoon Sam said were compressed files, maybe I need to decompress them

After reviewing what he said, it sounds like only one image is uncompressed in memory at a time

That’s what I found, the first image was extracted and I was able to copy/paste that block of memory into a .tm2 file and it opens in Rainbow

Next steps: extract uncompressed broccoli image and compare to file, see if I can reverse engineer the compression algorithm like Raccon Sam and automate the extraction of the TIM2 files

Working on decompiling ELF file with Ghidra

Ultimate goal is to find where it references the data related to the text and see if I can point it elsewhere

Make sure to select experimental aggressive option

Ghidra addresses are 00100000 above other hex editors

This turned out to be a lot more difficult than I thought

I’m not sure how to re-compile it once its been decompiled, if that’s even possible

Do more research on that—best case scenario, I can add data back in without a problem

Can unpack .iso with Apache 2

Can repack with IMGburn

May not produce perfect files, but whatever

Text is in the SLPM\_652.55 file, not compressed

Data block above the text starts at 001024C0

??? ps2dis and wxMEdit show different hex value data here

Even in the top pane where ps2dis should jst be showing the raw hex

Actual script starts at 001032D0, ends at 00195EF0

Options text located at 001CA930

001E69B0 has what I think are names

Several places with text tables:

00101C40 – UTF-16 LE makes this legible in kanji

001E8680

See if adding certain amounts works (e.g., 4 pairs of hex values)

Did not work—removing an empty line still broke it

Seems like offset isn’t the issue, it must be memory/file addresses

Currently trying to edit SLMP with English text

From the Shift JIS table, it looks like Japanese text (kanji and kana) are written in 4 hex digits (2 bytes), where English characters are written in 2 hex digits (1 byte)

Actually there may be ‘kana’ English characters written in 4 hex digits, I will have to experiment with whether the game engine can work with just 2

The 2-digit text files do not show up, it just appears empty

SIJS.tbl has the corresponding hex values needed for the text, but opening in Notepad it doesn’t know the encoding for the Kanji

May have to find the table in memory/SLMP file that contained all the characters alphabetically

WindHex doesn’t seem the easiest program to edit with

Currently trying ~~Madedit~~ Use wxMEdit

001CF050 has a full table of kana in SLMP file

001E1DD0 has a full table of kana and the letter ‘C’ in kana

O=82 6E

P=82 6F

S=82 87

C=82 62

Remaining characters are documented in roman\_kana\_hex\_table.xlsx

0A seems to be a newline

Only 3 lines can fit

00 80 00 seems to have special meaning, like triggering a new scene

Half-width characters only apply to katakana

17 full-width characters comfortably fit in the box

Boxes have 3 lines for text

19 characters are fully visible, 20th is mostly visible, any more is offscreen

51 safe characters, 60 generous

Boxes have 3 lines for text

Testing some translations using deepl translator: <https://www.deepl.com/en/translator>

Full width character converter: <https://dencode.com/en/string/character-width>

Consider using a thesaurus to shorten words, and remove unneeded padding

東京か…やっぱ都会だよなぁ。

オレの田舎なんか、緑と牛しか

なくてまったりしてたんだけど,

12345678901234567890 – monospace font

Ｔｏｋｙｏ－Ｒｅａｌｌｙ　ｉｓ　ｂｉｇ．

Ｈｏｍｅ　ｏｎｌｙ　ｈａｄ　ｇｒａｓｓ

ａｎｄ　ｃｏｗｓ，　ｓｏ　ｍｅｌｌｏｗ．

- full width can be pasted

Tokyo, huh... It really is the big city, isn't it?

Back in my hometown, there was nothing but greenery and cows,

and it was so laid-back, but...

都会はなんちゅーか華やかで、

いろいろと目にうつるものが,

12345678901234567890

The city is radiant,

all sorts of things

catch your eye.

Try to automate this process next—extract text and replace it in the file

At first just replace with ‘text box 1, text box 2’, etc.

Also try changing the file size

Pasting some text in seemed to grow the file size by just a bit, and now the game won’t boot. It may verify file size or depend on addresses in some way

Putting data at the very end doesn’t seem to break anything

May need to copy the text data table to the end and change the part in the ELF file where it is addressed

Worst case scenario would be it is addressed multiple places and even partway through the script, that would probably require decomplication to fix

Actually I think looking at the assembly the way I did \*is\* considered decompilation, but my ps2dis disassembler just isn’t very good

May have to read up on editing ELF files

Perhaps if I only add in complete lines it will work

Maybe try decompiling first, then editing the text, then recompiling?

Consider decompiling with: <https://github.com/chaoticgd/ghidra-emotionengine-reloaded>

May be able to compare edited code with: <https://decomp.me/new>

Another thing I could try is changing the text encoding to store English characters in 2-digits

Or add the existing shift-jis English back in somehow

Allow some text to roll over into the next box when there’s space?

F6BF0 – table of Kanji?

I’ve heard Ghidra is a good one, may be add-ins that let you do MIPS

MW MIPS C Compiler listed in code

Also noticed the term ‘PSIIlibgp 2530’ just above the text

Cutting/pasting:

Try to cut/paste the text to the end of the file and see if it still boots

PSIIlibgp – 1024C0 (contains text)

Empty line after the whole text is at 1C9210

Cut between these two

I made a python script for this, but it doesn’t seem to have worked

The game boots, but doesn’t hit the start screen if the full script is extracted

If only the first box is extracted, it bugs out and jumps to the first interactive scene with Chii

Try deleting as much as possible from the file and see if it still boots

Where is the data actually needed?

Tried that, here’s what I found:

Last try: 1C9210 – Does not boot

Earliest successful: 1E9A80

Old notes tools:

Intro to romhacking: <https://www.romhacking.net/start/>

PS2 romhacking guides: <https://www.romhacking.net/?page=documents&category=&platform=18&game=&author=&title=&perpage=20&docsearch=Go>

PS2 decryptor that works with PCSX2: <https://forums.pcsx2.net/Thread-PCSX2dis-v1-1-A-ps2dis-inspired-PCSX2-enabled-Game-Hacking-Tool-W-I-P-13-02-2015>

PS2DIS tutorials:

<https://gamehacking.org/faqs/1UpPS2Dis.html>

<https://www.consolediscussions.com/forum/forum65/>

<https://www.oocities.org/siliconvalley/station/8269/ps2dis/>

Consider learning PS2 MIPS (the assembly language): <https://theelitegamers.darkbb.com/t47-learning-mips-to-use-ps2dis>

Chiidake was made by Broccoli Games, maybe trying other games will help

List of games they’ve made: <https://www.giantbomb.com/broccoli-co-ltd/3010-6390/published/>

None of their other early games have fan translations either

Mention of another game being translated: <https://forum.evageeks.org/thread/21839/Fan-Translation-of-Evangelion-2-Shinseiki-Evangelion-2-on-PS2-PSP/>

Shinseiki Evangelion: Koutetsu no Girlfriend 2

(could be the psp version, I haven’t found evidence of the PS2 version being translated)

The PSP version has a translation: <https://github.com/tehmugi/patches_release>

This seems like a dead end, the PSP games are laid out entirely differently, and made by a different company

May want to use Quickbms: <https://github.com/LittleBigBug/QuickBMS>

Can extract text, is aware of obfuscation

Might need to find relevant .bms scripts to use

List of all scripts: <https://aluigi.altervista.org/quickbms.htm>

May want to try corrupting files and loading it into PCSX2 to see what it does

Also should probably practice re-assembling .iso files to make sure I can do that at all

Use ps2dis program to read the hex/binary files

Go to View > Kanji code set to decode kanji

Ctrl + G to get labels

I need to figure out where to ‘load from’ for these

See if you can find where it references the files

One of the bin files contains “encoded by TMPGEnc (ver 2.57.41.14E)

A920 folder > A938

From the Resident Evil Code Veronica translation guide:

May need to use another hex editor that allows hex tables (they translate hex values to readable characters) such as WindHex

Text translation hacking tutorial: <https://www.romhacking.net/documents/919/>

Downloaded into PS2\_translation\_tutorial.txt file in project folder

Results from exploring contents of files (with WindHex):

Root >

SLPM\_652.55 – can’t be read with WindHex, but can be read with ps2dis

An executable file?

SYSTEM.CNF – can be read with notepad, mostly contextual stuff

((All other folders))

MODULES >

EZMIDI.IRX - .ELF contained in first few characters, midi.driver

I think .elf files are executables

IOPRP254.IMG – something related to I/O

LIBSD.IRX - .ELF , Sound.Device.Library

... – other hardware libraries/executables I think

Files starting with A### seem to count up starting from the folder name in hexadecimal

Windows doesn’t organize them in order by default, but Apache does

A220>, A242>, A1A0>, A1C0>, A1E0>, A100>, A120>, A140>, A160>, A180>, A0A0>, A0C0>, A0E0>, A060>, A080>, ((audio files?))

Many of the files in here have ‘IECS….’ At the beginning of lines

Ex: A1C7 contains IECSsreV, IECSuqeS, IECSqseS

Ex: IECSigaVP, IECSdaeH@, IECSbteS

Googling some of them yields this forum post: <https://hcs64.com/mboard/forum.php?showthread=26379>

Seems to be related to sound files for the PS2

.HD/.BD files? Used to store sound or other media?

Not all are identical, so some might be different kinds of files

Perhaps these are headers for the following files?

Said files are also shorter than the rest

All the smaller files (1-5KB) are like this

The larger files seem to have many random characters, some of which can be converted into kanji/kana, but don’t seem to form coherent sentences

Larger files all end in 0077777777777777 (..wwwwwwwwwwwwww)

Some are empty at the beginning

Is it compressed or encrypted?

There’s more …’s than in a compressed file I looked at, but its hard to tell

A900>,A920>, A940 >, A020>, A040> ((FMVs?))

All files in these folders have ‘encoded by TMPGEnc (ver. 2.510.49.157)’

MPEG streams are mentioned by guy below—these may be FMVs

Suggests it is a video format or similar

A000> ((Background sprites?))

Has large files, but the small files often do not contain ‘IECS…’

Some do

Large files do ***not***end in 0077777777777777

Large files have [E@.TIM2](mailto:E@.TIM2) somewhere at the beginning

Relative searching:

Searching Chii, Chi, or ちぃ has not yet been fruitful

ちぃ is coded as 82BF for ち and 82A1 for ぃ (chisai/small ぃ)

Have not had luck with this yet

MIPS R5900 is the cpu type

Decompiler may help: [https://github.com/Decompollaborate/spimdisasm#](https://github.com/Decompollaborate/spimdisasm)

Current best guess as to the code I should be running:

python3 spimdisasm singleFileDisasm --instr-category r5900 ./SLPM\_652.55.txt ./SLPM\_652.55.Decomp

Must be run inside the first ‘spimdisasm’ folder in repos in WSL

I haven’t had any luck with this yet

File formats:

<https://www.ps2-home.com/forum/page/formats_and_working_tools_chart>

Found some discussion online, the game folder is compressed:

<https://archive.vg-resource.com/thread-33484.html>

Someone on here is writing a decompressor, will look into

“Yeeaaaaah the files are standard TIM2 files, packed with a typical LZish compression. <=$7F are raw bytes, anything above that is a compression word (2 bytes) where 6 bits are for amount and 10 bits for distance.”

The maker of the program is the owner of this Twitter: <https://x.com/vervalkon>

<https://www.instagram.com/vervalkon/>

I sent him a message on Instagram, we’ll see if he gets back to me

<https://www.youtube.com/channel/UCgyrTxYpBaB1Dahpz94rOEg>

<https://www.linkedin.com/in/vervalkon/?originalSubdomain=fi>

Sounds like he started by comparing the contents of savestates in PCSX2 to the contents of the game files, let’s try this with text and see where it leads

After dumping the first text box to a save state and extracting it, I found the beginning of the script at line 002032D0

In WindHex editor using the sjis encoding format

It goes on for a *very* long time, I think the game actually loads the entire script into memory at the very start

Entire script between 002032C0 and 01151A10

First textbox between 002032C0 and 00203330

In extracted text file, ends in 000C5EA0

Hex values for the first text box are:

2C 00 7C 11 29 00 04 00 20 10 01 00 00 11 31 00

01 20 00 93 8C 8B 9E 82 A9 81 63 82 E2 82 C1 82

CF 93 73 89 EF 82 BE 82 E6 82 C8 82 9F 81 42 0A

83 49 83 8C 82 CC 93 63 8E C9 82 C8 82 F1 82 A9

81 41 97 CE 82 C6 8B 8D 82 B5 82 A9 0A 82 C8 82

AD 82 C4 82 DC 82 C1 82 BD 82 E8 82 B5 82 C4 82

BD 82 F1 82 BE 82 AF 82 C7 81 41 0A 00 15 02 11

31

HxD hex editor lets you copy/paste easier, try using that for now

Does not have support for SJIS encoding

I’m currently going through and comparing it to entire\_script.bin in HxD, see where this leads

Script is ~700kb when uncompressed, ~200-250kb when compressed

A000>A014 has lots of wwwwww and 777777’s in it, but is 241kb

Also very dense, maybe a possible script file

I may try comparing the uncompressed text to the disc files and see if there are similarities, and keep an ear out for a response from mr script writer

May want to use WindHex to search for ƒIƒŒ (O-RE) from the first line of text in the game

Bookmark: A004

Currently sorting files into folders based on content

Bookmark: A040

There is a pattern in the files: (may want to re-extract from ISO to put it back this way)

The count in hex so windows sorting is sometimes confusing

Take folder A100 for example:

The 1st file is A100, and contains IECSsreV, IECSdaeH@, and IECSigaV

The 2nd file is A101, and contains IECSsreV, IECSuqeS, and IECSqseS@

Sounds like these are related to sound waves

The 3rd file is A102, and much larger and more densely packed

The first row is all 00’s

The first column is usually a number/hex digit

The second column is usually a 00 or a 02

The final row is 00 77 77 77 …. (or wwwwwwww)

Sometimes there will be a row of 00’s followed by a row of 77’s in the middle of the file

Then file A103 is just like A100, so the pattern repeats

It seems as if most folders follow this pattern

Sometimes the folder starts mid-pattern, but that may be because it continues the pattern of a previous folder

Text is in the SLPM\_652.55 file, not compressed, around 00106D70

Encoded in typical Shift-JIS, wxMEdit can view

Will need to extract