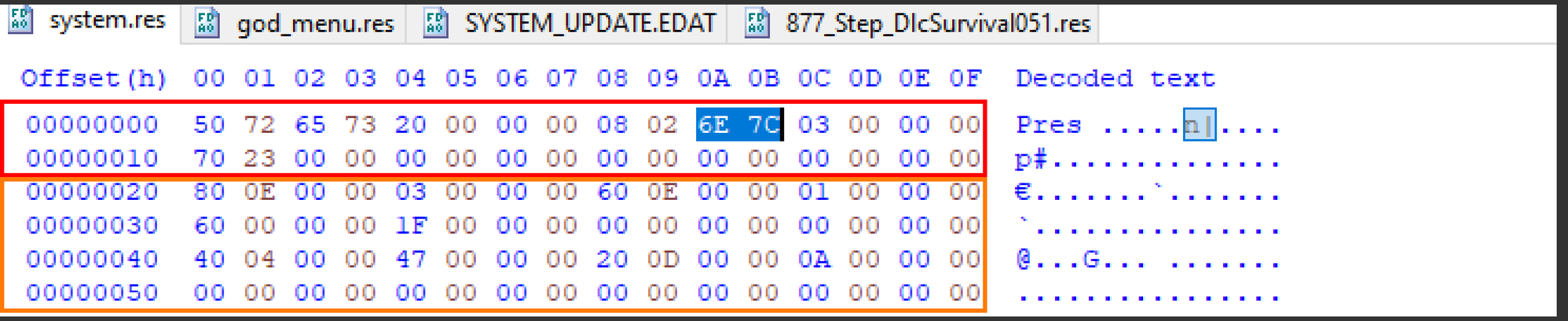


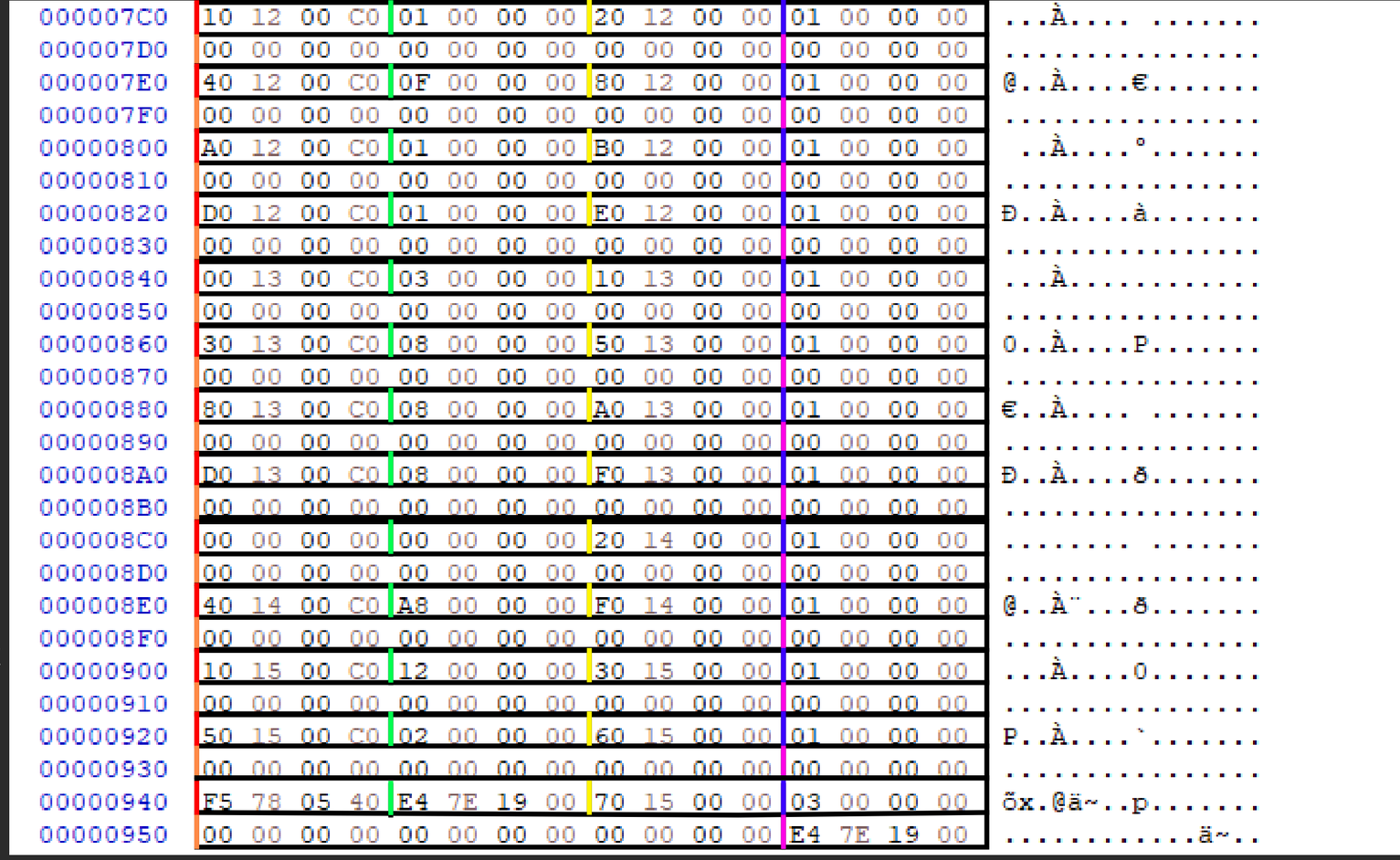
[Start Point]
~ 0x00: Magic/Header: 4 bytes (Pres)
~ 0x04: 4 bytes, little endian offset (0x20). jumps to table offset
~ 0x08: byte, group table count. (always set to **08**)
~ 0x09: some kind of value?. changes on different RES archives.
~ 0x0A: 2 bytes checksum?. not sure what to do with it. (but i guess it could be a a MD5 hash due to **libmd5.prx** being used)
~ 0x0C: 4 bytes, seems to be stuck at 3, must be a version of that RES archive?
~ 0x10: 4 bytes little endian offset, jumps you to where certain chunks of data are stored (related to ToC)
~ 0x14->0x1F: lengthy zeroes, 4 bytes each. not really useful. but can be a table offset out of nowhere so keep an eye out.

[Table Offsets] ranging: *0x20-0x5F*
~ 0x20: empty usually on some other .res archive, but it represent as ToC offset (`00 15 00 00` -> 0x1500 for example)
~ 0x24: empty usually on some other .res archive, but it represent as count for ToC (`03 00 00 00` -> 3 for example)
- rinse and repeat for others like 0x28 as ToC offset and 0x2C as ToC count.
- Table Offsets always uses 16 bytes. be creative when reading them.

[Table of Contents] (ToC) [32 bytes in size]

~ File Offset: a little endian offset. brings you to a area where a offset chunk is stored.
~ Size/Compressed Size: defines the size of that area starting with
File Offset. Size can vary if it's ZLIB compressed or not based on it's header (blz2)
~ Name Offset Table: a little endian offset. brings you to a area where the name is located.
~ Name Element: a name element (that's how i call it). can vary from 1 and 3. (can go 4 and 5 in PSVITA SIDE)
~ Zeroes: bunch of zeroes, length of 12 (0xC)
~ Size (Decompressed): Game's expected decompressed size of that Compressed Size. Value can match with Size if it's not compressed by ZLIB.

[Identifying External File Data]
in 0x940. you see that the 4th byte has `40`. let's call this Address Types. If you encounter any of these values, it means that the file source is external. here's the following lists for these values
~ 40 (0x4) = package.rdp
~ 50 (0x5) = data.rdp
~ 60 (0x6) = patch.rdp
~ C0 (0xC) = Current (no need to go to a external source)
~ D0 (0xD) = Current (no need to go to a external source)
If you encounter any of these. You will need to do multiplications.



000007C0	10	12	00	C0	01	00	00	00	20	12	00	00	01	00	00	00	...À.... .....
000007D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000007E0	40	12	00	C0	0F	00	00	00	80	12	00	00	01	00	00	00	@..À....€.....
000007F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000800	A0	12	00	C0	01	00	00	00	B0	12	00	00	01	00	00	00	..À....°.....
00000810	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000820	D0	12	00	C0	01	00	00	00	E0	12	00	00	01	00	00	00	Ð..À....à.....
00000830	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000840	00	13	00	C0	03	00	00	00	10	13	00	00	01	00	00	00	...À.....
00000850	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000860	30	13	00	C0	08	00	00	00	50	13	00	00	01	00	00	00	0..À....P.....
00000870	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000880	80	13	00	C0	08	00	00	00	A0	13	00	00	01	00	00	00	€..À.... .....
00000890	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000008A0	D0	13	00	C0	08	00	00	00	F0	13	00	00	01	00	00	00	Ð..À....ð.....
000008B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000008C0	00	00	00	00	00	00	00	00	20	14	00	00	01	00	00	00	....
000008D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000008E0	40	14	00	C0	A8	00	00	00	F0	14	00	00	01	00	00	00	@..À¨...ð.....
000008F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000900	10	15	00	C0	12	00	00	00	30	15	00	00	01	00	00	00	...À....0.....
00000910	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000920	50	15	00	C0	02	00	00	00	60	15	00	00	01	00	00	00	P..À....`.....
00000930	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000940	F5	78	05	40	E4	7E	19	00	70	15	00	00	03	00	00	00	õx.@ä~..p.....
00000950	00	00	00	00	00	00	00	00	00	00	00	00	E4	7E	19	00	.....ä~..

Obtaining Absolute Offset (for external files)
~ Reassign 0x940's File Offset into Big Endian order, you'll get something like this: "400578F5"/"0x400578F5".
~ Remove or Change "4" to "0" and multiply the new offset by 800: "400578F5" * "800" = "2BC7A800"/"0x2BC7A800"
~ The true/absolute offset is "2BC7A800"

***.rtbl files have this structures but spreads out of nowhere with lengthy paddings keep an eye out.**

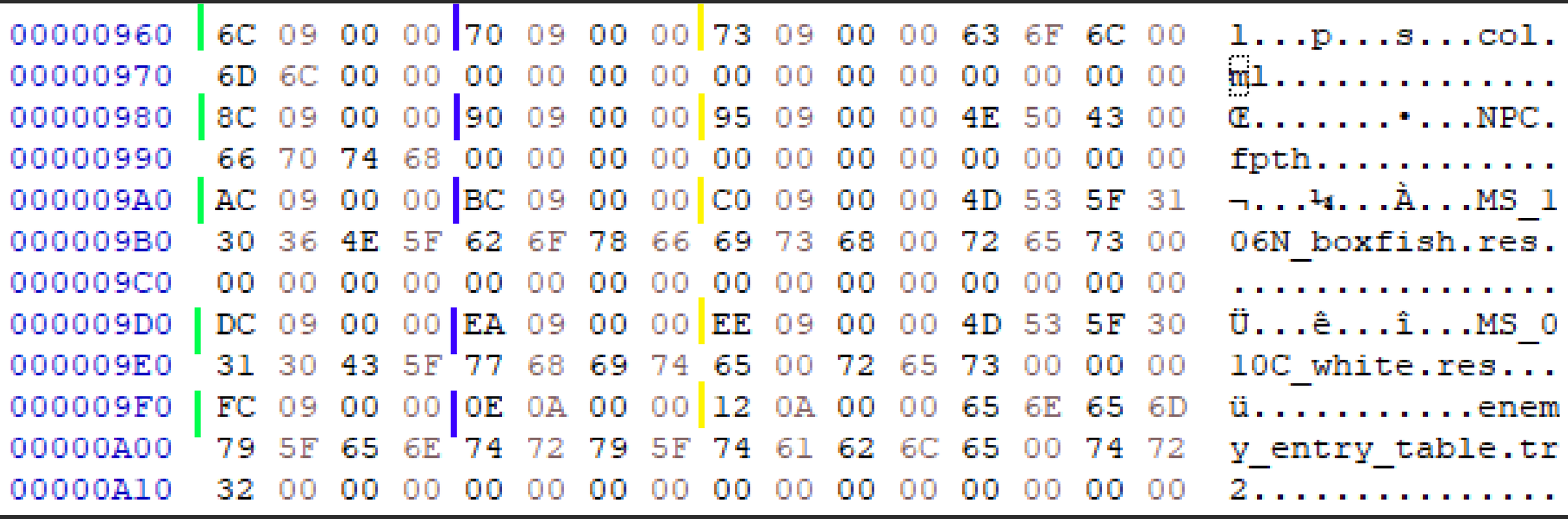
[Name Tables] (PSP SIDE)

Note: If Name Element is less than three (sometimes it could be 1). it will rely only on Name Offset. There's only **1**, and **3** value on the psp side, so if its **3**, then there's **3** tables despite not having a number **2** value.

~ 1 = Name Offset: can be treated as UINT16 or UINT32, this is an offset that jumps you to the File Name.

~ 2 = Extension Offset: can be treated as UINT16 or UINT32. this offset jumps you to the File Extension.

~ 3 = Path: an offset that jumps you to the zeroes. but if its non zero, then it is a path (folder).



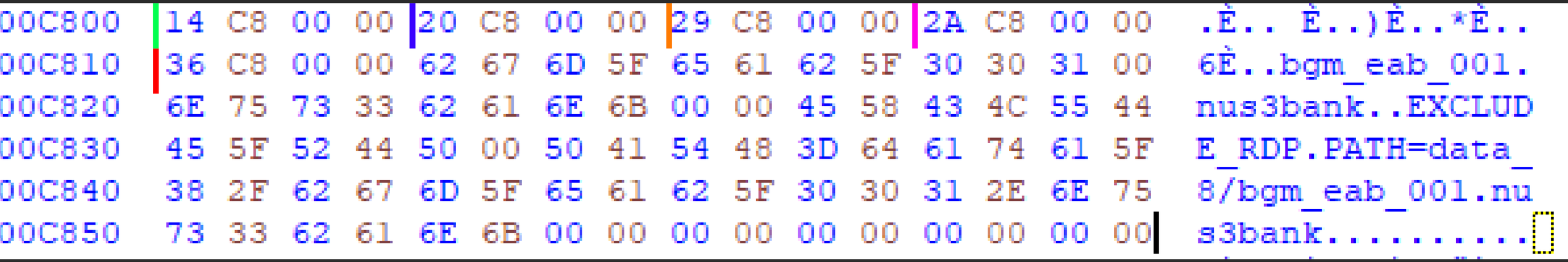
[Name Tables] (PSVITA SIDE)

Note: continuation of Name Tables (PSP Side). this is for the value **4** and **5**. value **3** is treated different here if value is above **3**.

~ 3 = Path: an offset that leads you into a text

~ 4 = Path: an offset that leads you into a text. but with a +1 offset on the previous Path Value

~ 5 = Instruction Directory: an offset that jumps you to a instruction of that file's directory. A exclusion file that is outside of the RDP file



[ZLIB Compression] *BLZ2 only*

0x00: Header 4 bytes. “blz2"

0x04: Compressed file size, 2 bytes.

the rest: compressed data.

Padding: adjustment for another file chunk or compressed data

[decompression sample code (python)]

```
def blz_decompress(data, csize, dsize):
    data = BytesIO(data)
    magic = data.read(4)
    if magic != b"blz2":
        raise ValueError("Data is not in BLZ2 format.")

    decom = b""
    if dsize >= 0xFFFF:
        size = int.from_bytes(data.read(2), "little")
        ekor = zlib.decompress(data.read(size), -15)
        while data.tell() < csize:
            size = int.from_bytes(data.read(2), "little")
            decom += zlib.decompress(data.read(size), -15)
        return decom + ekor
    else:
        size = int.from_bytes(data.read(2), "little")
        decom = zlib.decompress(data.read(size), -15)
        return decom
```