uOFW

AN UNOFFICIAL OFFICIAL FIRMWARE PROJECT

State of PSP Documentation

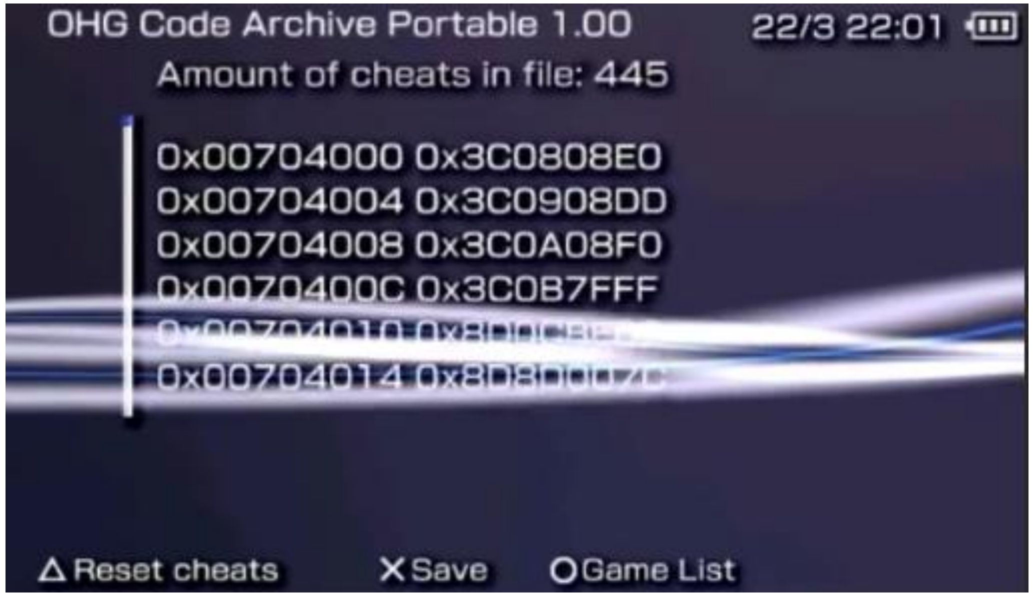
- Scattered across the web
  - Forums (ps2dev, wololo,…)
  - Blogs (Silverspring,…)
  - Websites (PSPTEK, hitmen, pspdevwiki,…)
  - Open-source projects (PPSSPP, JPCSP,…)
- Lost to history
  - Many private blogs and even entire forums dedicated to the PSP are no longer available (i.e. LAN.st)
    - Web archives can only recover a tiny bit of the original research

State of PSP Documentation (2)

- Severely outdated and incomplete hardware documentation
  - Some public documentation exists, i.e. [YAPSPD] (last modified: Sat, 23 Dec 2006)
  - However…really incomplete for the 1st gen of the PSP
    - Hardware registers (hardware/kernel interface)
    - Coprocessor registers
    - specific hardware component documentation (GE, ME, VME, SYSCON, WLAN,…)
    - detailed PSP boot process documentation (hardware & kernel initialization)
  - What about later hardware revisions (new hardware components, changed hardware & kernel communication, improved security,…)?
  - [PSPTEK] is an updated hardware/kernel reference including uOFW research
    - Last modified in 2015 though

State of PSP Documentation (3)

- Some homebrew only released as closed-source → forces other PSP developers to “reinvent the wheel”
  - Example: [VLF] – a powerful library for homebrew developers to use XMB UI elements in their UI
    - Original authors looked into official VSH GUI modules to write the library
    - Research not published



```
OHG Code Archive Portable 1.00                22/3 22:01
          Amount of cheats in file: 445

          0x00704000 0x3C0808E0
          0x00704004 0x3C0908DD
          0x00704008 0x3C0A08F0
          0x0070400C 0x3C0B7FFF
          0x00704010 0x[illegible]
          0x00704014 0x8D8[illegible]
```

→ Enter uOFW

What is uOFW?

- An effort to reverse-engineer the PSP hardware & kernel
- Aims to create an up-to-date hardware/kernel reference
- Provides a free and open-source vanilla firmware

Reverse-engineering the PSP kernel & hardware

- ~130 kernel modules (as of 6.60)
- Not all modules are in scope for uOFW (PlayStation Network modules, PSX emulator,…)
- Focus on core kernel services
  - Kernel initialization
  - System memory manager
  - Program loading & execution
  - Exception handling (interrupts, system calls, exceptions)
  - I/O file manager
  - Power management
  - Clock/Timers
  - …

Reverse-engineering the PSP kernel & hardware (2)

- Hardware wrappers
  - System controller
  - Graphic Engine
  - Media Engine
- Human-machine interfaces
  - Controller service
  - LED service
- Audio

Reverse-engineering the PSP kernel & hardware - Progress

- 35 modules have been worked on so far (complete & incomplete)
- 22 modules have been completed
  - ME/GE/SYSCON wrappers
  - System memory manager
  - Program loader
  - Exception managers
  - I/O file manager
  - Audio
  - Controller service
  - Clockgen, System timer
  - …

What is uOFW?

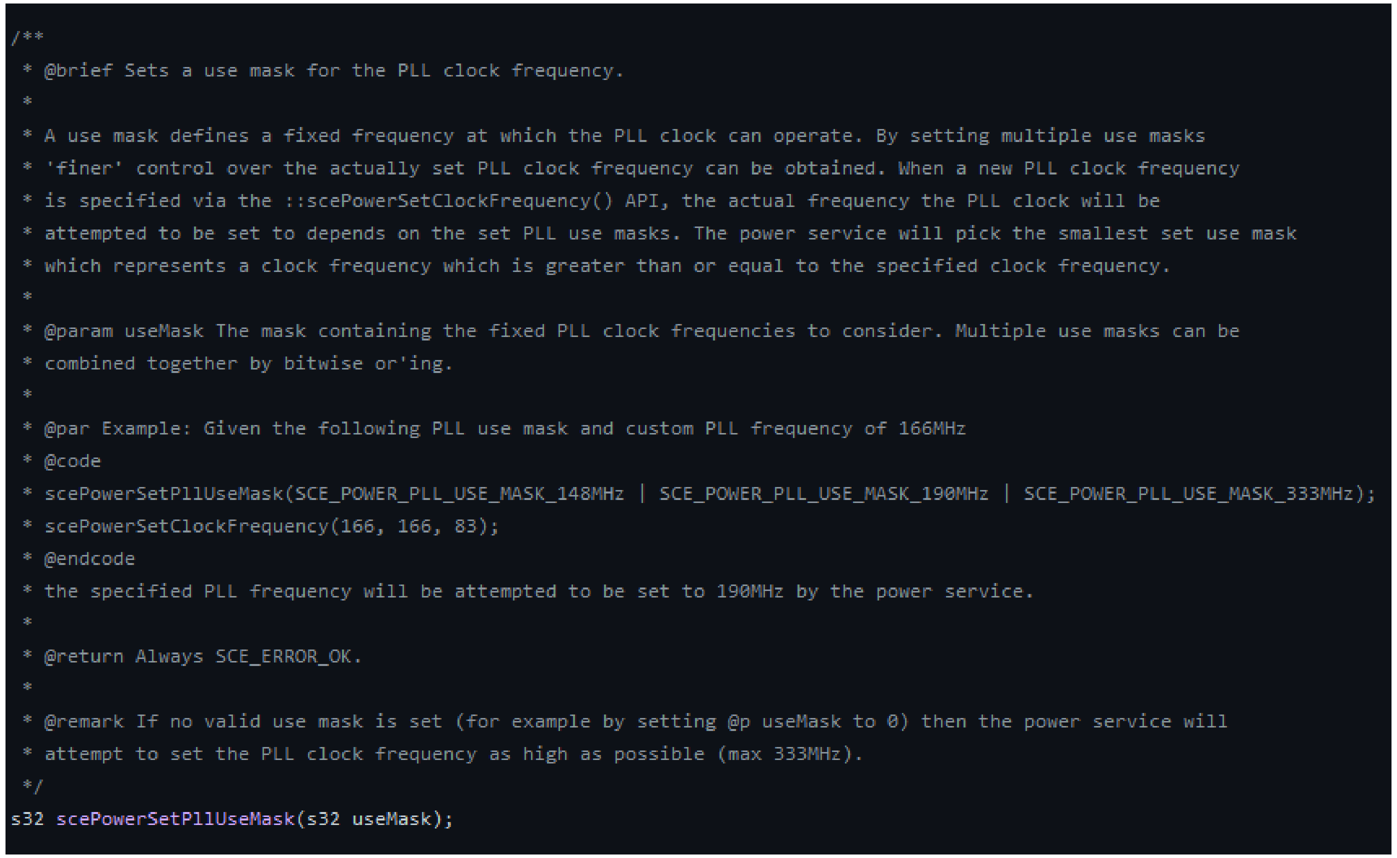
- An effort to reverse-engineer the PSP hardware & kernel
- **Aims to create an up-to-date hardware/kernel reference**
- Provides a free and open-source vanilla firmware

Creating a hardware/kernel reference

- Document public APIs
- Document inner workings of the kernel
- Document hardware/software interface

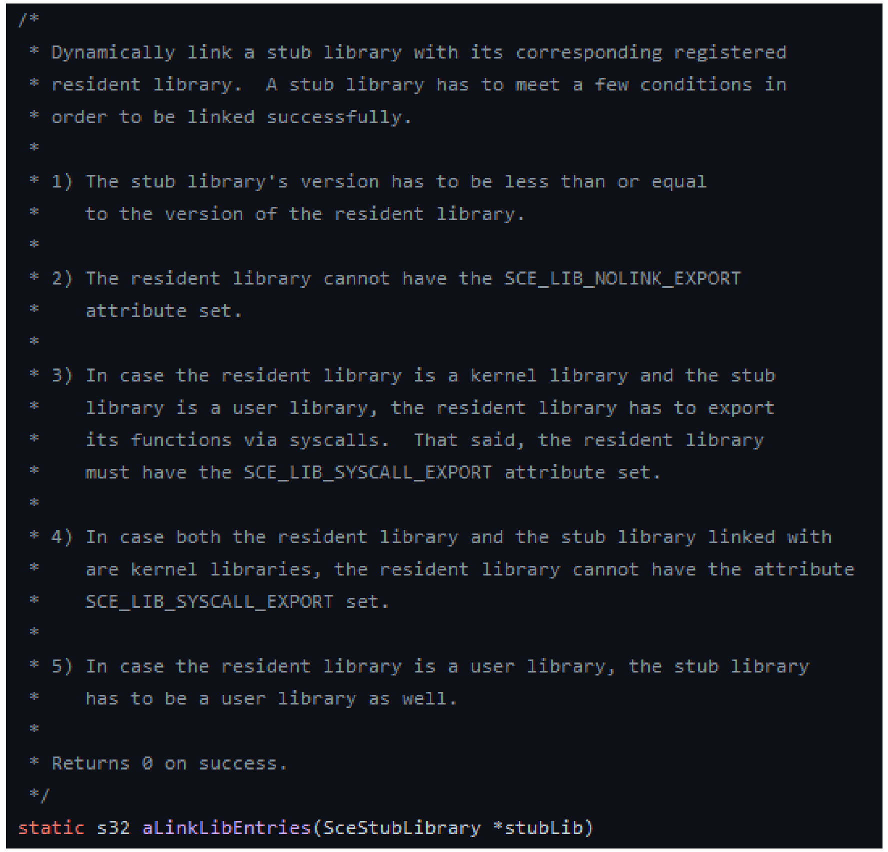
- Goals
  - Central place for PSP documentation
  - Preserve existing findings and document new research

Documenting public APIs



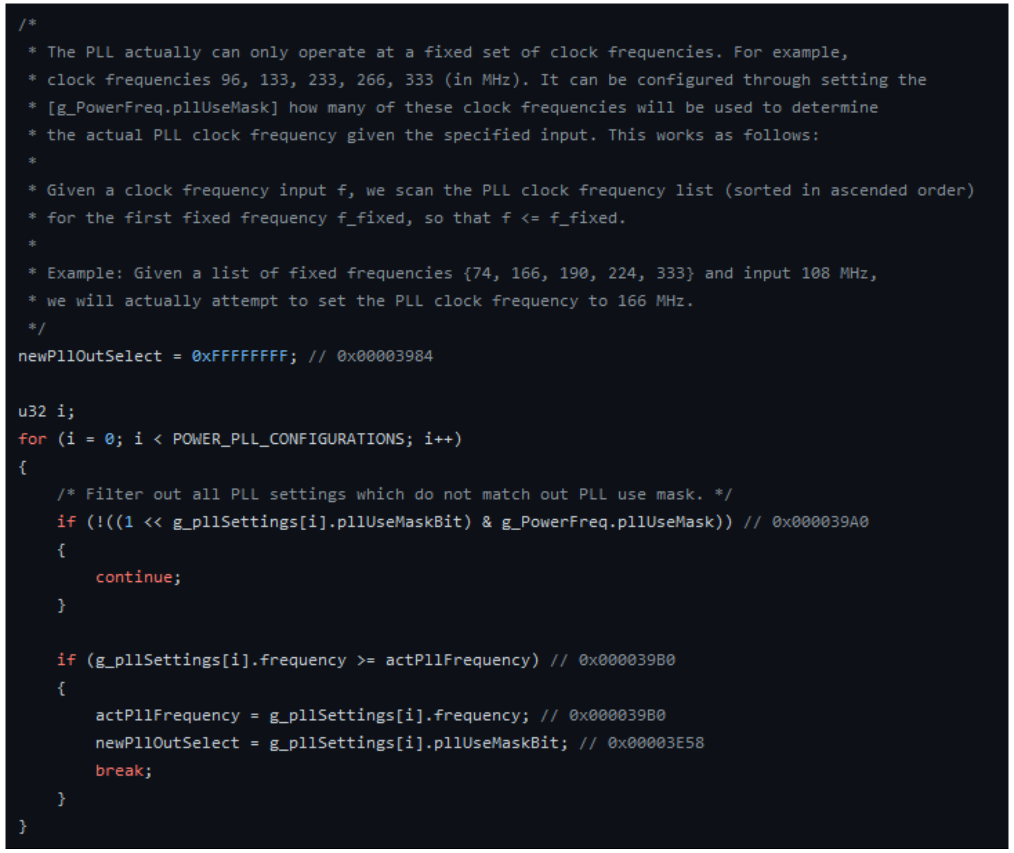
```
/**
 * @brief Sets a use mask for the PLL clock frequency.
 *
 * A use mask defines a fixed frequency at which the PLL clock can operate. By setting multiple use masks
 * 'finer' control over the actually set PLL clock frequency can be obtained. When a new PLL clock frequency
 * is specified via the ::scePowerSetClockFrequency() API, the actual frequency the PLL clock will be
 * attempted to be set to depends on the set PLL use masks. The power service will pick the smallest set use mask
 * which represents a clock frequency which is greater than or equal to the specified clock frequency.
 *
 * @param useMask The mask containing the fixed PLL clock frequencies to consider. Multiple use masks can be
 * combined together by bitwise or'ing.
 *
 * @par Example: Given the following PLL use mask and custom PLL frequency of 166MHz
 * @code
 * scePowerSetPllUseMask(SCE_POWER_PLL_USE_MASK_148MHz | SCE_POWER_PLL_USE_MASK_190MHz | SCE_POWER_PLL_USE_MASK_333MHz);
 * scePowerSetClockFrequency(166, 166, 83);
 * @endcode
 * the specified PLL frequency will be attempted to be set to 190MHz by the power service.
 *
 * @return Always SCE_ERROR_OK.
 *
 * @remark If no valid use mask is set (for example by setting @p useMask to 0) then the power service will
 * attempt to set the PLL clock frequency as high as possible (max 333MHz).
 */
s32 scePowerSetPllUseMask(s32 useMask);
```

Documenting the inner workings of the kernel



```c
/*
 * Dynamically link a stub library with its corresponding registered
 * resident library.  A stub library has to meet a few conditions in
 * order to be linked successfully.
 *
 * 1) The stub library's version has to be less than or equal
 *    to the version of the resident library.
 *
 * 2) The resident library cannot have the SCE_LIB_NOLINK_EXPORT
 *    attribute set.
 *
 * 3) In case the resident library is a kernel library and the stub
 *    library is a user library, the resident library has to export
 *    its functions via syscalls.  That said, the resident library
 *    must have the SCE_LIB_SYSCALL_EXPORT attribute set.
 *
 * 4) In case both the resident library and the stub library linked with
 *    are kernel libraries, the resident library cannot have the attribute
 *    SCE_LIB_SYSCALL_EXPORT set.
 *
 * 5) In case the resident library is a user library, the stub library
 *    has to be a user library as well.
 *
 * Returns 0 on success.
 */
static s32 aLinkLibEntries(SceStubLibrary *stubLib)
```

Documenting the inner workings of the kernel (2)



```
/*
 * The PLL actually can only operate at a fixed set of clock frequencies. For example,
 * clock frequencies 96, 133, 233, 266, 333 (in MHz). It can be configured through setting the
 * [g_PowerFreq.pllUseMask] how many of these clock frequencies will be used to determine
 * the actual PLL clock frequency given the specified input. This works as follows:
 *
 * Given a clock frequency input f, we scan the PLL clock frequency list (sorted in ascended order)
 * for the first fixed frequency f_fixed, so that f <= f_fixed.
 *
 * Example: Given a list of fixed frequencies {74, 166, 190, 224, 333} and input 108 MHz,
 * we will actually attempt to set the PLL clock frequency to 166 MHz.
 */
newPllOutSelect = 0xFFFFFFFF; // 0x00003984

u32 i;
for (i = 0; i < POWER_PLL_CONFIGURATIONS; i++)
{
    /* Filter out all PLL settings which do not match out PLL use mask. */
    if (!((1 << g_pllSettings[i].pllUseMaskBit) & g_PowerFreq.pllUseMask)) // 0x000039A0
    {
        continue;
    }

    if (g_pllSettings[i].frequency >= actPllFrequency) // 0x000039B0
    {
        actPllFrequency = g_pllSettings[i].frequency; // 0x000039B0
        newPllOutSelect = g_pllSettings[i].pllUseMaskBit; // 0x00003E58
        break;
    }
}
```

Documenting hardware/software interface

- Map memory regions which are used to communicate with PSP hardware
  - Memory starting at 0x1C000000 (0xBC000000)
  - Used across critical kernel services to perform operations (ME/GE/SYSCON, System timer, Audio, Power management,…)
  - Used by homebrews like [Melib] which provides access to ME capabilities beyond the officially exposed ones
  - Allows direct configuration of the PSP hardware such as the power on/off states of LEDs, clock speeds,…

  → Will give full control over the PSP hardware – not limited to Sony's kernel API restrictions
- Document various interfaces/APIs used to interact with/offload work to hardware components
  - ME, GE, SYSCON

Documenting hardware/software interface (2)

- Document transformation from hardware design to public APIs
  - PSP controller button mapping – useful to find button combos checked during boot process (such as service mode button combo)
- Research unknown coprocessor registers
  - CP0 (i.e. provides memory management support for the operating system such as cache management)
  - FPU
  - VFPU

What is uOFW?

- An effort to reverse-engineer the PSP hardware & kernel
- Aims to create an up-to-date hardware/kernel reference
- **Provides a free and open-source vanilla firmware**

Provide a free and open-source vanilla firmware

- Ambition: Reverse-engineered modules should be able to replace original Sony modules
  - Good indicator if a module has been reverse-engineered correctly
  - Open-source module implementations can replace proprietary Sony code in community projects
- A challenging task
  - Of the 22 modules completed so far, half of the modules are seemingly running without issues
    - Controller service, SYSCON-, GE- & ME-wrapper, Loadexec, Power, System timer,…
    - Full list available here: https://github.com/uofw/uofw/wiki/Current-Modules-Status
  - Remaining modules have issues to various degrees
    - Crash during boot phase
    - Crash at a later stage
    - Limited functionality

Challenges

- Making mistakes during RE'ing is easy 😑
- Locating mistakes can be hard
  - No elegant debugging tools available to us
  - File & display logging are options, however….
  - Adding to many log statements can in itself cause a crash 😒
  - Some modules (sysmem, loadcore) run before the necessary drivers are initialized
    → Need to fall back to framebuffer debugging
  - Other modules might not crash themselves but can still cause a system crash due to mistakes made

Credit: [devnoname120]

uOFW module overview

Module	Partially completed	Completed	Working (*)	Documented
Audio		✓		✓
Audiocodec	✓			
Chkreg		✓	✓	✓
Chnnslv		✓		
Clockgen		✓	✓	✓
Codec		✓	✓	
Ctrl		✓	✓	✓
DMAC manager	✓			
Exception manager		✓		
GE		✓	✓	

uOFW module overview (2)

Module	Partially completed	Completed	Working (*)	Documented
HTTP Storage	✓			
IDStorage		✓		
Init		✓	✓	✓
Interrupt manager		✓		
I/O File manager		✓		
Led		✓	✓	✓
LibAAC		✓		
LibAtrac3Plus		✓		
Loadcore		✓		✓
LoadExec		✓	✓	✓

uOFW module overview (3)

Module	Partially completed	Completed	Working (*)	Documented
LowIO	✓			
MCCtrl	✓			
ME wrapper		✓	✓	
Media Manager		✓	✓	✓
Memlmd	✓			
Mesgled	✓			
MlnBridge_msapp	✓			
Module Manager		✓		✓
NP Installer	✓			
NP Core	✓			

uOFW module overview (4)

Module	Partially completed	Completed	Working (*)	Documented
OpenPSID	✓			
Power		✓	✓	✓
RTC	✓			
Psheet	✓			
Syscon		✓	✓	✓
Sysmem		✓		✓
Systimer		✓	✓	✓
Usersystemlib		✓		✓
WLAN firmware	✓			

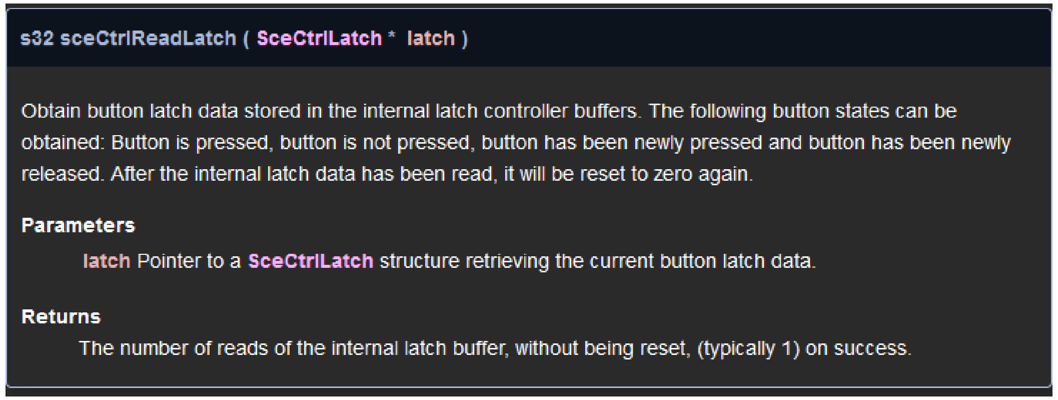
* As far as the modules were tested

uOFW installer

- Runs our uOFW modules
- Based on PRO-CFW (6.60)
- Not yet user-friendly
  - Pro-CFW’s module patches based on original Sony modules still hard-coded
  - Support for some modules (like modulemgr) has to be added manually
- Executable was recently uploaded to the repo
  - https://github.com/uofw/uofwinst/releases

Community use of uOFW

- Improve PSPSDK API documentation



s32 sceCtrlReadLatch (SceCtrlLatch * latch)

Obtain button latch data stored in the internal latch controller buffers. The following button states can be obtained: Button is pressed, button is not pressed, button has been newly pressed and button has been newly released. After the internal latch data has been read, it will be reset to zero again.

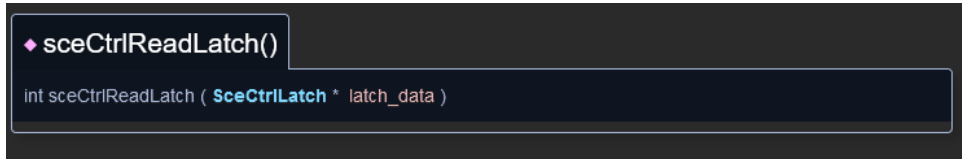
Parameters

latch Pointer to a SceCtrlLatch structure retrieving the current button latch data.

Returns

The number of reads of the internal latch buffer, without being reset, (typically 1) on success.

[uOFW Ctrl]



◆ sceCtrlReadLatch()

int sceCtrlReadLatch (SceCtrlLatch * latch_data)

[PSPSDK Ctrl]

Community use of uOFW (2)

- Act as a reference for leading PSP emulator projects (PPSSPP, JPCSP,…)

“uOFW is a great place to document into details how the PSP is working internally. I've used uOFW quite often as a reference during the implementation of some features in Jpcsp. Additionally it is very helpful to have a clear documentation of not just the good cases, but also under which circumstances errors are being returned by PSP APIs, which is critical for proper emulation in Jpcsp. I'm glad to see that uOFW is being revived.” – Gid15 – Lead developer of JPCSP

- Used by community
  - DS3Remapper
  - [Baryon Sweeper]
  - Relocation type support for PRXTool & Ghidra

Community use of uOFW (3)

- A project [PSV-PSP] which aims to run PSP games on the PS Vita in its native display resolution benefits from the reverse-engineered Graphic Engine wrapper
- Replace proprietary Sony code
  - This has been specifically asked by @Acid_Snake, a leading developer of the ARK CFW for the PS Vita and PSP

Reverse-engineering

- PSP main CPU is a 32-bit MIPS-R4000 chip
  - Modules compiled into MIPS ASM (similar to MIPS IV)
  - Docs & tutorials available on the repo
    - https://github.com/uofw/uofw/wiki/Reverse-Engineering-Tutorial
- Other components?
  - ME (MIPS)
  - SYSCON firmware (Renesas/NEC)
  - WLAN firmware (ARM)

Tools

- [PRXTool] + text editor
  - Very reliable output (except for variable imports)
  - “By hand”
- IDA (or related software)
  - SYSCON firmware
- Ghidra?
  - Lots of potential (decompiler!)
  - Not fully ready yet
    - Missing full PSP relocation type support
  - Cross-checking always required

Roadmap

- No hard promises – the uOFW team “comes and goes”
- Power & chkreg modules currently in review
- Looking into fixing interruptman, exceptionman
- Finish nearly completed modules (IDStorage, memlmd,…)
- SYSCON firmware

Want to help?

- Come talk to us on the PSP Hombrew Discord server!
  - Moral support is always nice 😁
- Feel free to improve a partially completed module or pick a new one – PRs welcome
  - Repo wiki has tutorials and tips on reverse-engineering
- Easy going team – no hard deadlines

Links

- Repo url: https://github.com/uofw/uofw
- Wiki: https://github.com/uofw/uofw/wiki
- Kernel API documentation: https://uofw.github.io/uofw/
- Technical documentation: https://github.com/uofw/upspd
- uOFW installer: https://github.com/uofw/uofwinst

- PSP Homebrew Discord: https://discord.gg/PVBqwtmD

Questions?

References

- [PSV-PSP]: https://github.com/vita-nuova/bounties/issues/5
- [uOFW Ctrl]: https://uofw.github.io/uofw/group__Controller.html#ga95f41716b1ffa89303934f2eee1895be (accessed: 03/10/2021)
- [PSPSDK Ctrl]: https://pspdev.github.io/pspsdk/group__Ctrl.html#ga440b3c15e9c77294fd04a4c1d378b8a6 (accessed: 03/10/2021)
- [Baryon Sweeper]: https://www.psx-place.com/threads/baryon-sweeper-r1-release-unbrick-1000-2000-and-3000-psp-consoles.32503/
- [Melib]: https://github.com/IridescentRose/MElib
- [VLF]: https://psp.brewology.com/downloads/download.php?id=9308&mcid=1

References (2)

- [YAPSPD]: http://hitmen.c02.at/files/yapspd/psp_doc/frames.html (accessed: 03/13/2021)
- [PSPTEK]: http://daifukkat.su/docs/psptek/ (accessed: 03/13/2021)
- [devnoname120]: https://twitter.com/devnoname120/photo (accessed: 03/28/2021)
- [PRXTool]: https://github.com/uofw/uofw/wiki/PRXTool