Clover Of Khaki Color

Author Slice

Clover Of Khaki Color. Revision 5172
Moscow, 2026

Table of contents

Preface

What are we talking about? Well, certainly not about a flower growing in a meadow for joy cows. We are talking about software, about the bootloader of operating systems, which allows you to run an unusual system on a regular computer – Mac OSX. Apple did this does not allow it to be done, primarily citing the fact that “we cannot provide It works on computers not made by Apple”ю Well, let’s put it system at your own risk. And you shouldn't get any commercial benefit from this in order to avoid other legal complications. Non-Apple computer with the installed MacOS system is called Hackintosh, the origin of the word is clear.

To boot MacOS on a Hackintosh, you need a special bootloader, there are many of them different, but based on their basis they can be divided into two classes: FakeEFI and RealEFI.

FakeEFI was invented by David Elliot many years ago, and operated on a simple principle: let's pretend that EFI has already worked for us, leave traces of its activity in memory (boot-args and the entire table tree), leave EfiRuntime in memory in a simplified form "Not supported", and launch the mach_kernel kernel. This is how Chameleon works, and it works successfully but with a few exceptions like the "Boot Disk" panel. It is possible that with Apple will give us other problems related to the lack of Runtime Services in time. January 2013: It happened! iMessage stopped working because it absolutely needed SetVariable(), which is "Not supported" in Chameleon. We somehow overcame it, but Chameleon has problems again. Legacy bootloader options: Chameleon, enoch, Chimera, PC-EFI, revoboot. Can we already declare the death of this method by 2020?

RealEFI should have been flashed instead of BIOS, but for those who have a motherboard on BIOS-based, bootable EFI was invented. This EFI boot system on a BIOS machine was invented by Intel, an open source project at tianocore.org. (Source the project has sunk into oblivion! It survived only in the Clover project, and was copied to Opencore). Actually, this bootloader is called DUET. But here's the problem, it loads EFI, but loading Mac OSX operating system is not provided there. The next step is to adapt DUET to Mac OSX requirements. New motherboards already have EFI, even UEFI, but it is also unsuitable for loading Hackintosh. EFI bootloader options are divided into two categories: for PC BIOS - bareboot, XPC, and for UEFI BIOS – Ozmosis, OpenCore. Clover serves both categories.

The name Clover is given to this loader by one of the founders of the project. kabyl, who saw the similarity between the “Command” key, which only exists on Macs, and four-leaf clover.

Four-leaf clover is a single clover plant that has at least one four-lamellar leaf, unlike the usual ones three-plate. In the Western tradition, there is a belief that such the plant brings good luck to the finder, especially if it is found by chance. [1]

According to legend, each of the four-plate leaflets represents something specific: the first is hope, the second is faith, the third is love, and the fourth is good luck[2].

By the way, the original green logo looks more like a hare's cabbage than a clover. Namely Oxalis, and it is available in flower shops as both a three-leaf and a four-leaf variety :)

In the Russian version we call the loader "Кловер". That is, not «Клевер».

The project is being developed on the
~~http://www.projectosx.com/forum/index.php?showtopic=2562&st=~~0 RIP
https://www.applelife.ru/threads/clover.42089/
http://sourceforge.net/projects/cloverefiboot/
https://www.insanelymac.com/forum/327-clover/
https://github.com/CloverHackyColor/CloverBootloader
There are many more forums with a smart look telling about Clover, but they have nothing to do with real developers.

According to the text of the book:
Red highlights what you need to pay attention to. Moderator color.
Green highlights what is outdated and no longer supported, but is retained in the text for reference.
Mold color. Headings and links are highlighted in blue. Key words are highlighted in **black bold**. Another point. The original version of the book was written 15 years ago, and in some ways it looks naive. This is history, but the current state is also present.

1 I wonder how you can find clover by chance? Regularly plucking grass in the meadow?!

The cornerstones of hackintosh

The very first hackintosh was made by Apple, it was a regular IBM PC with a Pentium 4 processor, the model was called ADP1, the system was 10.4.0 Tiger. The bootloader was invented boot123 (three boot steps), which, like the kernel source and some kernel extensions (kexts), was published. The only obstacle to making other hackintoshes at that time was the encryption of some parts of the system. However, the protection is very weak, the decrypting kext is called DontSteelMacOS and the encryption key was originally in the kernel itself, it just looked like the poem "We've done a great job, don't steal our system!". However, before that, hackintoshers still had to figure it out. It seems that Semtex was the first.

So, t**he first stone**, deliver the key to the system:
- insert into the kernel;
- dsmos.kext by Netkas, injects into the kernel
- AppleDecrypt.kext, I don't remember the original author, but essentially it is the same dsmos,
- FakeSMC by Netkas, this is another method, not into the kernel, but into the SMC protocol,
- VirtualSMC by vit9696, essentially the same FakeSMC. Other copies of the same.

The second stone is the bootloader. Boot123 only boots computers with the correct DMI (SMBIOS), initially they simply reworked the system kernel so that it wouldn't complain, and then they decided to make it so that the kernel was untouched ("vanilla"), and entrust the mimicry to the bootloader.
- Chameleon 1 based on Boot123, works in RealMode 16bit
- Chameleon 2 with a graphical shell, and already in ProtectedMode 32bit,
- Clover, which is what we are talking about here,
- OpenCore by vit9696, made for the sake of "everything in its own way"...

The third stone is the adaptation of UEFI services present or missing on IBM PC computers, so that they work with macOS.
- on legacy computers, this is done by a modified DUET, not pure.
- on UEFI computers, this is OsxAptioFix and numerous followers. At the moment, it has turned into OpenRuntime, using Dmazar's original developments.
- and a number of other protocols that are present in AppleEFI and are absent in IBM PC, for example protocols for FileVault2.

The fourth stone is a set of drivers for the hardware that is not used in real Macs. The open source drivers that Apple initially made publicly available were of great help. And then the Hackintosh developers simply finished them. Some drivers were written based on Linux and FreeBSD drivers, using the canvas laid out in the published Apple drivers. This list includes
- network drivers: Intel, Broadcom, Realtek, Atheros and others,
- drivers for the PS2 keyboard and touchpad - VoodooPS2 and similar,
- drivers for sound: VoodooHDA, kxAudio and variants of the patched native AppleHDA,
- the USB3 driver was developed by Hackintoshers earlier than Apple itself,
- the SDHC driver was a year earlier than Apple had such a driver, but it has not shown its source code,
- but what is missing is drivers for video cards, Hackintoshers can only patch the native drivers. A patch is a "patch" in English, so to correct a dozen bytes in the native driver by several megabytes, so that it at least somehow works. And here OCLP - Open Core Legacy Patcher comes to the rescue, its essence is to bring drivers and frameworks that depend on them from the old system to the new system. For example, HD4000 worked in BigSur, but stopped working in Monterey, so with the help of OCLP you get drivers from BigSur in Monterey. This OCLP is positioned as if for real Macs, Opencore is clearly involved in the name, nevertheless, it works on hackintoshes and under Clover.

Development timeline

The need for a new bootloader arose due to Chameleon's inability to boot the then-emerging 10.7 (Lion) system.

The project started on March 4, 2011, on the initiative of Kabyl, who, however, having told everything he had managed to understand by that time, avoided development, and soon disappeared altogether. I have serious suspicions that he is no longer in this world.

The first launch of the MacOSX system with a modified DUET took place on April 6 2011 http://www.projectosx.com/forum/index.php?showtopic=2008&view=findpost&p=13810

On May 4, serious problems with the new bootloader were formulated; without solving them, there was no point in the new project. The pause dragged on until August, because it seemed unrealistic to me to cope with these problems alone.

Meanwhile, Chameleon came to life, having managed to boot Lion, and I worked on my branch for some time. However, the Chameleon admins ignored me, so I gave it up. Then Ninja appeared with his iBoot, and I joined him in trying to make an EFI bootloader and solve the hanging problems. This project started in August 2011, and along the way I modified DUET (CloverEFI), using the sum of CloverEFI + iBoot. However, the dirty origin of this iBoot did not allow me to properly develop.

09 August 2011 with the participation of dmdimon made a Russian font for the bootloader. In the meantime I make SMBIOS and ACPI at a much higher level than it was in Chameleon.

October 19, 2011 finally solved the problem of running Duet on a laptop. Before that it was just a reboot. A completely non-trivial investigation!

November 14, 2011 appearance of cats in the Clover interface. That is, for 10.4 we draw a Tiger, for 10.5 a Leopard, and so on. A nice innovation! Appearance plays a role.

December 14, 2011 The memory panic problem on low-end OSX systems was resolved. For some reason, Lyon and older didn’t have this problem.

05 January 2012 the sleep problem was solved. It was from this moment that the project could be considered viable. By this time Ninja had already left the scene, and I decided to start my own project of a graphical bootloader menu based on the already known rEFIt. It is license-clean, and now it was possible to fight for international recognition of the project. Thus Clover-v2 appeared, with my developments from Clover-v1.

The creation of the new shell took two months, and the first publication took place on the 29th February 2012. Actually, rEFIt already existed, it was just not suitable for compilation in the EDK2 environment, and all its libraries had to be rewritten and replaced with our own. Well, and add everything hackintosh-specific to the project. Along the way, together with jadran, the project compilation tools were developed. Now with the help of gcc-4.4, and now 64 bits.

It should be noted that Clover's project turned out to be unusually timely. The project EDK2 has only just begun to develop, there are still years of growth and improvements ahead, in which I also took part. The gcc project is also developing, as they say, just now. There are still only a few percent of computers with EFI BIOS, and we have already launched EFI bootloader.

09 March 2012 Dmazar, whom I have known since August 2011, joined the project with his idea to make a UEFI bootloader based on Clover. March 31, 2012 Usr-sse2 made interactivity for entering parameters in the bootloader shell - Options Menu.

April 12, 2012 crazybirdy made an installer for Clover.

On April 21, 2012, Dmazar defeated UEFI boot, but continued to work on the project — improving and fixing. Namely, he created OsxAptioFixDrv - the key technology - the third stone of Hackintosh.

On June 5, 2012, pcj appeared and offered its sources with new technologies: DSDT patch, Kexts Inject, Kernel patch, which raised our bootloader to a completely new level, unattainable by competitors. (its codes were not taken in Chameleon)

September 18, 2012. Pene called me and Dmazar to a round table to think about iCloud issues. Resolved by September 21st.

September 30, 2012 mouse appears in the bootloader interface.

On October 19, 2012, animation was made in the bootloader shell.

October - December 2012 step by step made native permission in bootloader for Nvidia, ATI, Intel, for CloverEFI and for UEFI. These achievements have raised the bar for what a good bootloader GUI should be. I would also like to note Blackosx's contribution to improving theme support.

January 09, 2013 the problem of paid iMessage was solved, which could not be done in Chameleon. This was a fundamental victory of the very idea of the EFI bootloader. Nothing is impossible, and in Chameleon a month later they repeated this method, but users now went to Clover.

Spring 2013. Thanks to JrCs, Clover has acquired additional utilities and internationalization. 20 languages in the installer, control panel and automatic update service. Compilation, start and end scripts have significantly increased in functionality. The bootloader became a complex for servicing hackintosh-systems.

July 27, 2013 finally solved the problem of sleep during UEFI boot, which had been hanging since autumn 2012. It seems to have appeared with an OS update.

September 29, 2013 and also fixed sleep, shutdown and restart in UEFI boot. From now on, you can set UEFI boot as the main method on those computers where it is possible.

January 20, 2014 the possibility of deep sleep was made - Hibernation. Not in all cases, but at the moment Clover is the only loader that can do this.

February 2014. Project of the Month award on sorceforge.net.

April 5, 2014. The end of development at revision 2652 has been announced. This, of course, does not cancel possible improvements in the future, it’s just that nothing will change radically, the main thing has already been done.

June 2014. Apple released the 10.10DP1 Yosemite system, the first installation successes and new fixes in Clover for the new system began. And then it turned out that Chameleon is not able to boot this system. There are single successful reports from bareBoot and Ozmosis bootloaders, which are also EFI bootloaders using part of Clover codes (namely AptioFix, at least). Clover became the main Hackintosh bootloader. Chameleon was fixed again, it boots Yosemite, but without iMessage so far.

August 21, 2014. Dmazar fixed the functionality of NVRAM during UEFI boot for some people who had problems with it before.

Meanwhile, version 2k has crossed revision 3000. The new compiler with LTO optimization, new code size, fixes for old bugs, improvements in algorithms. Almost nothing new, but Clover has become better.

January 2015. Since Clover-3, announced by Apianti, is still not here, Clover-2 purchased version 2.3k, which reflects revision >3000. All the developers have fled, the projectosx.com website has ceased to exist. I am the only one left, but I continue to work, although without much zeal. I have a new toy - Wine.

June 2015. The release of 10.11 El Capitan. Clover loaded it without any issues. Chameleon got stuck again. We managed, but as I see, regular users are no longer interested in Chameleon. Only the same 6 people who support it are interested. Chameleon/Chimera is still used by newbies who started life with Multibist and have not yet heard of Clover, or AMD CPU users who are not developers and they simply do not know what to do with Clover. One new problem, kext injection stopped working. The solution was provided by user solstice (revision 3258), who dropped by chance. Ozmosis users were left without a new system for a long time before its developers provided a new version with this patch.

Spring 2016. Zenith432 made a compilation with Xcode, we no longer we need gcc.

Summer 2016. With new coders, new major changes in Clover's interface. Scroll wherever you need it, checkboxes and radio buttons, new design styles.

Autumn 2016. Together with vit9696, the long-awaited support for FileVault2 technology was made.

Winter 2017. Released revision 3999. Next will be 4000 and version 2.4k. Clover is still needs development, because it needs support for new systems, for example for Sierra 10.12.4, for example, required a new patch.

Summer 2017. The High Sierra system with a new file system apfs appeared. For There is no driver for Chameleon yet. For EFI bootloaders there is a native one from Apple. By the way, multibist switched to Clover a long time ago, like other commercial projects.

Autumn 2017. Victory over the black screen of Radeons. This was done by vit9696 in his kext WhateverGreen, Mieze explained how to do it from the ground up, that is, not to edit consequence, but the cause, and I added it as a checkbox in Clover's config RadeonDeInit=true. Looking more closely, I realized that the solution is different, Miez took one step more, what made her do it I don't know, but now I know that it's not just a copy. The solution is incomplete, in terms of the graphics plant, but good enough for the Radeons to sleep.

And again vit9696 found a solution on how to fix AptioFix so that the native one works NVRAM, now almost everyone! Except for some left-wing semi-legacy computers.

Spring 2018. vit9696 figured out how RTC is used in macOS, and thus The problem of saving the hibernation key has been solved, as well as other RTC problems. However… Hibernation again almost doesn't work for anyone. Apple doesn't need it.

June-December 2018. Support for SVG vector graphics, and therefore scalable themes. My nanosvg-based project and Clovy's SVG theme sample, and very much thanks to his help.

January 2019. Sound appears in Clover's interface, thanks to Goldfish64. However, the toy had to be put aside due to problems. That is, it works, but is incompatible with VoodooHDA.

July 2019. A series of new patches for download 10.15. And revision 5000! Now Clover received version 2.5k.

May 2019. vit9696 created a new OpenCore bootloader based on new ideas with his ideology, but Clover continues to live and develop. We will coexist and cooperate. Clover should be cleaned of the unnecessary, the crooked, take the best from OpenCore, but it's an unrealistically big job, we're improving slowly. The most important thing is that he did injection and patch of kexts into the KernelCollection, which is used since the system BigSur, and thus he was able to load this

system. Clover had been guided until then on prelinkedkernel, and BigSur was not available for it yet.

I would also like to note that Vector Sigma has joined Clover's team and has taken on the entire user-space. Controls the compilation scripts, execution, installer, and created special application Clover.app. That is, it controls everything that was previously under guardianship of JrCs. So the bootloader binding is also alive and well.

September 2019. We did a major overhaul of the entire project, moved to GitHub, which became stable and fast after Microsoft bought the site, and along with this the following changes occurred:

- Clover's sources are now controlled by the GIT protocol instead of the SVN protocol. In this It has its downsides, namely the problem with revision numbering, but it also has its upsides, such as the ability to bisection to find a recently made error.

- I'm tired of following strange "improvements" of EDK2. Before that, we used a stable version of UDK2018, but it was clearly outdated, and with the new EDK2 Clover turned out to be simply incompatible. The solution is to include all the necessary libraries from the new EDK2 simply in the Clover project with our patches, and add those libraries that are alreadyexcluded from it, taking them from the old UDK2018. We are talking about support for legacy computers on Core2Duo and the like. Now Clover does not need external libraries, everything necessary is contained in its repository, and all this will be updated by the developers of Clover itself, based on changes in EDK2. By the way, bisection became possible precisely with the rejection of any external sources, everything necessary is contained in one folder.

- now the main compiler of the project is again gcc-10 and further, unlike clang it is able to implement LinkTimeOptimisation, i.e. discarding unused libraries at the linking stage. The size of the codes has decreased from 960kb to 870kb! At the same time, we get additional diagnostics of possible errors. One of them is that EDK2 allows the use of pointers to an empty array, and this contradicted some of Clover's algorithms. After a small correction, the design played differently! And other bugs could also go away.

- the structure of the drivers folders has been changed, since we have completely abandoned the idea of compiling in 32-bit, and specifying 64-bit is redundant.

- the kexts folder structure has been changed, now you can enable and disable different versions of kexts in the interface to check which one works better.

January 2020. The next major transformation of Clover. A new developer, Jief Machak, came along and offered to convert the project to C++. This is not a trivial action, since it requires changing some libraries and compilation methods. But he did not come empty-handed, he brought the necessary files, dating back to 1997, obviously already solving such a problem with C++ back then. And now he is adapting it to the new conditions. I accepted the challenge and now we have already started transformations in Clover as a team. And Pene too helps us. Of course, C++ implies compilation of files written in C, so at the first stage of the transition to C++, nothing happened. But there is no point in talking about a new language and programming in the old one. To take advantage of C++, you need to use them. Let's leave Linus Torvalds alone with his curses against C++, his motives are clear, Linux is already written in C, and it can no longer be remade, all that remains is to look for excuses for this. And we can rewrite our sources, at least partially, after all, Clover is much smaller in size, and not everything can be rewritten, keeping in mind the compatibility of the language. And yet, what do we do? We introduced new programming paradigms and rewrite codes to meet new requirements. And this simply leads to its rethinking, the development of new algorithms, and the elimination of old bugs, the untangling of pasta. GUI is written on new algorithms, on classes, and therefore old themes suffered, but they can be cured, and

in Clover not everything has been fixed yet, the work is in progress. And the result of the transformations is faster work and elimination of bugs that previously could not be untangled. A side effect of this work is that users left Clover and began to use the new OpenCore, without waiting for them to figure out Clover.

Summer 2020. The appearance of the 11.0 Big Sur system showed that the myth about the inability to fly symbolic patches is unfounded. It flew off both in Clover and in OpenCore. Well, let's move on.

Autumn 2020. Jief Machak got down to work again, connecting the Clover and OpenCore loaders. OpenCore already loads BigSur, because vit9696 made kexts loading directly into the cache, and BigSur rejected Clover's method of injecting into the device tree with its new security requirements. What is easier, to understand this problem, or to take the method from competitors? Jief decided to go the second way. Well, so be it. How combine? Vitaly's idea to take Clover's GUI as a set of libraries and attach it to OpenCore is, unfortunately, unrealistic, Clover is already written in C++, and no one will cut it back into C libraries. But taking OpenCore libraries and running them from Clover is no problem, just painstaking work with delegation of authority. So, he did it.

Now the combined bootloader, from the two repositories,
https://github.com/CloverHackyColor/CloverBootloader
https://github.com/CloverHackyColor/OpenCorePkg

compiles as a single unit, runs as Clover and looks like Clover with its GUI and settings, and when selecting a system to load, delegates settings to OpenCore and gives it control. Yes, OpenCore is not original, since its team does not want to cooperate, we have no choice but to maintain our own version and make our own changes.

January 2021. Once again, Jiff has done the trick. Clover is now able to launch DMG files, and specifically BigSur Recovery, as well as installers.

Summer 2021. New Monterey System Brings New Surprises. All Users went to OpenCore, and it couldn't upgrade Monterey. But Clover could! And people started coming back...

September 2021. Of course, there are many new achievements in OpenCore, so we upgraded our OS version to the modern 0.7.3.

Summer 2022. Now up to 0.8.3. One fix was very relevant. Ventura beta 3 stopped loading (OpenCore too), it turned out that Apple changed the definitions. How did they find out about this? No way, Marvin stole the codes from Apple and saw this change. Sent them to OpenCore, from where they got to Clover, with the written blessing of the same Marvin. I must admit, I had no way to figure all this out. Disassemble the kernel (there were no sources yet), and look for the cause of the crash in these megabytes, a discrepancy with the old codes? A way, of course, but with very little chance of success ...

I should also say that Clover is not only trailing behind OpenCore with its patches. Clover's GUI and the ability to change settings on the fly are still Clover's unique feature. ACPI patches, and not only DSDT, like in Clover, are not available in OpenCore. Kernel patches are almost the same, but with more capabilities, hackers just don't use them, since most hackers have switched to OpenCore. So Clover is not dead, it continues to live for those who understand it. We fix errors, OpenCore is not without sin either. But when people come to me with a request to help "I have the latest OpenCore", I refuse. Sorry!

December 2023. Jiff is back again, updated OpenCore to the latest version 0.9.7. (and they, in revenge, increased the version to 0.9.8, without making any changes, except for formatting the documentation). But he did two more notable things: fixed memory leaks that affected if you operate in Clover's GUI for a long time. (OpenCore does not have a GUI at all), and fixed the

memory libraries, in particular copymem and setmem, this turned out to be a solution to the bug with BlockSkywalk. For some reason, pure OpenCore did not get this bug, although all the algorithms are from there.

Tactical and technical characteristics

EFI – Extensible Firmware Interface – an extensible interface for accessing hardware-dependent functions. Unlike BIOS, which takes up 64 KB and is written in 16-bit codes, EFI takes up 4 MB, is written in 32 or 64-bit codes, and is positioned as hardware-independent, although... of course, miracles do not happen, and 100% compatibility with any platform is impossible to achieve. The real UEFI BIOS is, of course, hardware-dependent. Clover is an EFI bootloader for operating systems that already have UEFI BIOS (Unified EFI...), and for computers that do not have it. At the same time, the operating systems themselves can support EFI boot (OSX, Windows 7-64EFI, Linux), or not (Windows XP), in the latter case, legacy-boot is provided - a return to the old BIOS boot scheme through boot sectors. Strictly speaking, Clover is not a bootloader, but rather a Boot Manager that prepares and launches native bootloaders of different OS (boot.efi, grub.efi, bootmgfw.efi, ntloader). EFI is not only the initial stage of OS loading, it also creates tables and services that are available for use in the OS, and its performance depends on the correctness of this stage. It is impossible to load OSX on the built-in UEFI, just as it is impossible to load OSX from pure Duet. CloverEFI and CloverGUI do a lot of work to adjust the built-in tables to be able to launch OSX:

- the SMBIOS (DMI) table is filled with data emulating real Apple Macintosh computers - a condition for launching OSX. The serial numbers are fictitious, but suitable. However, it is desirable for the user to substitute unique numbers. Why, for example, does GRUB not load HackOS? Because they do not have the right to include serial numbers of Macs in it...

- ACPI tables contained in the computer's ROM usually contain errors and shortcomings, most often due to the laziness of manufacturers: the APIC table has an incorrect number of CPU cores, there is no NMI data, the FACP table does not have a Reset register, the power profile is incorrect, the SSDT tables do not have data for EIST, and as for DSDT, it's a long story. That is, MacOS has its own requirements for ACPI, which do not match the ACPI of regular computers. Clover is trying to fix all this. It is generally a bad idea to use the same DSDT for different systems, they are different not only in the executable part;

- OSX also tries to get data from the bootloader about additional devices, such as video card, network, sound, etc. through the Device Properties string mechanism. Clover generates such information;

- for BIOS-based computers, it is typical to use USB at the initial stage of booting in Legacy mode, which becomes unacceptable when transferring control to the OS (BIOS Ownership, absent in Apple). The bootloader switches the USB operating mode;

- OSX also exchanges information with EFI via special NVRAM memory, access to which is carried out via RuntimeServices, which are absent in legacy bootloaders. Clover provides such information exchange, and two-way, which provides correct operation of Firewire, the ability to load the Nvidia Web driver, the ability to use the "Boot Disk" control panel for automatic reboot into another system. NVRAM is needed to be able to register in the iCloud and iMessage services, for hibernation, for system updates, for FileVault2 operation, for rebooting during system installation, etc.;

- the ConsoleControl protocol is required, which is absent in Duet, a dozen necessary or desirable protocols, NVRAM variables, and other similar little things;

- it is necessary to fill in some data in EFI/Platform via DataHub protocol, absent in Duet, and not always present in UEFI. The most serious value is FSBFrequency, the definition of which is the task of the bootloader, the data in DMI is inaccurate or absent altogether, and we also use this protocol for sensors, although not all hackintosh users use it, and someone goes their own way, deletes DataHub, and wonders why something does not work for them. I got tired of this, and now DataHub is always there, regardless of the wishes of a kind prompter; - before starting work, the CPU must be correctly initialized, but, since system boards are made universal, for a whole range of different CPUs, the internal tables do not contain data on the processor, or some universal data is presented, incorrect in a particular case. Clover carries out a full detection of the installed CPU and makes the necessary corrections in the tables, and in the CPU itself. As one of the results - the Turbo mode is turned on;

- BIOS somehow uses the computer's memory, divides it into regions of a certain purpose, but often this is not done as needed to launch macOS; The driver is responsible for adjusting the memory card and other related details OsxAptioFixDrv, and its descendants, which for a number of reasons was separated from Clover into a separate module, first of all, so as not to interfere with legacy loading; - another little thing. The source code of DUET, and of all EDK2, are written universally for different hardware...., but the dependence on hardware is made through constants. That is, it is assumed that the user compiles Duet for one specific configuration. The goal of Clover is to be universal, with platform autodetection. But Clover not only edits the raw information it provides to the system, it also edits the system itself to make it more accommodating to Hackintosh, these are all sorts of patches and quirks. And in some cases, Clover changes the state of devices that were incorrectly initialized by the native BIOS:

• RadeonDeInit

• HDAReset

• HaltEnabler

• FixOwnership

• Intel backlight

All this is done automatically, and a beginner can use Clover even without understanding anything about the problems mentioned. But for an advanced user, Clover provides the ability to manually change many parameters. Their consideration is the purpose of this book. By the way, some people begin their acquaintance with Hackintosh by studying the config. This is the wrong approach. You will correct the config after you try to load something. More on that later.

What is what?

To recap, Clover performs four main functions:

1. Intercepts control so that the BIOS does not load the operating system, so that it is done through Clover. The BIOS calls Clover, and it loads the operating system, by the way, it will also ask the user which of several operating systems to load.

2. Modifies the data that the BIOS transfers to the operating system, this is the main thing requirement for loading MacOS. For example, a serial number.

3. Modifies the MacOS system itself so that it works on this hardware.

4. In some cases, Clover corrects the state of devices, which it did not do at the time.
made a BIOS.

Windows and Linux do not need point 3, and the second one is rare. Point 4 was once useful to launch Windows with sound: launching directly - no sound, launching Windows through Clover - there is sound. Fantastic reality! But the first point requires clarification, since beginners start with the question "Why is it not loading". When turning on or rebooting the computer, loading the operating system using Clover occurs in the following way.

Option A. A computer based on BIOS (legacy loading)

```
BIOS→MBR→PBR→boot→CLOVERX64.efi→OSloader
```

OSLoader is boot.efi in case of MacOSX, bootmgfw.efi in case of Windows, grub.efi in case of Linux.
Not clear? In words:
1. When you turn on the computer, the **BIOS** starts. Based on its settings, it selects which disk to start from if you are booting legacy.
2. The BIOS reads the zero sector from this disk. The sector is called **MBR**. The codes are written to memory, and the BIOS passes control to them.
3. The program embedded in the MBR looks for a boot partition on this disk, reads the first sector from there into memory, and passes control to its codes. This is called the **PBR** sector.
4. The program embedded in the PBR finds a file called boot in its file system, loads it into memory, and passes control to it. In our case, the **boot** file contains DUET, or **CloverEFI**.
5. CloverEFI searches for the CloverX64.efi file, loads it, and passes control to it. Then as in option B.

Option B. Computer based on UEFI BIOS (new scheme, UEFI boot)

```
UEFI_BIOS→CLOVERX64.efi→OSloader or
UEFI_BIOS→BootX64.efi→OSloader or
UEFI_BIOS→EFI/microsoft/boot/bootmgfw.efi→ «Windows»
```

1-5. If the BIOS is UEFI, then it can find the startup EFI module in the EFI section itself. By default, such a file should be called **EFI/BOOT/BOOTX64.EFI**. Some BIOSes have the name **EFI/microsoft/boot/bootmgfw.efi** baked in. But we can teach the BIOS to load the **EFI/CLOVER/CLOVERX64.EFI** file right away. You just have to learn how to do it yourself. For those who don't know how, the EFI/BOOT/BOOTX64.EFI file is also replaced by Clover.
6. Finally, we loaded CLOVERX64.EFI, that is, the GUI, we see a choice of operating systems, and we can load one of them.

For all this to happen, the following files must be written in the following

MBR sector

Is the zero block of the external storage device from which the boot occurs (HDD, SSD, USB Stick, USB HDD, DVD). The first 440 bytes should be written to this block. One of the options is **boot0af** (Active First) – its role is in searching for an active partition in the MBR disk layout, and transferring control to its PBR sector. A hybrid MBR/GPT partition scheme is possible. If the layout is pure GPT, that is, there are no active partitions, then control is transferred to the EFI partition,
which is recognized by GUID=C12A7328-F81F-11D2-BA4B-00A0C93EC93B. If you suddenly reformat this partition, check what GUID it now has. There is an option when the first partition is FAT32 as expected, but the boot does not work, because its signature is 0C00, and it should be EF00. It is fixed with the gdisk utility.

boot0ss (Scan Signature) – searches for the first partition with the signature 0xAF, i.e. the HFS+ partition with OSX installed, and transfers control to its PBR. In this way, you can boot the system from an HFS+ partition on a GPT-partitioned disk, but only from the first such partition. If such a partition is not found, then it continues searching for FAT32 or ExFAT partitions - in this order of signature checking, and passes control to it.
boot0ab - search for a partition with signature 0xAB - Apple Boot Partition. That is, Recovery. For perverts. In reality, Recovery has never served for legacy booting.
boot0md - a combined option that searches for an HFS+ partition on several disks, and not just on the main one. A strange option, not included in any installation options.

PBR sector

Is the first blocks of each partition on the selected media. The phase number two bootloader is written here. This bootloader knows the file system of its partition and is able to find a file named boot there in order to load it and transfer control to it. Accordingly, there are options for different file systems.
boot1h2 - for the HFS+ file system and support for a boot file length of up to 472 KB. The old boot1h option, distributed with the Chameleon bootloader, only supports a 440 KB file (472 is required). For those who first installed Chameleon, I remind you once again: sector PBR needs to be updated with the boot1h file from the Clover kit, otherwise it will not start. The boot1h2 option has a 2-second pause to switch the bootloader. It seems that they finally fixed this in Chameleon... boot1h — also, but without a pause.
boot1f32alt — for the FAT32 file system. This file system has write support, so it is very convenient for installing a bootloader on it. You can use the EFI partition, you can use the flash drive as it is, since flash drives are already formatted in FAT32[2] . Has a 2-second pause. boot1f32 — an option without a pause. Attention! The file system must be FAT32, not MSDOS, because FAT16 is unacceptable. The bootloader does not work on it!
boot1x — Zenith432 made a boot sector that is intended for the exFAT file system. Unfortunately, the driver inside Clover is only ReadOnly, however, this file system is very interesting for external media, because it supports files larger than 4 GB, and at the same time is supported by both Mac and Windows for writing.
These sectors (or rather, phase 2 bootloaders) have another useful function. They have an initial delay at start-up of two seconds, waiting for keyboard input. The entered digit will be appended to the file name, i.e. by pressing key 1 on the black screen at the very beginning of booting (after the BIOS message "booting from...") we will boot the file **boot1**, by pressing key 3 we will boot the file **boot3**, by pressing key 6 we will boot the file **boot6**. The point is to keep different bootloader options in one place, or even different bootloaders, simply by giving them different digits. For example: boot — Clover, current version, or test version boot1 — Chameleon boot3 — Clover-32bit, tested, working version (displays number 3 on the screen) boot4 — Clover-32bit, with BiosBlockIO driver boot5 — Clover-64bit, shortened in size so as not to overlap EBDA (see below) boot6 — Clover-64bit, tested, working version (displays number 6 on the screen) **boot7** — Clover-64bit with BiosBlockIO driver, working with any controllers supported by BIOS. (displays letter B or L or 7 on the screen).
In addition to these options, the PBR sector may contain the Windows boot manager, which knows the NTFS file system; GRUB, which knows the EXT4 file system; and others that have nothing to do with Clover. At least at this stage.

2 for historical reasons, it is recommended to reformat each flash drive, and it is also necessary in Windows system. This recommendation is now outdated and has no basis.

Boot or CloverEFI (including boot6 or boot7)

In the case of Chameleon, the "boot" file is the entire bootloader. In the case of Clover, this The file contains the entire EFI system, as well as the boot service for transferring control to the next stage. All of this, in option B, should be considered already present in the computer's ROM. In reality, it turns out that not everything is there, and some details (the so-called drivers) should be loaded additionally. Details already collected in the boot file of option A. Unlike the previous stages, the boot file already differs in bitness, i.e. separate options for 32-bit and 64-bit booting. In most cases, you should choose the 64-bit option if the processor supports this instruction set (attention! Pentium 4 and Yonah - only 32 bits). However, if for some reason you intend to work only in a 32-bit OS, then it makes sense to load EFI32. It is 20% smaller in size, and accordingly faster. Unfortunately, it is incompatible with Windows 7 EFI, which is always 64-bit.
November 2016: Goodbye, Clover-32! I'm not interested in supporting him anymore.

The boot file is essentially a modified DUET. Moreover, corrections in essence are unlikely to be 1%, but this percentage provides a fundamental difference from Duet - Clover works for the purpose for which it is intended. If someone believes that I have been working on Nnya for three years, editing DUET, and that it is enough to take vanilla and add AppleSim to it, then have a good trip! This is far from true, but I have no purpose to describe all the intricacies of this development. The deed has already been done. From now on, we will call this program CloverEFI. With each revision, its binaries differ, because both EDK2 and sometimes its sources change, for example, the latest changes were like StrCpy => StrCpyS, a more secure version. However, the functionality of CloverEFI has not changed for a very long time, so if you have a problem with a new version, install the version that worked for you. So, the kit that was officially distributed on SF.NET includes the old file boot7-MCP79, because for some reason the new boot7 stopped working with this chipset. Another minor problem. PBR is looking for the boot file. But the Windows installer has a folder with the same name, in the root. You have to rename your boot file to boot5, and use an alternative boot. Somewhere I made a version of boot1f32, which by default loads boot5 right away, a one-byte fix. But this version is not present in the standard installer. That is, the solution is not for everyone. A Hackintosher should at least have some brains.

CLOVERIA32.efi and CLOVERX64.efi are CloverGUI

This file, in two versions of different bitness, is a graphical shell of the bootloader for selecting the operating system, for installing additional options, for loading additional drivers and actually for starting the OS. The graphics and menu were originally based on the rEFIt3 About project. The original part was barely 5% of the entire program, and even then in a corrected form. And now both the graphics and the menu , which is reflected in the directory name and in the menu have been rewritten to C++ classes, so that only memories remain of the original codes. From now on, we will generally call this program CloverGUI.

Folder structure

In addition, the bootloader needs additional files, the folder structure will be roughly the following:

```
EFI:
        BOOT:
                BOOTX64.efi
        CLOVER:
                ACPI:
                        WINDOWS:
                                SLIC.aml
                        origin:
                        patched:
```

```
                DSDT.aml
                SSDT-1.aml
        CLOVERX64.efi
        themes:
                black_green:
                        BoG_LucidaConsole_10W_NA.png
                        icons:
                                func_about.png
                                os_clover.icns
                        banner.png
                        background.png
                        Selection_big.png
                        theme.plist
                Clovy:
                cesium:
                        theme.svg
        config.plist
        drivers:
                BIOS:
                        FSInject.efi
                        SMCHelper.efi
                UEFI:
                        CsmVideoDxe.efi
                        DataHubDxe.efi
                        FSInject.efi
                        VBoxHFS.efi
                        OpenRuntime.efi
                        SMCHelper.efi
        kexts:
                10.7:
                10.11:
                Off:
                Other:
        misc:
        OEM:
           Inspiron 1525:
                ACPI:
                        origin:
                        patched:
                                DSDT.aml
                config.plist
                kexts:
                        10.5:
                        10.6:
                                Injector.kext
                                VoodooSDHC.kext
                        10.7:
                                VoodooTSC.kext
                        Other:
                UEFI:
                        ACPI:
                                origin:
                                patched:
                                        DSDT.aml
                        config.plist
                        kexts:
        ROM:
        tools:
                Shell64U.efi
```

That is, the CLOVERX64.efi file should be located at /EFI/CLOVER/, and the font BoG_LucidaConsole_10W_NA.png in the folder /EFI/CLOVER/themes/black_green/. In reality, these and other folders are more filled with content. As the story progresses, we will describe in more detail what is what. A few words about the folder /EFI/CLOVER/OEM/ The folder is intended for storing boot options for different configurations. A typical situation is when we create a bootable USB flash drive, and on it, in addition to the general config /EFI/CLOVER/config.plist, there are

also well-verified /EFI/CLOVER/OEM/Inspiron 1525/config.plist and /EFI/CLOVER/OEM/H61M-S1/UEFI/config.plist, as well as our own well-developed DSDT.aml, different for different computers, and different sets of kexts.
The folder name is determined from SMBIOS and you can see in boot.log what exactly your computer is called:
0:100 0:000 Running on: 'Z170X-UD5 TH' with board 'Z170X-UD5 TH-CF'
The first line is the name of the entire system, typical for laptops, but on desktops there is something abstract. The second line is the motherboard model, convenient for desktops, but not for laptops. Both names are suitable for the name of your folder, choose the more understandable.
Your folder can also contain a UEFI folder to have different configs for UEFI (option B) and for legacy boot (option A) on one computer (although I personally doubt that anyone needs this).

EFI Drivers

It is worth noting that these drivers are only valid for the duration of the bootloader. They do not affect the loaded operating system, except indirectly, by how the devices are initialized (Clover's fourth function). What should be put in these folders? The user's choice.

- **NTFS.efi** – NTFS file system driver, for the ability to load Windows EFI, however... it seems not particularly needed, because the Windows EFI bootloader is also located in the ESP, on FAT32
- **HFSPlus.efi** – HFS+ file system driver, required to run MacOSX. It is necessary for option B, but in option A it is already present in the boot file. Unfortunately, this file is proprietary and cannot be present on an opensource project, so ->
- **VboxHFS.efi** — is a legal alternative to HFSPlus.efi, it is slightly slower. The new version supports links, even more than the native Apple HFSPlus.efi. HardLink, SymLink are supported! While HFSPlus.efi only supports hard links. In the modern version of Clover, the downloaded file displaces the built-in one. That is, if the boot file contains a built-in VboxHFS.efi, and the drivers/BIOS/ folder contains HFSPlus.efi, the latter will work.
- **VBoxExt2.efi** – EXT2/3 file system driver, required to run Linux EFI. Similar to VboxExt4.efi. Again, if Grub is not in the EFI partition.
- **Ext4Dxe.efi** - a more robust EXT2/3 file system driver
- **Fat.efi —** is the FAT32 file system driver. Of course, it's already there. However, AMI UEFI BIOS contains a very crooked driver, so I preferred to load my own version.
- **FSInject.efi** – a driver that intercepts the file system, to enable force injection of some kexts into the system if the system itself does not consider it necessary. Is it difficult to understand? We will return to this issue later when we consider the ForceKextsToLoad key. Does not work in new systems.
- **PartitionDxe.efi** – actually, such a driver is in CloverEFI, and it is also in UEFI, but it is not designed for either the Apple partition or the MBR/GPT hybrid. Conclusion: in option B, the driver is sometimes needed. If you have a normal GPT, you can do without it.
- **OsxFatBinaryDrv.efi** – a necessary driver for option B, ensures the launch of fat (Fat) modules, such as boot.efi in systems up to 10.9. In newer systems, you can do without it.
- **OsxAptioFixDrv.efi** – a special driver designed to correct the memory card, which is created by the crooked AMI AptioEFI, otherwise the OS cannot be started.
- **OsxAptioFix2Drv.efi** - a slightly modified version. It turned out to be possible with it Hibernate in 10.9.1 system with UEFI boot! But unfortunately this option does not load 10.7.5.

- **OsxAptioFix3Drv.efi** - new version, corrected by vit9696 to support native NVRAM, now you can do without nvram emulation!
- **AptioMemoryFix.efi** - this is a completely reworked driver from this series, author vit9696, with additional automatic search for the best value slide=xxx. At the moment it is recommended to use it first
- **OsxLowMemFix.efi** — a simplified version of AptioFix, suitable for some strange UEFI BIOS options (Insyde H2O).
- These five Aptio options will not be used at the same time, even if all are present, Clover will select one of them by priority: AptioMemoryFix, OsxAptioFix3Drv.efi, OsxAptioFix2Drv.efi, OsxAptioFixDrv.efi and lastly OsxLowMemFix.efi.
- For the new Clover revision 5120, it is recommended to forget all these *Fix* drivers and use the ~~OcQuirks+~~OpenRuntime combination, which are now included in the Clover package, and starting with 5122, OcQuirks is no longer needed, it is contained inside Clover;
- **NvmExpressDxe** — NvmExpress controller driver, which is positioned as a SATA replacement for SSD drives, keep in mind that if the controller is on the motherboard, then such a driver is most likely already in the UEFI BIOS. Needed for an additional controller.
- **Usb*.efi, UHCI.efi, EHCI.efi, XHCI.efi, OHCI.efi** – a set of USB drivers for those cases of option B, when the built-in drivers for some reason work poorly. Why would suddenly? Perhaps there is some connection to other functions that had to be disabled.
- **PS2Mouse…, PS2MouseAbsolute…, UsbMouse…** - a set of drivers to support the mouse pointer/trackpad/touchpad in the CloverGUI interface. These drivers do not affect the operating system. There is such a driver in the UEFI BIOS, you should not interfere with it
- **UsbKbDxe.efi** - this is not just a USB keyboard driver, it contains a kind of protocol necessary for Apple FileVault to work which is not enabled by default, the driver is required for FileVault to work.
- **AptioInputFix** – also for mouse and keyboard, if the UEFI BIOS option works crooked. Author vit9696.
- **DataHubDxe.efi** – this driver is already present in option A, and it is quite possible, is also in UEFI. But if it is not there, it is worth loading the external one. There will be no conflict, but you will be sure that it is there. More and more often I see complaints from users that something does not load or does not work for them. And they do not have DataHubDxe! Where did you come from, so cunning? Did they convince you on Tonimac that this driver is unnecessary? Then do not come to me either. In the new version of Clover, I simply forcibly included this driver inside. Now it is always there, and you can forget about it as an external one.
- **CsmVideoDxe.efi** — video driver that provides a wider range screen sizes than the built-in UEFI, is needed for option B if the video card does not have UEFI VideoBIOS. There are options when this driver is incompatible with the video card, for example, with the built-in Intel, be careful. The driver requires enabling the CSM Enable checkbox in the BIOS.
- **GrubNTFS, GrubEXFAT, GrubHFSPLUS, GrubUDF**… - set of drivers for a variety of file systems adapted from GRUB sources to work as part of Clover. It is very gratifying that we now have support for all these file systems, and in a license-clean manner. Mmm… about purity. They have a GPL license of some version. That is, open source, but you cannot use them in other projects. For example, if you compile with Clang, an artificial stop occurs "These codes are not intended for use in commercial projects". Well, without thinking twice, I deleted this check… Generally, thanks to AnV, who got hold of and adapted the source codes. They have some advantages over our drivers, for example, GrubHFS supports compressed volumes, but the status of this entire set is rather "beta". They are slow and

buggy. And the range of features is not impressive. HFS - no support for links, UDF - does not read all headers, EXFAT - no writing (and I would like it that way!).

- **AppleImageCodec.efi -** PNG and BMP file decoder, needed for FileVault2. In Clover 5150 it is already included.
- **AppleKeyAggregator.efi** — creates a special protocol for entering a password in FileVault2 interface. Also included in the 5150.
- **AppleUITheme.efi** — creates a protocol by which FileVault2 draws the background
- screen image (in Sierra, for example, these are blurred mountains). Also included in 5150.
- **AppleKeyFeeder.efi** – required to enter a password in the FileVault2 interface when using a PS2 keyboard. There is an alternative AptioInputFix.efi by vit9696. If you use FileVault, this driver is required. Otherwise, it is better not to install it, it conflicts with Shell.efi.
- **FirmwareVolume.efi** — creates the FirmwareVolume protocol, which, however, is not used anywhere except FileVault2, which expects to find a cursor image there. Yes. It is there. Also included in 5150.
- **SMCHelper.efi** — creates the AppleSMCProtocol protocol, which should interact with the SMC chip and receive information about the state of the peripherals from there. We have made an emulator that returns the values that are expected from it. Filling with real content is done by both Clover itself and FakeSMC, which saves keys in NVRAM, if FakeSMC with such a function is installed. All this is primarily necessary for FileVault2, but also affects the system without it. Return of keys to the system is not provided, since there are doubts about the expediency of this. Also included in 5150.
- **HashServiceFix.efi** — creates protocols of the HashService group if they are not present in UEFI. FileVault2 interacts with them, which is visible in the logs. But I did not see any visual effect.
- **APFS.efi** - a driver for a new file system that appeared in 10.13. This driver does not have an open analogue yet, so we use what is in the installed system from Apple at /usr/standalone/i386/apfs.efi, or →
- **ApfsDriverLoader** — an open alternative to the APFS driver. Thanks to savvas and other members of the acidanthera group. Its essence is to load the required APFS driver simply from the container, thus always having the correct version. However, if we load an old system, we get the old driver, and cannot read the APFS partition with the new system. This is more about the apfs.kext kext, but the problem is similar.
- **apfs_aligned.efi -** this driver, originating from Sequoia, turned out to be necessary when installing the Tahoe system, instead of other APFS drivers, because FileVault is being forcibly enabled, and the system password is not accepted due to the apfs driver! Apple's bug, of course. There are two solutions: install apfs_aligned, or prohibit FileVault with patches.
- **AudioDxe** — sound driver, developed by Goldfish64. Can be used to play sound when starting the computer. However, this is decoration, if such sound is not needed, then you can live without the driver. A bug has been noticed that after this, if the sound disappears in the system itself, then just delete this driver. It was notices with VoodooHDA 3.0.2 and resolved with VoodooHDA 3.0.3 and up.

Loading kexts

In the folder structure we see a common folder /EFI/CLOVER/kexts/ with folders 10.7/, 10.11/ and so on, as well as folders named Off/ and Others/. Let's consider the issue in more detail. Kexts are "Kernel Extensions" for the MacOSX operating system, most of them are drivers for some devices. Or a pseudo-driver for a non-existent device, like FakeSMC. We load them because

the pure Apple system simply does not support our devices, or we want to introduce new functionality. We are talking specifically about MacOSX here, no kexts are loaded for Linux and Windows.

Starting with revision 3281, all kexts from the Others/ folder are loaded first, and then from folders with a specific operating system number. This makes sense, since most kexts are not tied to a system version and work equally with any of them, FakeSMC in particular. Specific folders may be needed if the driver is loaded only for a certain system. VoodooHDA, for example, can be loaded with Clover for Mojave, but not for Monterey, there we load it in a different way directly in the system. Options: First, if you write the menu manually (Custom -> Entries), then for each item you can describe InjectKexts=Yes/No/Detect.
Secondly, there is a general setting for all

```
<key>SystemParameters</key>
<dict>
	<key>InjectKexts</key>
	<string>YES</string>
```

Another point. In Captain, an additional check for unsigned kexts has appeared, to overcome it, it is necessary to disable SIP (System Integrity Protection), this is done in the config in the CsrActiveConfig key

```
<key>RtVariables</key>
<dict>
	<key>CsrActiveConfig</key>
	<string>0xBE7</string>
```

Only the 0x01 bit (ALLOW_UNSIGNED_KEXTS) matters if you have unsigned kexts. I have all my kexts signed and can boot with zero. There is new information. You should never set the 0x10 bit. Updates will not boot. Starting with revision 4233, theClover menu has the ability to block loading of individual kexts from the /EFI/CLOVER/kexts/* folders. Thus, you can put a new, questionable kext there and try to boot. If not, then when you try to boot again, you can simply block this kext. And starting with revision 5052, you can put a questionable kext in the /EFI/CLOVER/kexts/Off/ folder, for example, VoodooPS2 from Vasya Pupkin, who promises that his kext is better than yours. And your kext is in the Other folder. After loading into the Clover menu, press Space, and there you will see a menu for disabling kext injection. Kexts in the Off folder are disabled by default, and in the Other folder they are enabled by default. To test, you enable this new VoodooPS2, and disable the old one. If nothing works, then the old one will work again upon reboot. If everything is fine, and Vasya is right, then you can manually change where each kext should be located.

Development

The project has no commercial significance for licensing reasons, and is also too large in volume to do it alone, so the most reasonable decision is to make it open source. For 4 years, the project was based on the sf.net server in the repository http://cloverefiboot.sourceforge.net/ , and let everyone who wants to contribute. Until 2019. With version 2.5k, we moved to a new repository at
https://github.com/CloverHackyColor/CloverBootloader
where compiled versions are also located,
https://github.com/CloverHackyColor/CloverBootloader/releases

Lyrical digression. Creating a project, especially such a large one, requires work on the following points:

- Collection of documentation, datasheets, specifications, sample programs for the task at hand, and also information about hacker finds. I had a good starting point — Chameleon, in which “everything” works (in quotes, however!). True, time goes by, and new processors, new video cards, new requirements for new versions of OSX are added, and you need to look for descriptions again, or do new tests. The starting point was systematized by Kabyl, my bow, first of all, to him..
- When there is a problem statement, input data and what you want to get at the output, you need to write an algorithm, preferably compact and fast, preferably error-free and safe. Programmers love such tasks, and most of the bows in this project are given to such assistants. I would especially like to highlight Dmazar and Gyk, they did really difficult things in the first days of the bootloader formation.
- The time comes for tedious and difficult, but not difficult (?) work on writing thousands lines of code where you don't really have to think, copy-paste with corrections. But here, Open source projects do, however, have some limitations in use, such as say "licensing". The GPL license is annoying, Intel does not recommend using it. Clover goes like BSD. Sorry, almost no one helped me for the past three years. Your project is you and work hard. Only Apianti contributed somehow, special thanks to him. Such rhetorical question: if I adapted someone else's source code to my project, how big is my contribution? We write from samples anyway...
- Next comes the work of testing and identifying errors. I have a lot of work here assistants, you can't even remember them all. 20,000 posts on applelife.ru alone. Testing can be different, from simple whining that nothing works, to specific instructions on what needs to be changed in the code. All of this is useful, even the whining, because makes you think about how to make it not exist (it doesn’t matter whether you solved the problem or No, the whining needs to stop!).
- The worst kind of error is a fundamental problem. The first three problems I I solved it alone for more than half a year: 1. KP in junior systems, 2. No sleep, 3. No start on a laptop. The following problems were solved together with Dmazar and Pene: 4. iCloud, 5. iMessage. And another problem is that legacy-boot hangs. I don't see anyone willing work, although there are those who want to see the result. 2018: about legacy-boot you can forget it already, it's not needed..
- There are also some additional things worth mentioning: compilation scripts, installer, system scripts and applications that are directly related to the project, although and are not part of the bootloader. All this huge work practically lies outside my competence, although I started. The main contribution here is Jadran, Crasybirdy, JrCs, and apianti contributed. Vector-sigma also left. Chris1111 and LabyOne are still making some contribution.
- Jief Machak also did a huge amount of work on translating to C++ and merging with OpenCore. And I learned from him how to use C++ correctly..
- Well, the program should be documented. Again, painstaking and not too interesting work. Here special thanks to xsmile for translating this book into English, and in this form it entered WIKI, where different people from time to time contributes... Now I made my own translation.

A special word about Dmazar. He is not just a contributor, he is the second author. Clover in its current form is not my original work, without Dmazar it would be just a legacy bootloader. Dmazar made UEFI boot. He owns the following technologies:

- **OsxAptioFixDrv** with all options, otherwise UEFI does not load macOS. Note that the operation of this driver is still unclear to any programmer, only the author understood it; vit9696 came, figured it out, and made a new version of AptioMemoryFix, with NVRAM support. And now renamed again to OpenRuntime;

- **DmpUefiCalls**, without which we would not be able to find out what MacOS needs;
- **EmuVariableUefi** - for those who do not have a hardware NVRAM, and there were such most. Without this driver on some configurations and boot impossible.
- and also working with **NVRAM** inside Clover;
- **kext and kernel patches.** Initial sources from pcj, who threw them in and ran away, and Dmazar brought it to perfection without even putting his own copyright;
- **FSInject** is needed to patch kexts. Again, no one knows how it works anymore;
- **OsxFatBinaryDxe**, needed for systems before 10.8, because those systems had boot.efi FatBinary, which UEFI does not understand;
- **hibernate**. We worked together, but it was Dmazar who pointed out how this could happen can be achieved;
- and in general helped me correct semantic errors in my codes. That's right, щпе in the field is not a warriorю

The rest of the developers and contributors are mentioned in one way or another in the source code and in the installer. I am writing this chapter in the hope that there will be others who want to work on the project, and for this you need to learn how to at least compile.

Tools

In order to compile a project, you also need a compiler and libraries – a simple truth. What in this case? The role of libraries is played by the source modules of EDK2. What is the correct name for this? Framework? Environment? It is downloaded from the same server. http://sourceforge.net/projects/edk2/ or https://github.com/tianocore/edk2

Starting with the modern version 2.5k and this has changed dramatically. All necessary libraries of the environment are now included in Clover, and its repository is complete for compilation, no external links are needed anymore. And the gcc compiler is again recognized as the best option, now version 14. Nevertheless, the EDK2 sources still exist in their repository and are improving, we should sometimes look at them, compare them with ours. Recently, for example, they found an incorrect use of the handle argument, writing &handle, they fixed it and not everywhere. The error is barely noticeable, but OpenCore does not pay attention to it, perhaps EDK2 themselves will fix it someday.

Writing codes

If there are those who want to, you can join the team of programmers and offer your piece. We always go to meet you halfway, although we have certain requirements for the codes. Initially, the project was done in the C language, formatted according to special rules
Tiano Code Style.
https://github.com/tianocore/tianocore.github.io/wiki/Code-Style-C
However, with the transition to C++, we do not want to use so many macros, and prefer to use the standard programming style. It should be understood that Tiano's requirement is the ability to compile sources on any platform, with the weakest C compilers. And in general, EDK2 is not a working project, it is just a set of sources for building your project. But we are doing a real project, and we set our own rules for powerful compilers and modern hardware.

Regarding C++.
1. By collecting structures and functions into logically united classes, we simplify the code and make it more visual. Moreover, it turns out that many techniques are reused, so this is a direct path to reducing codes. Code Reusing.
2. You don't need to declare variables in advance, as the C language required. A variable should be declared where it is used. Firstly, it is more visual, you don't need to scroll back the text to see the declaration, secondly, the compiler can arrange additional optimization, thirdly, it reduces the programmer's errors on repeated declarations, which are not prohibited by the language. An example is the for (int i=0; i<0; ++i) loop. In the old C, this was impossible, and therefore prohibited in Tiano, but it is better for us to declare the loop index this way.
3. Allocate/Free nightmares can be avoided by using the compiler's automatic constructor/destructor memory management capabilities.
4. Stop using arrays in favor of vectors. Arrays don't have bounds control, while vectors do.
5. Refuse to copy pointers - the biggest source of bugs in the project. Although it allows you to save something. Use links or copy structures entirely. It seems expensive, but it pays off with flawless work.
6. Avoid retyping. The notation (INTN)N is a logical error. If the variable N is used in a signed expression, that is, one that can become less than zero, then why was it declared unsigned? And similarly with pointer types. C++ discourages such type substitution, and we require that type substitution be used as little as possible.
7. Use constant variables and functions more, that is, we tell the compiler that their value will not change, the compiler will then apply a different code accordingly. And for the programmer who violates the ban, there will be a warning. This is called security, and it is supported by the compiler.
8. We don't have an STL library, but we create our own classes and translate all the source codes into theirs. usage.
9. Tiano Code Style implies the use of CONST, VOID, STATIC written in capital letters. Why??? So that the language would be more similar to Fortran, or what?! We tend to use the more standard spelling of **const**, **void**, **static**, and so on.

Compilation

Now to business. The reader who has looked at this chapter cannot by definition be a simple user, and he certainly does not need to be told how to use the terminal.
0. Preparing the system for compilation.

```
xcode-select --install
```

Install Python 3.12.x in any way — currently the latest version.
Make a link to this Python in this way

```
cd /Library/Frameworks/Python.framework/Versions/3.12/bin/
sudo ln -s python3.12 python
cd -
```

And another thing

```
pip3 install setuptools
```

1. Download Clover sources

```
cd ~
mkdir src
cd src
git clone -recurse-submodule https://github.com/CloverHackyColor/CloverBootloader.git
cd CloverBootloader
```

2. Build the compilation tools

```
./buildgettext.sh
./build_gcc14.sh
```

`./buildnasm.sh`
However, you don't have to build NASM yourself, just download the latest release from the official website
http://www.nasm.us/pub/nasm/releasebuilds/
currently version 2.16.01, and place it in the `~/src/opt/local/bin/`
If you forgot any of this, the Clover compilation script will do this for you

3. Build Clover either partially or completely, the
`./buildme`
will ask you what exactly you want. Vector Sigma made a menu there with a choice from a test build to a full release.

Starting with revision 3059 it was possible to reduce the size of the boot file (CloverEFI) by shifting the Translation Page Table from address 0x90000 to address 0x88000. The file size was 483 KB, became 450 KB, and at the same time it works more safely on some chipsets. Actually, Zenith432 had a problem that Clover did not work on his laptop, and he found the overlap of addresses. And since the code size has significantly decreased with LTO, it became possible to shift the TPT. Now Clover works for him too! Compilation with the key
`./ebuild.sh --low-ebda`
Now this key is the default, to cancel it, you need the key `--std-ebda`.
It is also interesting for developers to know where the letter "B" or "6" is formed, which is displayed on the screen when CloverEFI starts. Using a hexadecimal editor, we look for offset 0x02a9, there we see byte 0x42 or 0x36, respectively. By the way, you cannot cancel this letter, since this is a trick for Clover to start at all. The BIOS initializes the video adapter at this point. You can change it to a space 0x20.

Making DEBUG version of Clover

When people complain that Clover does not start and therefore cannot provide any reports, I cannot help except for a couple of standard tips. But a person who can edit the codes and compile a little will be able to get to point X, and then we will jointly determine what is wrong.
Option 1. During legacy loading, it hangs on message 6_. This means a hang inside Duet itself. It is impossible to create a log. It is possible, perhaps, to display messages on the screen. The sequence of Duet's work is as follows:
1. Start sector st32_64.s. In it, the output of this very number 6 occurs, the BIOS memory card is read, A20 is switched, and the processor is switched from 16 bits to 32 and 64. My version of the sector differs from the vanilla one in that it works on a laptop. It is possible that for someone there will be problems here too. You need to insert the output of other numbers in other places and observe.
2. Start of C codes. File efi32.s or efi64.s. It is unlikely that anything will slow down here.
3. File Efiloader.c. Here you can insert output to the screen using the PrintHeader('A'); procedure, which is currently commented out.
4. File DxeIpl/DxeInit.c. You can also insert PrintHeader, but also include Debug.c in the compilation of this module.
5. DxeCore. Here the execution is already spreading, and it is more difficult to track where and what. Output to the screen can be done in the same way.
6. The CLOVERX64.EFI itself is loaded in the `BdsBoot.c/BdsLibBootViaBootOption()`.
In this place of the program for output to the screen you can already use the standard AsciiPrint("Hello!\n");
Option 2. Duet itself works, which can be checked by pressing the spacebar immediately after displaying 6 on the screen. Either we have a UEFI boot, and there is no Duet, we should have

CloverGUI running, but it is not there, or it is there, but it hangs. The standard method Boot->Debug=true does not suit us, because we need to track the place in more detail. In this case, in the Platform.h file, remove the comment in the 11th line

```
//#define DEBUG_ALL 2
```

or in the files of interest in the upper lines, put DEBUG_xxx 2. In this case, the entire output from the DBG command ("Nightmare ÿ3\n"); will go to the screen. And in this way, you can interactively observe what place the program execution reaches before it will hang.
With the transition to C++, Russian letters can be used in debugging ещщю
Option 3. Compile the debug version with breakpoints, run under control QEMU with a special version of gdb installed. Dmazar once tried this way. In my opinion, these efforts are not worth the goal. Simple tracing is always enough.
Option 4. When compiling, specify

```
./ebuild.sh -D DEBUG_ON_SERIAL_PORT
```

and connect another computer to the serial port, capable of receiving letters on the serial port (terminal in Windows). This option works on QEMU if you specify the "-serial stdio" flag when starting.

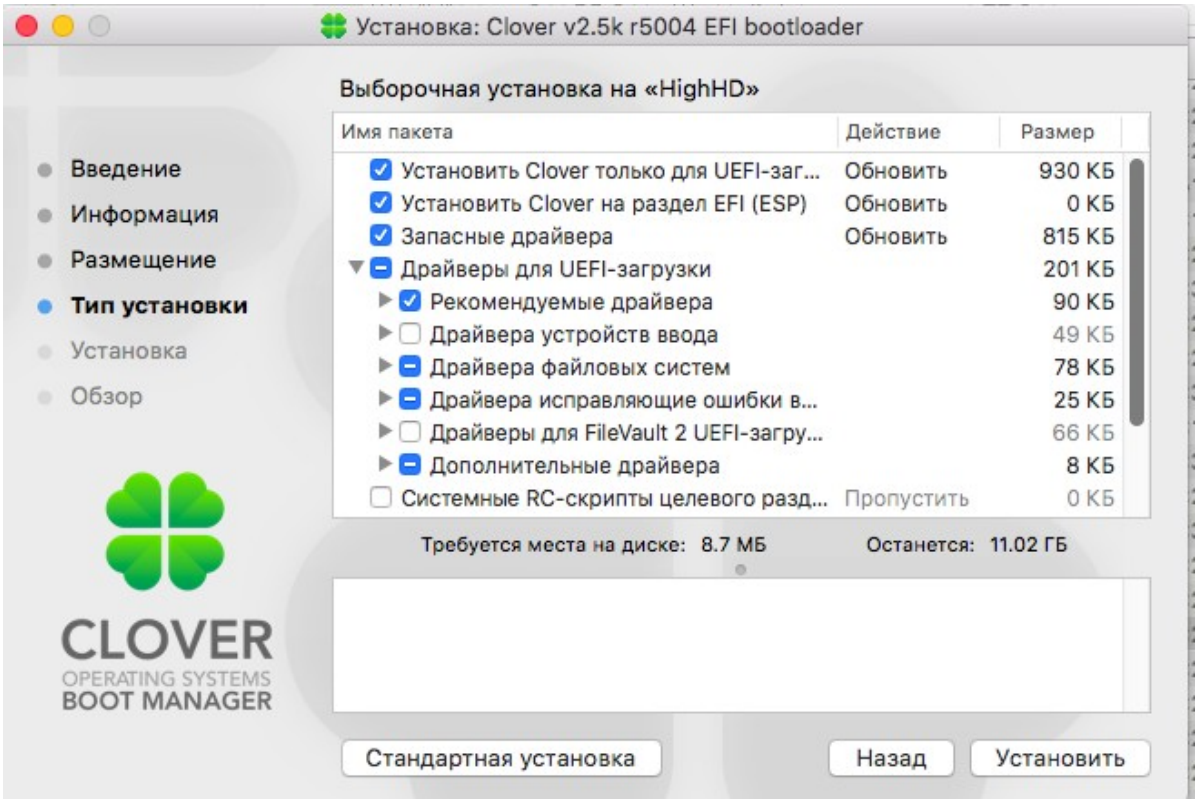
Installation

Using the installer

What is the installer for? To install the program! Why do it manually, the installer will do everything more accurately than you yourself! The only condition is that you already have MacOSX on this computer. One of the options is that you launched the installation DVD with a different bootloader, and launched the installer from the MacOSX installation interface. Depending on the OS language, the installer will work in Russian, English, or even Chinese. Here are the instructions for the English version, since you can figure it out in Russian anyway, and I don't even know Chinese. The current version has 20 languages, including Indonesian, maybe someone needs it.

So, we follow the Continue and OK buttons, read and agree with the license agreements (hmm, are they there?), and come to the choice of what we install, where and why..

Change Install Location – select where exactly to install the bootloader. If you intend to install it on the EFI partition, then simply select the partition with the current system. MacHDD in this example. And check the box "Install Clover in the ESP". The installer will find the EFI partition on the same disk.



Customize or Standard Install – choosing boot options If you place the cursor on one of the lines, then in the lower field there will be a short description of this option.

Install Clover only for UEFI boot - this option cancels the installation of boot files. In the English version it is written Install for UEFI motherboard, and users, of course, check this box. As a result,

as usual, Clover does not boot. I really regret that the installer developers came up with this item for their own purposes. People, for God's sake, do not check this box, even if you have a computer with UEFI BIOS. The UEFI boot you are interested in will still work!
Install Clover in the ESP (Install Clover on the ESP EFI partition) is the best option when such a partition is present (GPT partition scheme). The installer does not see this partition, so in the disk selection menu we specify the partition that is on the same disk, on the ESP of which we want to install the bootloader. We assume that this partition contains MacOSX, where scripts, a control panel and an updater will be installed. Before installation, you should unmount the ESP partition. And for Catalina, before running the installer, you may need to give the command

```
sudo mount -uw /
```

, which will allow scripts to write files to system folders. If the system writes that the installer cannot be opened because its author is an unknown developer, you can go to system settings, Security, and click the “Open Anyway” button. Or you can run the command in the terminal

```
sudo spctl --master-disable
```

However, it is not my business to teach you tricks of using macOS.
Bootloader is a variant with BIOS (variant A), which uses CloverEFI, or with UEFI (variant B).

- **Don't update MBR and PBR sectors** — do not update sectors because they are already yes, or just for option B;
- **Install boot0af in MBR** – boot using boot0af, i.e. search for active section. The installer will make the selected section active. The exception is the installation on the EFI partition, it is not made active, and boot0af will not find an active partition load the boot file from the EFI partition, which is what we need to implement legacy boot from GPT disk, from ESP partition.
- **Install boot0ss in MBR** – boot using boot0ss, i.e. search for the HFS+ partition, even if it is inactive. The installer does not change the currently active partition. This made for configuration with active Windows partition - it needs it.

CloverEFI - this is, as you can see from the list, a choice of bootloader bitness. Either 32 bits or 64 bits. Also here is a special variant of BiosBlockIO. This is a variant of CloverEFI-64, which has a special name **boot7**, and is intended for computers with non-standard SATA controller. This driver works via BIOS, and, as a rule, works with any controller (BIOS should work with them!). But there are also misfires, for example, Dell Inspiron 1525. Another special variant is boot7-MCP79. It's like the only working option for the MCP79 chipset, however there is nothing specific there for this chipset, may work in other cases. This option was found by Oscar09, also offered a USB patch for this chipset.
Drivers. The choice of drivers is explained in the chapter "What is what"..
Install system RC scripts on the main partition - These are the rc.local and rc.shutdown.local, which is executed by the OSX system on login and logout, is a necessary part of the whole legacy-Clover concept. For those who have a working NVRAM for UEFI boot, these scripts are not necessary.
Install scripts on all other partitions - if there is more than one on the computer partition with MacOSX. The installer is smart enough not to install them on partitions with Windows or Linux.

Installing the bootloader manually

It is needed in two cases: Firstly, when a person is well knows what he is doing and wants to control every step, not trusting the installer (in vain!), and, secondly, when installing from under another OS, where running the installer is impossible.

OSX

It is highly inadvisable for someone who does not know what a terminal is to do this. Installing on an HFS+ partition in MBR or hybrid partitioning. Why MBR? This is a very standard situation when the computer already exists, and already has information, nothing can be lost, you can only install a new bootloader.

Installing the MBR sector

```
cd BootSectors
sudo fdisk440 -f boot0 -u -y /dev/rdisk0
```

What's in this command?

fdisk440 is a special version of the fdisk utility, modified to use only 440 bytes of the zero sector, there is information that this is necessary for compatibility with Windows (the wake-up problem), which Apple did not take care of.
boot0 is the file described above in the "What is what" chapter rdisk0 is the physical device you are going to install the bootloader on. Make sure that it really has the number 0. These files are supplied with Clover. In the new version, fdisk440 is excluded, primarily because it cannot be compiled in the new environment. Now we use dd instead, as shown for Linux. And you can make the partition active using the standard fdisk. But this is not necessary for GPT partitioning.

Installing a PBR sector s

```
sudo dd if=boot1h2 of=/dev/rdisk0s9
```

boot1h2 – PBR sector file for the HFS+ file system, differs from similar ones by supporting large boot files, and the ability to select boot1,3,6 using a hotkey. Details in the chapter "What is what".
rdisk0s9 – the ninth partition on the selected device... Why the ninth? And so that fools don't spoil anything by stupidly repeating the written commands, such a partition probably doesn't exist. And you need to set a real number, for example, the first partition. Well, after the MBR and PBR sectors are successfully written to the selected device/selected partition, you should make this partition active

```
fdisk440 -e /dev/rdisk0
>f 9
```

The digit Nine in the second line is again the partition number (there are only four!) - draw your own conclusion. Now you can copy the boot file and the EFI folder to the root of the partition to this partition.

Installation on a FAT32 partition.
Unlike the previous method, there is one subtlety here. The PBR sector must contain the partition geometry. This information is entered there during the partitioning process, so losing such information is fraught with consequences. The sector installation method itself is complicated

```
dd if=/dev/rdisk1s9 count=1 bs=512 of=origbs
cp boot1f32alt newbs
dd if=origbs of=newbs skip=3 seek=3 bs=1 count=87 conv=notrunc
dd if=newbs of=/dev/rdisk1s9 count=1 bs=512
```

boot1f32alt - already mentioned in the chapter "What is what" - a sector for installation on a FAT32 partition. But not FAT16! Be careful!
rdisk1s9 - again the ninth partition on the first device. Substitute your numbers. The remaining letters and numbers in this recipe are not subject to discussion and revision. The remaining steps are similar to installing on HFS+. For owners of hard drives with a sector size of 4k. Attention! In the first and fourth commands, instead of bs=512, you must write bs=4096.
P.S.
However... this is not a fact! For me, with a 4k disk, installation with 512 worked perfectly.

Installation on exFAT
partition Starting with revision 3040, it became possible to install Clover on an exFAT partition. For example, I have an external HDD with this format, I can no longer repartition it, and there is nowhere to install Clover the old way. To install the PBR sector, you need a special utility boot1-install, written by Zenith432. The command is simple

```
./boot1-install -u -y /dev/disk2s1
```

In principle, Zenith made it so that both HFS+ and FAT32 are also installed by this utility. It is important that the corresponding sectors (boot1h, boot1f32, boot1x) are in the same folder. For EXFAT, the sector is called boot1x. The option with a pause is boot1xalt. There is, however, one more thing to remember when installing these sectors. The volume where you want to install the sector must first be unmounted! And this is only possible if the disk is not a system disk. Otherwise, boot from another drive to perform this procedure.

Linux

Under Linux there is also a terminal, and almost the same commands, but installation is possible only on FAT32. The differences are as follows.
- instead of rdisk1 there will be sdb - look in your version of Linux more precisely.
- instead of fdisk440 to write MBR you need to use the same dd

```
dd if=/dev/sdb count=1 bs=512 of=origMBR
cp origMBR newMBR
dd if=boot0 of=newMBR bs=1 count=440 conv=notrunc
dd if=newMBR of=/dev/sdb count=1 bs=512
```

Windows

From Windows, it also makes sense to install the bootloader only on a FAT32 flash drive; to do this, simply run the

```
makeusb.bat E:
```

where E: is the letter of your flash drive. It is not supposed to have multiple partitions. This is Windows! All files needed for the script are included with Clover. However, the installer must first be unpacked, or these files must be obtained directly from github. They are located in the folder

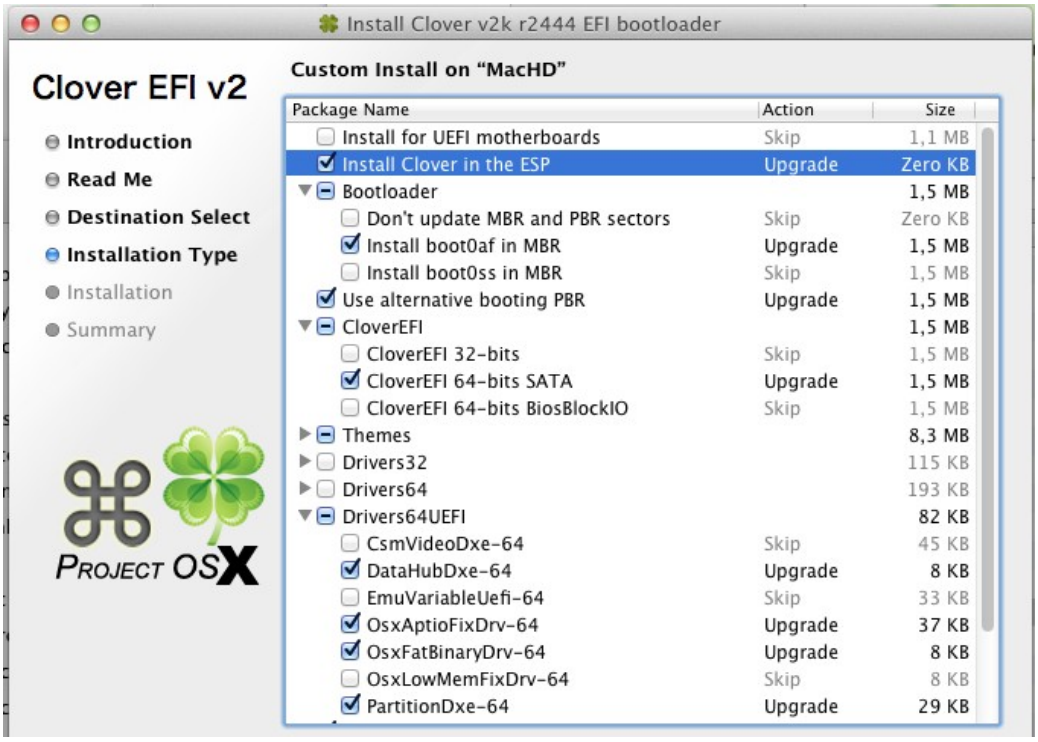
```
/CloverBootloader/CloverPackage/CloverV2/BootSectors
```

After running the script, the flash
drive must be removed and reinserted, then copy the boot file and the EFI folder to it. It is even better to use **BootDiskUtility.exe by Cvad,** which will help create a flash drive from under Windows.
http://www.applelife.ru/threads/delaem-zagruzochnuju-clover-fleshku-s-macosx-iz-windows.37189/

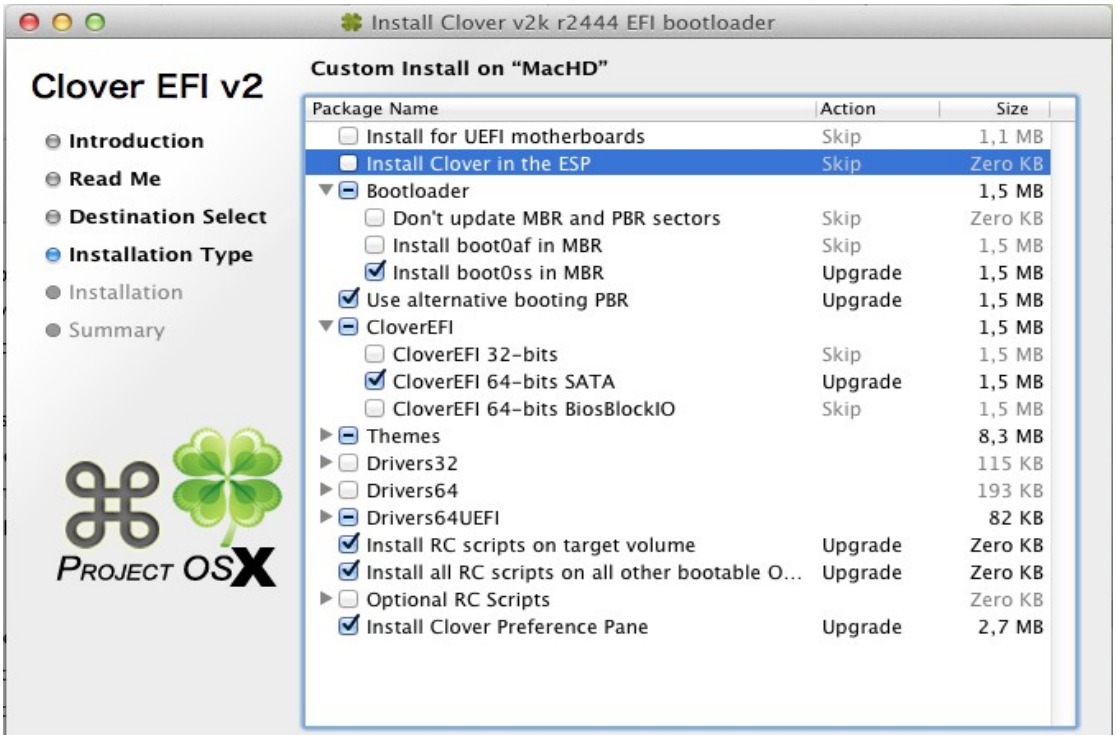
Recommended installation options

I would personally recommend these options for installation:
1. The main hard drive is partitioned in GPT (GUID Partition Table). This is a modern scheme, the most correct for Mac OS, and supported by Windows starting with 7. In this scheme, there is an invisible ESP (EFI System Partition) partition of 200 MB. By default, it is already formatted in FAT32 and there is no need to reformat it, it is even harmful. This is where Clover should be installed, both for legacy and for UEFI boot. For UEFI boot, you just need to copy (or install with the installer) the EFI folder with all the necessary contents. But in order for the BIOS to see this boot option, you need to copy the file /EFI/CLOVER/CLOVERX64.EFI to the file /EFI/BOOT/BOOTX64.EFI, if there is none. For those who do not understand, do it in two steps: copy it to the /EFI/BOOT/ folder, and then rename it to BOOTX64.EFI. For legacy booting, you

need to write the boot0af file to the MBR sector of this disk, and to the PBR of this EFI partition write file boot1f32 (or boot1f32alt with a pause). Do not set any partition as active under any circumstances! The Dell laptop has American Megatrend EFI BIOS, and it does not allow legacy booting on a GPT disk. Alas! For others, there is such an option, and in the installer we do this: (the screenshot is outdated, but the meaning remains). Attention! You need to format the disk in the GPT scheme using the MacOS installer. Windows does the wrong GPT and not the right way, after it you need to change everything.



2. The hard drive is already partitioned as FDISK Partition Scheme, commonly called MBR. An old scheme, always used for Windows XP. On this disk, we have allocated a partition where we want to install OSX for testing. We are not talking about UEFI boot here, but we can use legacy Clover. To do this, write the boot0ss file to the MBR sector. This sector will look for the HFS+ partition that we have made for Mac and transfer control to it. Windows can remain the active partition, it needs it more. You need to write the boot1h file from the Clover kit to the PBR sector of this partition.
In the installer set the following checkboxes

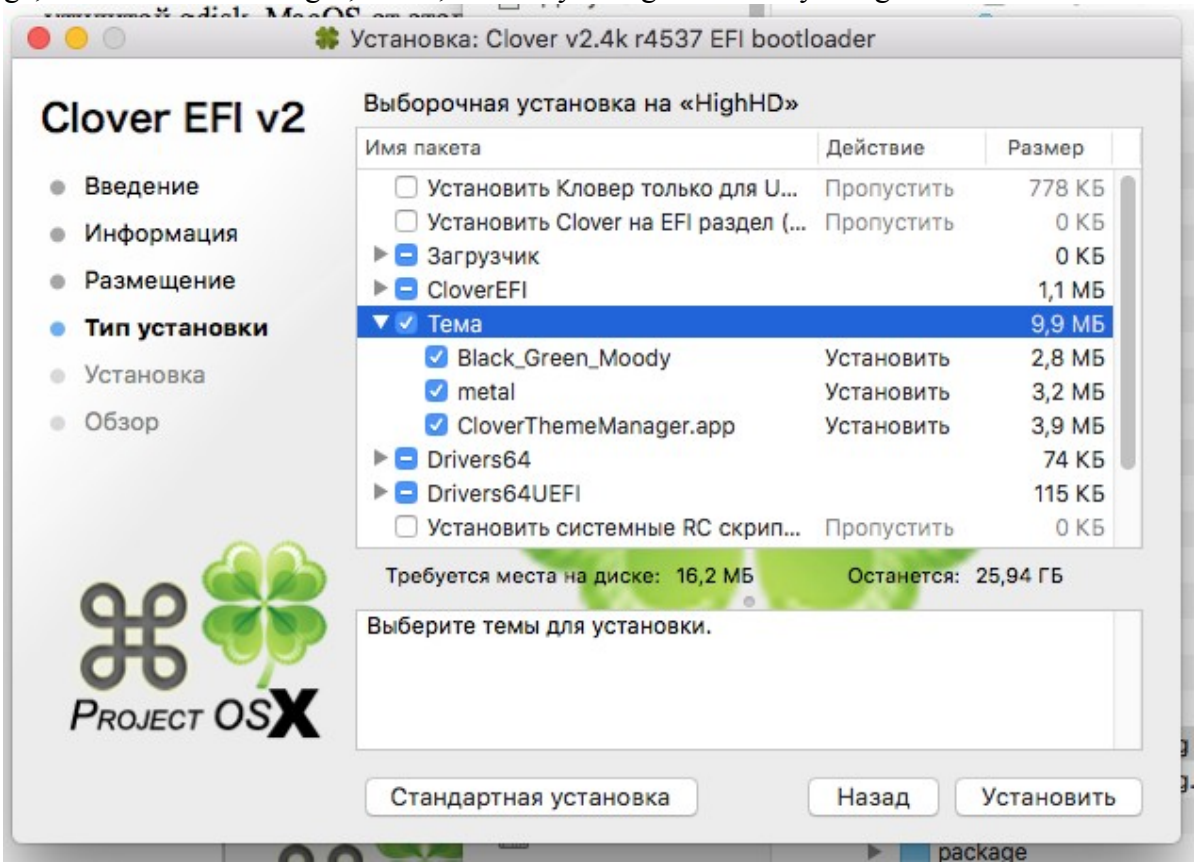


3. You have decided to allocate a separate hard drive for the Mac. Here you can simply choose what to load using the BIOS, and prepare the disk according to the first scenario. Or you can install the boot0md file in the MBR sector of the main hard drive. Its property is that it will search for an HFS+ partition on all hard drives, and in this case it will work like the second option. Now look again at the chapter on manual installation. And finally. Isn't it time to completely abandon Windows XP and install Windows UEFI, for example, the seven? The MBR disk can be converted to GPT without losing data using the gdisk utility. MacOS will only benefit from this, and we reinstall Windows one way or another every six months. If anyone didn't understand, Windows UEFI is perfectly installed by Clover on a machine that does not have a UEFI BIOS.

Design

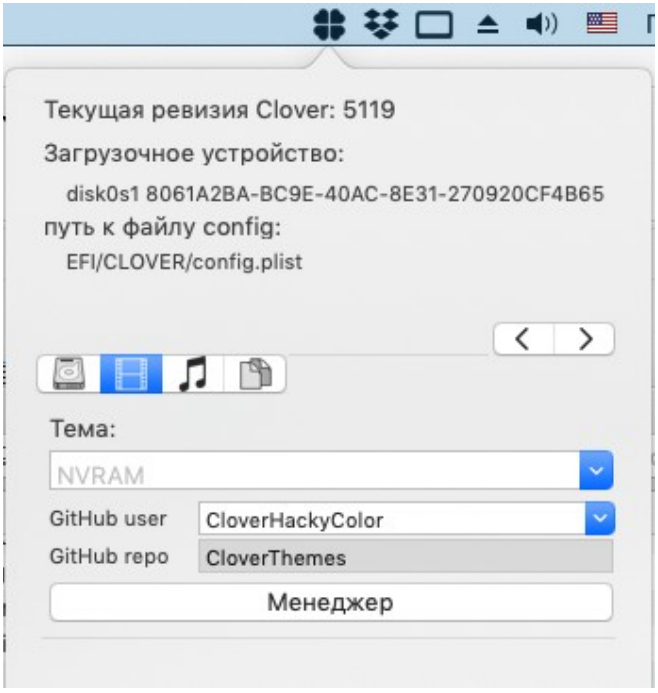
Choosing a theme

Now we choose a theme. What is a theme? These are design elements: a banner, a background image, icon and button images, a font, united by a single artistic by design.



There are four vector themes in the installer, my cesium, a theme from Clovy, Purple_Swirl from Pkdesign, BGM from BlackOSX. In addition to them, there are "Easter eggs", those who saw them know, for the rest it will be a surprise. And Clover also has a built-in theme from Clovy, called embedded. If you do not install any theme, this one will still be available. And there is an application that will download any theme for you online from 40 available on git. I will not tell you more, everything is transparent there.
http://www.insanelymac.com/forum/topic/302674-clover-theme-manager/
or use the Clover.app application available on github

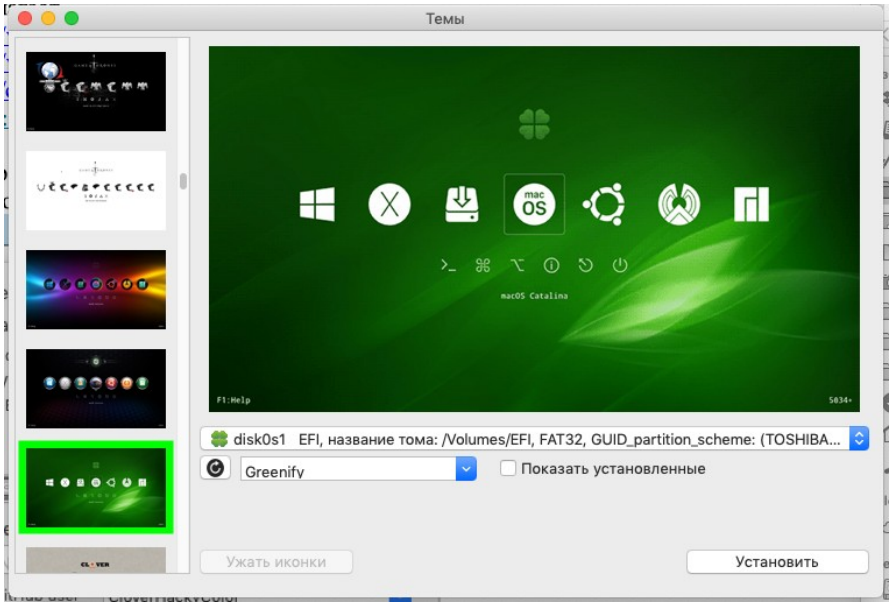
https://github.com/CloverHackyColor/CloverBootloader/releases/download/5134/Clover.app_v1.24.pkg



Clover Theme Manager inside the Clover.app application. The button looks like a Movie.
It is the latest version, posted in release 5134. It is absent from the following releases. In the modern version, you can specify the "random" theme and observe a different design with each subsequent reboot. Let's make a brief overview of the themes. In reality, the range of themes is much wider, look at the forums to see what people offer.

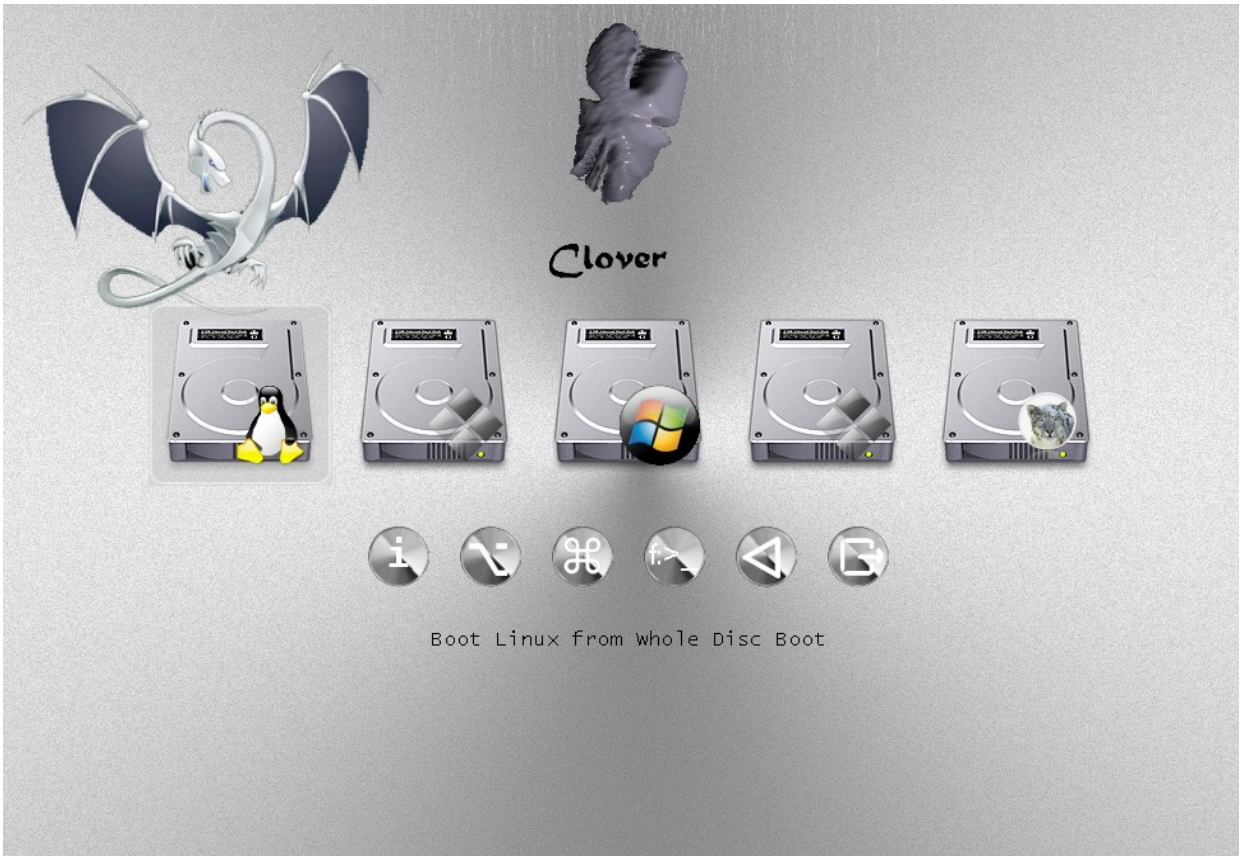
http://www.applelife.ru/threads/themes-temy-dlja-zagruzchika-clover.36074/
http://www.insanelymac.com/forum/topic/288685-clover-themes/
http://clover-wiki.zetam.org/Theme-database
https://sourceforge.net/p/cloverefiboot/themes/
https://github.com/CloverHackyColor/CloverThemes

Choose a theme



Clover Loader Themes

Metal. Author Slice. Historically the first theme with animation.



Black-green. Author blackosx.

steampunk. Author medik.

And dozens of others... Creating your own theme is a special story with its own rules.

Setting up the interface in config.plist

Themes also include a number of parameters specified in the config.plist file. For older versions see old versions of instructions.

Interface settings are made in EFI/CLOVER/config.plist in the GUI section

<key>GUI</key>

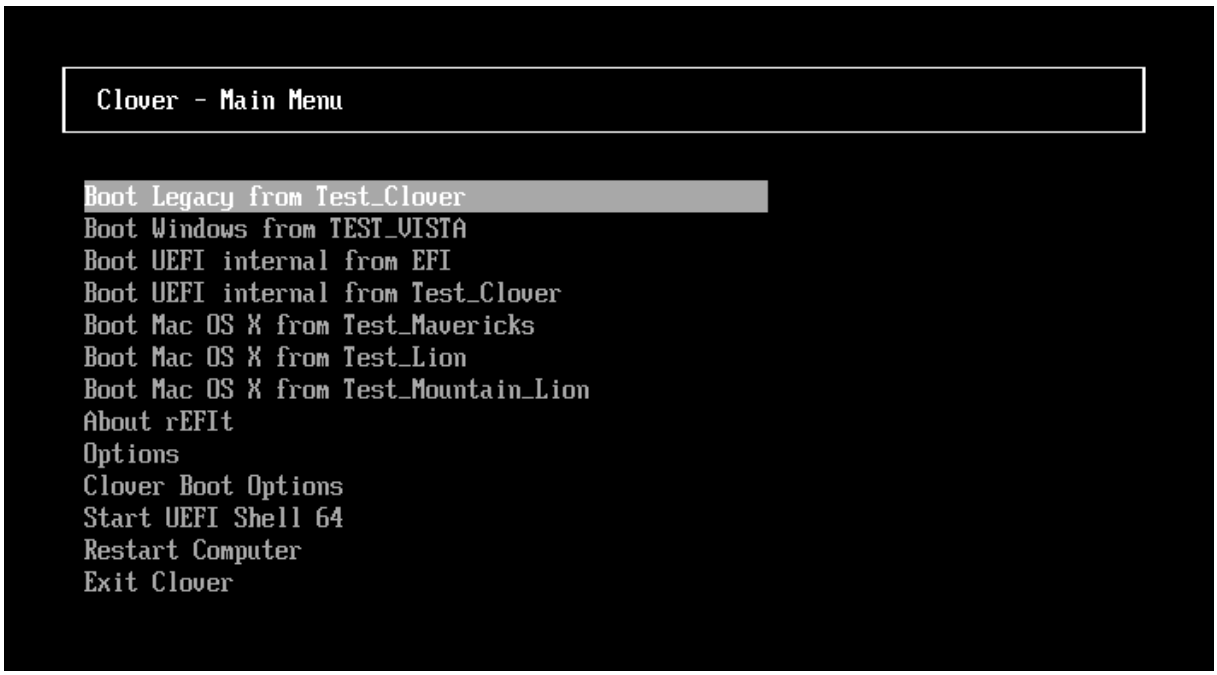
```
<dict>
```

The interface can be graphical, or it can be textual (starting with revision 1764). To do this, write

<key>TextOnly</key>

```
<true/>
```

Probably only in Russia live lovers of text interface,, Total Commander, Volkov Commander, DOS etc. For your pleasure!



If false, Clover works in graphical mode. The text screen resolution can be set here, as well as in Shell and on the boot.efi screen.

```
<key>ConsoleMode</key>

<string>0/Min/Max/some number</string>
```

You can find the number in your boot.log, or set it to Max - it will be the maximum possible resolution. 0 is the maximum allowed value by default.

The design of the graphical shell depends on the selected theme. The default theme is selected in the variable

```
<key>Theme</key>

<string>metal</string>
```

However, the theme can also be selected in the Control Panel, and that choice will be decisive. If the wrong theme is specified there (there is no such theme.plist or theme.svg file at the specified path), then the theme from the config will be selected. If a non-existent theme is specified there, then the built-in theme (embedded) will be displayed on the screen, the latest version from Clovy. The theme can also be changed in the bootloader menu, there will be a list of installed themes (rev 1955), and you can specify which one you need. The interface will be repainted after exit to the main menu.

random — the theme will be selected randomly from the list of installed ones at each boot. Another point in choosing a theme. If your monitor has a resolution of, say, 2560x1600, then the icons and font on the screen will be too small. You need to choose a large theme. Starting with revision 4438, Clover will do this automatically. On a small monitor, it uses a theme, say metal, and on a large monitor, it will try to find a theme metal@2x, if there is one. The criterion for a "large" monitor is the number of lines greater than 1100, because for me, a small theme is quite suitable for 1080 lines. A question for designers: when will they make large analogs of their themes? Starting with 4862, scalable (vector) themes have been made.

As for this config item, it looks the same for raster and vector themes, the only difference is in the name. Clover itself distinguishes one from the other by the contents of the folder. If there is theme.plist, then it is a raster theme. If there is theme.svg, then it is a vector theme.

<key>EmbeddedThemeType</key>

```
<string>DayTime</string>
```

Possible options Dark and Light. Introduced with revision 4644. The options were invented before the invention of the clock, you could only choose dark or light built-in themes. And now by the clock, light during the day, dark at night. Introduced with revision 4773.

<key>Timezone</key>

```
<integer>3</integer>
```

In the modern revision of Clover, 4773+, the concept of "time of day" is introduced. When starting Clover, it reads the current time from the BIOS, which is usually Greenwich Mean Time, and adds the value from this parameter, the zone, to it. You can write with the sign +3 or, say, -5. The sum is local time. Then Clover determines that if it is in the range from 8:00 to 20:00, then it is considered day, otherwise it is night. This affects the above-mentioned built-in theme, if DayTime is specified, this certainly affects any vector theme, it must determine what it affects, it affects the sound in any theme. During the day, the sound from the sound.wav file plays, at night from the sound_night.wav file. If the night sound is not defined, the day sound plays.

<key>PlayAsync</key>

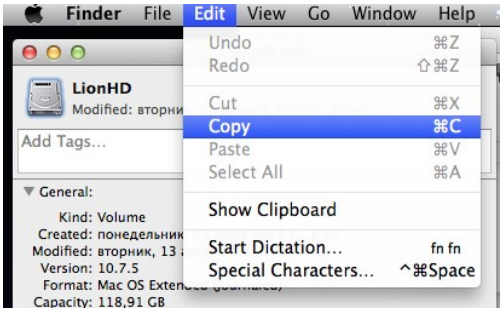
```
<false/>
```

Starting with revision 4833, sound appeared in Clover with the AudioDxe.efi driver, author Goldfish64, who doesn't want to deal with Clover, well, okay, he doesn't mind using its solutions. PlayAsync determines whether the sound plays synchronously with Clover's work, or asynchronously. That is, with synchronous playback, nothing works until the sound ends. With asynchronous, the sound flows regardless of what is happening on the screen. That is, the sound starts after the message on the screen ...scan entries... then the Clover interface appears, the sound continues, then the boot.efi work lines appear, the sound continues, then the kernel starts, and, the kexts are started one by one. Starting from one of them, the sound breaks off. Well, this is, of course, if the sound.wav file is long. Clover 4862 had a bug, asynchronous sound hung the computer. Starting with 4870 there are no problems, you can set PlayAsync=true.

<key>CustomIcons</key>

```
<false/>
```

set to <true/>, then for each partition with the operating system, the .VolumeIcon.icns icon will be searched for in the partition root and used instead of the icons specified by the theme. It is very convenient to create such an icon using MacOSX tools. Select the disk icon and copy-paste it



<key>ScreenResolution</key>

```
<string>1024x768</string>
```

can set the desired screen resolution, larger than the standard 1024x768, if the video card and the screen itself have such a mode. Clover tries to set the highest possible resolution, but it can make a mistake. Check the list of available modes in the boot log. If PatchVBios=Yes is in the graphics section, then you will have the maximum resolution available for this monitor. In this case, the ScreenResolution parameter may be redundant. With some configurations, the PatchVBios parameter can be fatal by a black screen with no signs of life.
There is one subtlety here. Clover must know the EDID of the monitor. Legacy Clover tries to get it through INT10 BIOS calls, often successfully, sometimes not. UEFI Clover requests data from the UEFI BIOS, which probably knows the EDID for the integrated video card, and probably does not know for the inserted one. Look at preboot.log, and if there is no EDID, enter it manually. Instructions below.

<key>ProvideConsoleGop</key>

```
<true/>
```

Creates a GOP protocol for console mode, i.e. so that the text output is not in text mode, as is customary to do in PC BIOS, but in graphical mode, as Apple does. In some revisions of Clover, this parameter was duplicated with the quirk ProvideConsoleGopEnable, but starting with revision 5128 this parameter has been removed from the list of quirks and is only in the GUI section with a slightly different name. The value of the ProvideConsoleGop parameter from the GUI section will override the value of the ProvideConsoleGopEnable parameter from the list of quirks, if you suddenly forgot to remove this parameter from the config when updating the bootloader. It is better not to turn off the parameter, there are no adequate arguments in favor of turning it off. Well, accordingly, why the hell do you need it if it is always on?

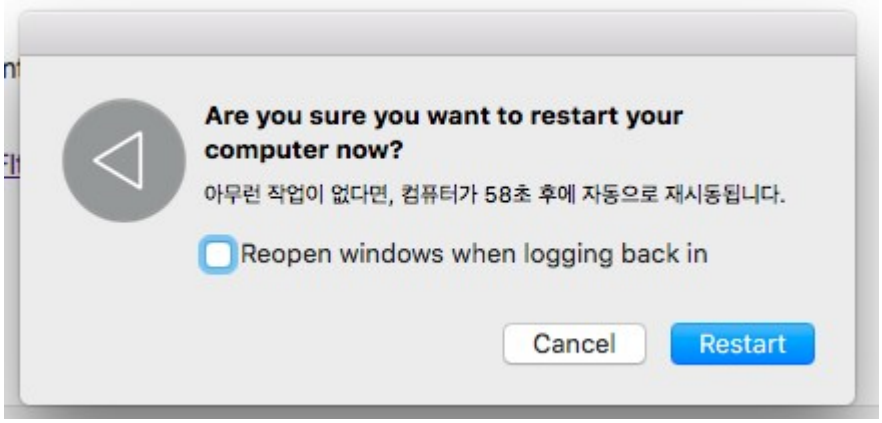
<key>KbdPrevLang</key>

```
<false/>
```

Enable if you want to keep the system language when updating macOS using the built-in NVRAM.
- this is the key to fix macOS language issues when using its own NVRAM.
- this macOS bug has been around for a long time.
- the fix only works when using AptioMemoryFix or OsxAptioFixV3 without EmuVariableUefi.efi, i.e. you need a hardware NVRAM
- fix works when using a language other than English You can fix the issues below using this key:

```
<key>GUI</key>
    <dict>
        <key>KbdPrevLang</key>
        <true/>
        <key>Language</key>
        <string>ru:0</string>          <----- you want language
    </dict>
```

1. Mixed language in restart popup after macOS update.



2. Always keep English language when updating macOS in recovery mode even if the user already uses another language in macOS. Added in revision 4719.

```
<key>Language</key>

<string>ru:0</string>
```

At the moment, setting the language only makes sense for the "Help" menu called by pressing F1. However, this value is passed to the system, and may affect the default language.

```
<key>Mouse</key>

<dict>
	<key>Enabled</key>
	<true/>
	<key>Speed</key>
	<integer>2</integer>
	<key>Mirror</key>
	<false/>
	<key>DoubleClick</key>
	<integer>500</integer>
</dict>
```

Enabled - there are configurations when the mouse does not work, or even freezes, well, then it can be disabled.
Speed 2 — the speed of cursor movement, reasonable values are 2 - 8. Some mice require negative speed, moving in the
opposite direction. Value 0
means the mouse is disabled.
Mirror - and also make the opposite direction only for one coordinate.
DoubleClick 500 — Obsolete.

n the Clover interface, you can see legacy and UEFI bootloaders for installed operating systems. At the same time, there may be several bootloaders on one partition. Maybe you don't need everything that Clover found, you just need to point to the real one. You can hide both individual sections and entire bootloader classes from the interface. The following sections in the config:
Hide — hide volumes by name or by their UUID.

```
<key>Hide</key>

<array>
	<string>WindowsHDD</string>
	<string>BOOTX64.EFI</string>
	<string>E223FF7F-F2DA-4DBB-B765-756F2D95B0FE</string>
</array>
```

This is an array of strings that are included in the full partition name, see boot.log. This way you can remove unnecessary partitions from the menu, such as Recovery or Preboot.
Or, conversely, specify what exactly to scan

```
<key>Scan</key>

<dict>
	<key>Legacy</key>
	<string>First</string>
	<key>Entries</key>
	<true/>
	<key>Tool</key>
	<true/>
	<key>Linux</key>
	<false/>
</dict>
```

For Legacy (i.e. loaders launched from PBR), there are options for the values
No, First, Last - do not show at all, place at the beginning of the list, or at the end.
Linux - don't search for Linux bootloaders on each partition, it takes a lot of time.
The interface can be customized more subtly, if you understand how and what to do. For you, the next section (also in the GUI section)

```
<key>Custom</key>
<dict>
```

contains arrays

```
<key>Entries</key>
<array>
<key>Legacy</key>
<array>
<key>Tool</key>
<array>
```

One array element contains a description of the selected item in the form of a dictionary.

```
<dict>
	<key>Volume</key>
	<string>454794AC-760D-46E8-.....2</string>
	<key>Type</key>
	<string>OSX</string>
	<key>Title</key>
	<string>OS X 10.8.5 (12F36) Mountain Lion</string>
	<key>InjectKexts</key>
	<true/>
	<key>NoCaches</key>
	<false/>
	<key>BootBgColor</key>
	<string>0x2C001EFF</string>
	<key>Hidden</key>
	<false/>
	<key>SubEntries</key>
	<array>
		<dict>
			<key>Title</key>
			<string>Boot OS X 10.8.5 </string>
			<key>AddArguments</key>
			<string>-v</string>
		</dict>
	</array>
</dict>
```

And each menu item can contain sub-items (SubEntries), which are different options for calling the main member. The code responsible for these parameters was developed by Apianti.

Description of internal keys:
• Disabled – disables the entire structure and it is not taken into account
• Volume – The volume name or GUID that will be used during scanning bootloader
• VolumeType - The volume type your custom entry should match. Can be a string of one type or an array of multiple types. Valid volume types are: Internal, External, Optical, FireWire
• Path - Location to scan for efi bootloader
• Arguments - Arguments to pass to bootloader. Overrides arguments
default boot in Boot/Arguments

- AddArguments - Adds default boot arguments. For SubEntry it adds arguments to the main record
- Title - Changes the display title. Uses the format "Boot <Title> from <VolumeName>"
- FullTitle - Sets the display title to just “<FullTitle>” without any format
- Image - Path to a custom image file. The search path is the root directory of the burning volume, the theme directory, the clover directory, the root directory of the clover volume, and finally the names of the OS icons
- ImageData – Built-in custom image. Can be in PNG or BMP format, defined in the config as a binary array. Path to custom drive image file. Search path is write volume root directory, theme directory, clover directory, clover volume root directory and finally OS icon names •
- DriveImageData - Built-in custom drive image. Can be in PNG format or BMP
- Hidden – Hides the entry. If true, the entry can be shown by pressing F3 in Clover menu. If the value is Always, the entry will never be displayed.
- InjectKexts – Injects kexts. Valid options are Yes, No, or Detect. And also some Entry can be set. Use Detect to inject kexts only if FakeSMC is not present in KernelCache or /S/L/E. For entries like OSX, OSXInstaller and OSXRecovery.
- NoCaches – Skip caches. For OSX, OSXInstaller and OSXRecovery Kernel type entries – Set the priority of Linux kernel scanning. If this option is not specified, Clover will show all found kernels. Valid options are: Newest, Oldest, First, Last, MostRecent, Earliest. For LinuxKernel type entries
- All - any kernel
- Newest - the newest
- Oldest - the oldest
- First - the first found
- Last - the last found
- MostRecent - the newest version Earliest
- Earliest - the oldest version

None - do not search for kernels.

- Type – The type of OS that is being scanned. Acceptable types: OSX, OSXInstaller, OSXRecovery, Windows, Linux, LinuxKernel. If the Type is not one of the listed, it will be treated as all types
- BootBgColor – Sets the background color of the boot screen. For OSX type entries, OSXInstaller and OSXRecovery
- CustomLogo – Specifies the style of the boot screen logo. For more information, see Boot / CustomLogo
- SubEntries – (Default: true) Disables subentries by default if set to false. Can also be used to create custom nested records using the same structure as the main record, any settings that are not set in the nested record will be inherited from the main record.
- KernelAndKextPatches – Uses the same hierarchy and offers the same functionality as KernelAndKextPatches, so you can selectively apply patches on a per-entry basis. Only works with version 2797 or higher
- Ignore – The entry will be ignored or unused, so it won’t affect anything

Example

How to boot ArchLinux. See https://wiki.archlinux.org/title/Clover

The follow custom entry

```
<key>Entries</key>
```

```
                <array>
                    <dict>
                        <key>Arguments</key>
                        <string>initrd=\initramfs-linux.img root=/dev/sda6 rw
    add_efi_memmap</string>
                        <key>Path</key>
                        <string>\vmlinuz-linux</string>
                        <key>Title</key>
                        <string>Arch Linux</string>
                        <key>Type</key>
                        <string>Linux</string>
                        <key>Volume</key>
                        <string>EFI</string>
                        <key>VolumeType</key>
                        <string>Internal</string>
                    </dict>
```

<key>ShowOptimus</key>

```
<true/>
```

The task was simple. My BIOS spontaneously turns off Optimus on my laptop, and I want to see during Clover loading whether it is on or not, so that I can press a key in time and fix the situation. The criterion is the number of video cards in the system, or the first Intel..
Intel + Discrete = Optimus. On the screen the word Intel
Discrete only. On the screen the word Discrete. If anyone needs it, I do.

Design: theme.plist

Now the actual design in accordance with the chosen theme. The theme.plist file is loaded from the theme folder and is unique for each of them. The path for the metal theme is:
`/EFI/CLOVER/themes/metal/theme.plist`
The first parameters of the theme are copyright, like this

```
<key>Author</key>
<string>Slice</string>
<key>Year</key>
<string>2012</string>
<key>Description</key>
<string>Main metallic looking theme</string>
```

Next comes the section with design parameters

```
<key>Theme</key>
<dict>
```

The format of all the mentioned images is PNG, and they must have the correct title. For example, the program "Preview" saves files in the correct format, but not always. Sometimes you need to re-save via Photoshop (it seems that this is no longer relevant, the new PNG reading algorithm in Clover copes with all images). Some interface elements can be excluded by the following set:

<key>Components</key>

```
<dict>
    <key>Banner</key>
    <true/>
    <key>Functions</key>
    <true/>
    <key>Label</key>
    <true/>
    <key>Tools</key>
    <true/>
    <key>Revision</key>
    <true/>
    <key>MenuTitle</key>
    <true/>
    <key>MenuTitleImage</key>
    <true/>
```

```
    <key>Help</key>
    <false/>
</dict>
```

If <true> then the element is present, otherwise it is not.

Theme Style (starting with 3586 by Needy)

```
<key>BootCampStyle</key>

<true/>
```

By default, the text in the refit is written on a separate line, and can be very long.
"Boot Recovery from RecoveryHDD". But in BootCamp, as in Chameleon, it is customary to write inscriptions directly under the icons, but only briefly: MacOS, Linux, Windows, because otherwise it will not fit. But this is only true if there are few partitions, and each partition has its own, unique bootloader. If, for example, we have two systems, and one of them is closed on FileVault2, we will have the following items
"Boot Recovery from RecoveryHDD"
"Boot macOS from RecoveryHDD"
"Boot macOS from SierraHDD"
The bootcamp style here is somehow inconvenient. So, you have a choice.
Screen background:

```
<key>Background</key>

<dict>
      <key>Type</key>
      <string>Crop</string>
      <key>Path</key>
      <string>MetalBack.png</string>
      <key>Sharp</key>
      <string>0x80</string>
      <key>Dark</key>
      <true/>
</dict>
```

The **Path** parameter specifies the file name (or rather the path!) in which the background image is located on the entire screen. In this case, the screen may be smaller or larger than the image, and what to do with this is determined by the parameter **Type**
Crop - crop a large image to fit the screen, or fill it with background.
Tile - to pave with a mosaic of tiles.
Scale — stretch proportionally so that the image takes up the entire screen and more, for cropping. With regular stretching, square pixels are obtained, so some smoothing is usually applied, however, such smoothing spoils the edges. Clover has edge detection, its value is determined by the parameter **Sharp**
If 0 - no detection, edges are blurred. Maximum value 0xFF = 255 - no blur. 0x80 - creates some intelligent blur with sharp edge lines. Also works in tandem with it parameter **Dark** If
<true/> means you have a dark image with white lines,
<false/> means you have a light image with dark lines. This affects edge detection.

```
<key>Banner</key>

<string>logo-trans.png</string>
```

The banner is the central image, it has size restrictions depending on the screen size. For example, in the dawn theme the image has a size of 672 × 190 pixels. This figure can be considered as the maximum. The logo should either be made opaque if we are not going to use a background image. Then the first pixel of the logo determines the background color. Or the logo has an opaque element

on a transparent background, and the entire screen is covered with a background image. Eps trick: make the upper left pixel opaque by 1%.
In new revisions, "Banner" has parameters:

```
<key>Banner</key>
<dict>
	<key>Path</key>
	<string>logo_trans.png</string>
	<key>ScreenEdgeX</key>
	<string>left</string>
	<key>ScreenEdgeY</key>
	<string>top</string>
	<key>DistanceFromScreenEdgeX%</key>
	<integer>10</integer>
	<key>DistanceFromScreenEdgeY%</key>
	<integer>10</integer>
	<key>NudgeX</key>
	<integer>8</integer>
	<key>NudgeY</key>
	<integer>5</integer>
</dict>
```

Path - path to file, including folder, for example VariantA\Logo.png
ScreenEdgeX - horizontal reference point (left/right/centre)
DistanceFromScreenEdgeX - banner position relative to the reference point, in percentage by screen size. This ensures correct positioning when changing resolution.
NudgeX - 1% is a lot, for a 1920 screen there will already be 19 pixels, so in this parameter we make a clarification in pixel
Similarly vertically.

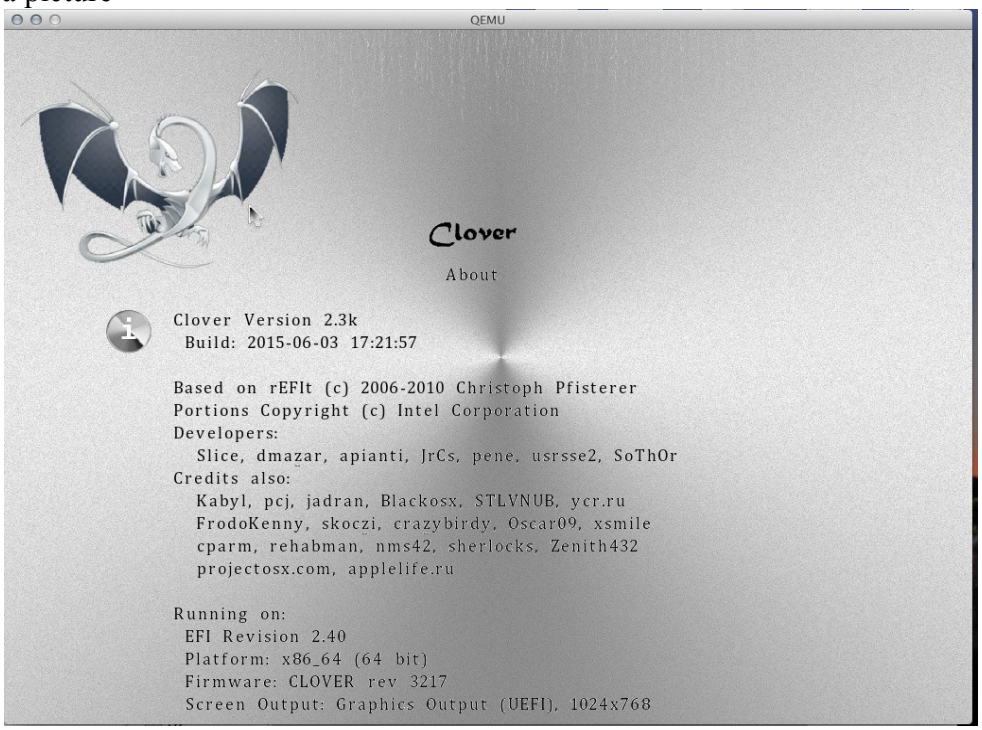
```
<key>Selection</key>

<dict>
	<key>Color</key>
	<string>0xF3F3F380</string>
	<key>Small</key>
	<string>Select_trans_small.png</string>
	<key>Big</key>
	<string>Select_trans_big.png</string>
	<key>OnTop</key>
	<true/>
</dict>
```

Color — the color of the menu line highlight. The artist sets the color according to the general tone of the theme. The value 0x11223380 means the color red=0x11, green=0x22, blue=0x33, alfa=0x80. The last number is the opacity, 0x80 corresponds to 50%. 0x00 will mean no highlight. 0xFF will cover the background image (letters on an opaque bar).
Big and **Small** are the images that highlight the icons in the main menu in the top row - large, and in the lower one - small ones.
OnTop — the location of the selection image (rev 1983). False — the selection is under the disk icon (traditional for Refit), True — above the icon (traditional for Chameleon).

```
<key>Font</key>

<dict>
	<key>Type</key>
	<string>Load</string>
	<key>Path</key>
	<string>BoG_LucidaConsole_10W_NA.png</string>
	<key>CharWidth</key>
	<integer>10</integer>
	<key>Proportional</key>
	<true/>
</dict>
```

Type — font type. There are two built-in fonts Black and White (rev. 3706+), and a dozen

downloadable – Load. In this case, the file name is specified in the following parameter **Path** - BoG_LucidaConsole_10W_NA.png For each theme, its author has selected a font that best suits his idea, you should look in the attached file. The following conventions are accepted for font names (blackosx)
BoG — Black On Gray — black on a gray background.
LucidaConsole is the name of the original font.
10W — the width of the letter
NA — No Antialiasing. Also well thought out. The size of one symbol in the file is 16 pixels, however, the symbols themselves take up less space, so the next parameter specifies the optimal width, and this, again, depends on the author's intention.
Proportional — starting with revision 3217, it became possible to use proportional fonts. For those who don't know, this is when the letter i takes up much less space than m. A monospace font is, for example, Courier, proportional, for example, Times. However, Clover is able to compact monospace fonts, but still, the result will be better with specially prepared ones. Here, for example, is a picture



Compare the width of the letters i and m.
CharWidth 10 — you can use the width recommended by the font author, or you can change it to your liking. 9 — tighter, 11 — more sparse. Starting with revision 3537, this parameter affects the text width in the Options Menu, since it always has Proportional=false, and the width of the letters is entirely determined by this parameter, while in the information line the text is proportional and is actually compressed for each letter, even if the width in the matrix is much larger.

```
<key>Badges</key>

<dict>
	<key>Show</key>
	<true/>
	<key>Inline</key>
	<true/>
	<key>Swap</key>
	<false/>
	<key>OffsetX</key>
	<integer>32</integer>
	<key>OffsetY</key>
	<integer>32</integer>
```

```
        <key>Scale</key>
        <integer>7</integer>
</dict>
```

The badge is a small image in the lower right corner of the main image. The original idea was for the main icon to represent a disk (like in bootcamp), and the badge to indicate what operating system is running.

Show — whether to show the badge.

Swap — change the meaning of the icon and badge. Now the icon depicts the OS, and the badge — the device (in this case, it is not interesting to show it).

Inline — show the badge in the line with information about the selected icon. It is always OS here, regardless of the Swap parameter. See the screenshot of the iClover theme.

OffsetX и **OffsetY** - badge offset from the upper left corner. If no offsets are specified, the badge is located in the lower right corner.

Scale - badge size in units of X/16 of the original size (in the example 7/16)

That is, in the standard theme the size is 48 pixels, which corresponds to 6/16 of the standard icons.

```
<key>Scroll</key>

<dict>
        <key>Width</key>
        <integer>N</integer>
        <key>Height</key>
        <integer>N</integer>
        <key>BarHeight</key>
        <integer>N</integer>
        <key>ScrollHeight</key>
        <integer>N</integer>
</dict>
```

Since the settings menu may be longer than the vertical size of the screen, a scroll bar appears in the menu, its parameters are set by the theme, and there are default parameters for images included in the theme.

```
<key>Anime</key>

<array>
        <dict>
                <key>ID</key>
                <integer>1</integer>
                <key>Path</key>
                <string>logo_3D</string>
                <key>Frames</key>
                <integer>15</integer>
                <key>FrameTime</key>
                <integer>200</integer>
                <key>Once</key>
                <false/>
                <key>ScreenEdgeX</key>
                <string>left</string>
                <key>ScreenEdgeY</key>
                <string>top</string>
                <key>DistanceFromScreenEdgeX%</key>
                <integer>20</integer>
                <key>DistanceFromScreenEdgeY%</key>
                <integer>20</integer>
                <key>NudgeX</key>
                <integer>1</integer>
                <key>NudgeY</key>
                <integer>1</integer>
                <key>RelativeXPos</key>
                <string>50%</string>
                <key>RelativeYPos</key>
                <string>10%</string>
```

```
            </dict>
        </array>
```

The theme may contain animated images (clips). A series of PNG images with consecutive numbers is supported.

ID — determines the usage of this clip.

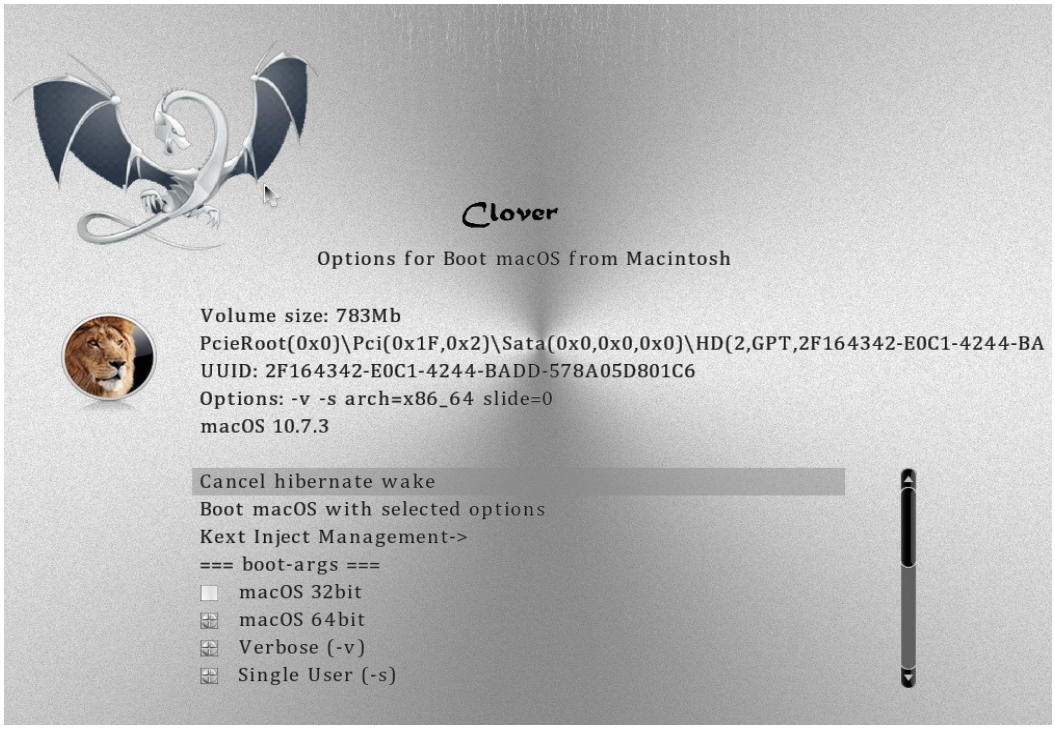
```
#Logo                   (1)
#About                  (2)
#Help                   (3)
#Options                (4)
#Graphics               (5)
#CPU                    (6)
#Binaries               (7)
#DSDT   fixes           (8)
#BOOT Sequence          (9)
#SMBIOS                 (10)
#Drop ACPI Tables       (11)
#RC Scripts             (12)
#USB                    (13)
#Themes                 (14)
#Apple                  (21)
#WinXP                  (22)
#Clover                 (23)
#Linux                  (24)
#LinuxEFI               (25)
#BootX64.efi            (26)
#Windows UEFI           (27)
#Recovery               (30)
```

title images in each submenu are animated, as well as this animation is played on the selected main menu item

---

1-10 — list of existing settings submenus.

21-27, 30-39 — this is the "Boot Options" details menu, called by the space bar on the icon in main menu, or by right-clicking.



That is, on this screen the Lion will be animated if ID 37 is specified.

Path - ML_Anim — The name of the animation, defines the name of the folder in which the individual animations are located
frames with names
ML_Anim_000.png
ML_Anim_001.png
ML_Anim_008.png
ML_Anim_014.png
In case of missing frames, the last valid one will be used, i.e. as Frame 001 will be used for frames 002-007, and frame 008 will be used for frames 009-013. This is convenient if the picture does not change during this period of time according to the plot.
Frames - 15 — the total number of frames in the animation. The missing ones will be filled in according to the above algorithm.
FrameTime - 100 — time interval between frames in ms. Variable interval is implemented using dropped frames.
Once — if <true/> is specified, the animation will be played only once, until exiting the main menu (right-click on the milk on the main screen, or the Escape key). If <false/> is specified, the animation is played in an infinite loop, after the last frame there is a zero frame after the same interval, without an additional pause.
ScreenEdgeX - horizontal reference point (left/right/centre)
DistanceFromScreenEdgeX - the position of the film relative to the reference point, in percentage of the screen size. This
ensures correct positioning when changing the resolution..
NudgeX - 1% is a lot, for a 1920 screen it will already be 19 pixels, so in this parameter we make a clarification in pixel units.
In the latest revisions, we began to experiment with changing the location itself theme elements:

```
<key>Origination</key>
<dict>
   <key>DesignWidth</key>
   <integer>1920</integer>
   <key>DesignHeight</key>
   <integer>1080</integer>
```

With these parameters we indicate what screen resolution the theme is initially designed for, so that at a different resolution the layout of the elements can be correctly recalculated.
A large section about the layout itself.

```
<key>Layout</key>
<dict>
   <key>Vertical</key>
  <true/>
```

<key>BannerOffset</key>

```
<integer>80</integer>
```

This is the distance from the banner to the main menu, sometimes it is necessary to replace it so that the banner animation does not overlap the main menu icons. Similarly

<key>ButtonOffset</key>

```
<integer>20</integer>
```

<key>TextOffset</key>

```
<integer>30</integer>
```

<key>AnimAdjustForMenuX</key>

```
<integer>30</integer>
```

You can also scale the main menu icons

<key>MainEntriesSize</key>

```
<integer>200</integer>
```

The default value is 128, as it was before. With
changing icons, you can also change the distances between them

<key>TileXSpace</key>

```
<integer>20</integer>
```

<key>TileYSpace</key>

```
<integer>20</integer>
```

You can also change the selection size.

<key>SelectionBigWidth</key>

```
<integer>288</integer>
```

Default value is 144. This matters if the selection is in the background.

Vector Themes

Why is it needed?

The decision to create the design of the Clover GUI based on vector graphics arose from one user complaint that they say you start up from the same flash drive on different computers, one has a small screen, and then a large theme simply does not fit, and the other has a huge screen, and then a regular theme looks too small, and the text is completely unreadable. How to combine this? The option was created quickly, a theme with the @2x prefix. That is, in the Clover config, the metal theme is specified, and depending on the monitor resolution, either metal or metal@2x is loaded. The disadvantage of this approach is that no one rushed to create double themes. And in general, it did not turn out very nice.

Then I decided to make truly scalable themes based on vector graphics. For those who don't understand what's going on, an example: a circle can be represented as PNG file, which has a very specific size in pixels, or you can write an instruction <circle cx="100" cy="200" r="50"> and instruct the drawing program to make a circle on the screen with such a position and such a radius in abstract units, which will be recalculated into the required number of pixels depending on the screen size. In the original, this could be a plotter working in millimeters, for example.

You can't make a good design with circles and squares alone, so you need some vector graphics standard with more options. For an open project like Clover, you obviously need an open standard, well-documented, and supported by drawing programs. The choice is clear - SVG graphics.

`https://en.wikipedia.org/wiki/Scalable_Vector_Graphics`

Now the next question is, how can I make it support? This is necessary interpret these commands and draw something corresponding, to put it mildly, is not an easy task. And then I find the nanosvg project,

`https://github.com/memononen/nanosvg`

It is open source, allows you to copy it to yourself, it contains only two files in the C language, that is, you can use it almost without changes. The author stopped the development, despite numerous suggestions for improvement, like it is done like this, use it as is. But this is not enough for me, I need more. In addition, almost without changes will not work, in Clover it is elementary impossible to use the standard C functions sin(), malloc(), sscanf(), qsort() and so on, I still had to make their implementation within the capabilities of EDK2. I did this, and also looked at the pull requests, with very good suggestions for improvement, and also made my additions. I compared what from the SVG1.0 standard was supported in the original project, and what is supported by me. It's night and day! Half a year of work says something. So, I can rasterize vector images and display them on the screen. Now I need to use them to create a scalable theme. Preferably without ruining support for existing PNG themes.

But theory without practice is dead, I can't do theme support if I don't have a sample of it. The theory is as follows, at the first stage. Let the whole theme be a single file named theme.svg, which any viewer, for example GoogleChrome, Safari or just by pressing the space QuickLook, will look like the Clover interface. And Clover will figure out how to parse this file into icons, and output them as it outputs raster theme. Thanks to Clovy, he accepted the challenge and started drawing such a file, and as we worked we found together what needed to be done in Clover and how such a file should be formed so that the ends meet. About this on the forum .Как сделать векторную тему в Adobe Illustrator

Scaling the theme means that the interface will look the same on any screen size, the icons will occupy the same percentage of the screen size, in the sense of the screen height. The theme does not scale in width, in favor of preserving squares. That is, on screens with different aspect ratios, the picture will be different, strictly speaking, but a circle remains a circle, and a square a square.

Vector themes also have the advantage of using fonts. In raster the theme was only one font, somehow made proportional or not, while in vector it can be several different fonts, truly proportional, and very smooth. In the future, even greater improvement is possible, for example with animation, in general, prospects are unfolding. Although yes, I understand, for those who set Timeout=0, or TextOnly, or embedded, this chapter not interesting at all.

How to make a vector theme

First, you need to install a drawing program that you can comfortably draw in and that can export this drawing to SVG version 1.1. Although with some improvements compared to this standard. There are options, you can Google what is offered on the market. My brief overview
1. **Adobe Illustrator.** The higher the version, the better. This is what I was looking for compatibility with first. It is convenient and functional. But be careful! Some AI features cannot be exported as SVG, it turns out raster.
2. **Inkscape**. Advantage #1 is that it is free, and therefore legal. SVG is its native format, but it has a number of its own SVG extensions that are not supported by anyone else. Hooray! Version 1.0 now works in MacOS 10.11 and higher!
3. **LibreOffice Draw.** Also free and legal. But it's a bit clunky and the resulting SVGs aren't very good. Doesn't support embedded SVG fonts.
4. **CorelDraw**. I just don’t know, I haven’t used it.
5. **BoxySVG**. A nice, simple, cheap program that allows you to create and edit individual SVG drawings and immediately control the code being created, unlike its older brothers, which will only generate the code upon final export. The program's drawback is the lack of tools for solving the problem (I want to see this, how to do this?). In Chandelier it's clear, you draw and see what happens as you draw, as you edit, but here you need to have a good imagination to guess the result.
6. **UltraEdit**, Awesome text editor with SVG preview. Unfortunately, it's paid, but I edited the entire theme with this editor, and I immediately saw the effect of changing one digit. I won't consider other programs, at your discretion, just follow the standard version 1.1, and the recommendations below. Secondly, the theme.svg image should consist of very specific components with very specific names, since Clover selects images from there by name (id).
Full list::
Background, Banner, selection_big, selection_small, selection_indicator, pointer, scrollbar_background, scrollbar_holder, checkbox, checkbox_checked, radio_button, radio_button_selected,
For some reason, scrollbar was defined with a minus instead of an underscore, because that's how it is in raster themes, but it's fixed in vector ones.
Disk icons
vol_internal - internal hard drive;
vol_external - external drive (usually USB);
vol_optical - CD/DVD drive;
vol_clover - bootloader disk (not used?)
vol_internal_hfs - disk partition with the HFS+ file system;
vol_internal_apfs - disk partition with the APFS file system;
vol_internal_ntfs - disk partition with the NTFS file system;
vol_recovery - disk partition Recovery,

RecoveryHD and others Operating system icons
os_clover
os_legacy
os_unknown
os_mac — for any other Mac OS o
os_tiger Mac OS X 10.4 Tiger
os_leo Mac OS X 10.5 Leopard
os_snow Mac OS X 10.6 Snow Leopard
os_lion Mac OS X 10.7 Lion
os_cougar OS X 10.8 Mountain Lion
os_mav OS X 10.9 Mavericks
os_yos OS X 10.10 Yosemite
os_cap OS X 10.11 El Capitan
os_sierra macOS 10.12 Sierra
os_hsierra macOS 10.13 High Sierra
os_moja macOS 10.14 Mojave
os_cata macOS 10.15 Catalina
os_bigsur macOS 10.16/11.x Big Sur
os_monterey macOS 12.x Monterey
os_ventura macOS 13.x Ventura
os_sonoma macOS 14.x Sonoma
os_sequoia macOS 15.x Sequoia

os_win Mandatory, used when there is no icon for the current Windows OS (on a volume with the NTFS file system)
os_vista Windows Vista, Windows 7, Windows 8, Windows 10
os_freedos
os_freebsd

os_linux Mandatory, used when there is no icon for the current Linux OS (on a volume with an EXT file system)
os_ubuntu
os_suse
other Linuxes may someday be added to the list.
Second-row service icons
tool_shell, func_clover, func_options, func_about, func_reset и **func_shutdown**, as well as **func_help,** which is displayed when calling help with F1.

Third, the placement of these icons inside the work area in the theme.svg file plays a role for the preview, but not for Clover itself. Except for the Banner. This is the central image, and it is not simply located in a fixed place in the center of the screen, as it was in raster themes, the banner is now located on the screen as intended in the theme. An example can be seen in the cesium theme. Neither the flower nor the dragon are centered. Another point, all sorts of buds and scrolls are generally not needed on the preview, as well as numerous icons of axes and disks, so they should be hidden from the preview, but left accessible to Clover. The attribute visibility=”hidden” helps, which, however, Clover will ignore when it needs to draw this icon on the screen.

Vector themes support a new feature called “night design”. The built-in clock will tell Clover what time it is, recalculate it to local time, вand the time from 8:00 to 20:00 is considered daytime, and the rest is night. In vector themes, all icons can be duplicated with the addition of _night to the name. For example, in addition to the main banner named “Banner”, you can draw a

second picture named “Banner_night”. And so on with all the icons, for example “pointer” – “pointer_night”, “os_mac” – “os_mac_night”. They can differ simply in illumination, or they can be completely different. So we actually have two different themes in one, one is displayed during the day, the second at night. Each icon, in addition, must contain an invisible rectangle, defining its boundaries. The reason is simple, the icon may contain less content than its size, and Clover needs to know its full size, which needs to be scaled. The name of such a rectangle necessarily contains "BoundingRect_", but since names must be unique, some other symbols, just numbers, or the full name of the icons.

Example.

```
 <g id="tool_shell" transform="translate(300, 600)">
<rect visibility="hidden" id="BoundingRect_ts" y="0" width="64" height="64"/>
<g transform="matrix(0.55 0 0 0.55 15 10)">
<use xlink:href="#knopka"  width="85.5" height="85.5" />
</g>
<g transform="matrix(0.55 0 0 0.55 25 43)">
<text class="st35 st36 st38 st39">$_</text>
</g>
</g>
```

Let's decipher it word by word. <g ...> is a group of images that make up one summary. Its name is "tool_shell", which Clover uses to draw the Shell icon in the second row. transform="translate(300, 600)" - sets the icon offset on the preview. Clover ignores it. The second line <rect..> is the same invisible utility rectangle that sets the icon size, in our case 64x64. The unit of measurement is pixels, compared to the size of the theme itself, which is 1600x900. Accordingly, on a screen with a height of 1800, the icon will be 128x128, for example. The third line again combines the images, and does a general transformation: the size relative to the above boundaries, and the offset within them. The thing is that in a graphics editor it is convenient for you to draw in a different scale, but then you need to bring the scale of individual icons to the scale of the entire theme.

`<use xlink:href="#knopka" width="85.5" height="85.5" />` -

this was not in the original nanosvg project, and in general it is probably from SVG2.0 - support for symbols. The point is not to draw the same object several times, but simply use the same drawing. All my second-row buttons use the same image named "knopka". And the next image is superimposed on it, already with its own transformation, and as its text, consisting of two letters, for example "$_". The classes of this text determine the font, size, fill color, and stroke color. The classes are defined in the theme.svg file itself, somewhere at the beginning. There is a problem with transformation in Lustre, probably like in other vector editors. It tends to recalculate the coordinates of all internal objects of the image, instead of leaving the transformation attribute outside the group. I did not learn to deal with this, so I drew all the objects at the origin of the coordinates, and then wrote the scale and offset simply in a text editor. Maybe someone will have more luck with this exercise. Why is this important? Because either in Clover or in the preview, the parts begin to creep in different directions, and catching them with Illustrator tools spoils the whole picture even more.

SVG standard support in Clover

Far from perfect. But we can see that a lot of things Illustrator does work as expected.
Shapes: <rect>, <circle>, <ellipse>, <polyline>, <polygon> and just <path>
Shapes can be combined into groups <g> and into symbols <symbol>. Each group can contain a name and a transformation. A group is a static group, both defined and displayed. Group names are important for Clover, because it divides the image into icons, which it then uses to draw the GUI. So <g id="vol_internal"> defines a group of shapes that together create the image of the internal disk. Inside this group, the required element is

`<rect id="BoundingRect_001" width="128" height="128"/>`.

It is needed to define the size of the group, so that Clover can position the image correctly, because the internal drawings will most likely be smaller than the overall size. Each element's name is unique within a single design, so the name of such a rectangle must contain a BoundingRect, and any additional symbols for uniqueness. This does not contradict the SVG standard, but is necessary for Clover to manipulate individual elements. The symbol is a dynamic union. Browsers will not display symbols defined in the theme, this set of objects is a template for use by other objects. It will be used by the <use> operator.
Example:

```
<symbol id="HDIconBase" viewBox="-64 -64 128 128">
    <rect id="BoundingRect_3_" x="-64" y="-64" class="st0" width="128"
height="128"/>
    <path id="bottom_3_" class="st1" d="M26,49h-52c-3.9,0-7-3.1-7-7v-66c0-
3.9,3.1-7,7-7h52c3.9,0,7,3.1,7,7v66C33,45.9,29.9,49,26,49
        z"/>
    <path id="top_3_" class="st2" d="M26,47h-52c-2.8,0-5-2.2-5-5v-55c0-2.8,2.2-
5,5-5h52c2.8,0,5,2.2,5,5v55C31,44.8,28.8,47,26,47z"
        />
</symbol>
```

Here is defined a symbol consisting of two closed curves (path) and one utility square.
Styles/classes are defined in the file, here for brevity I omit them. We draw this figure inside the icon

```
<g id="osx_moj">
    <rect id="BoundingRect-2" class="st0" width="128" height="128"/>

    <use xlink:href="#HDIconBase"  width="128" height="128" x="-64" y="-64"
transform="matrix(1 0 0 -1 64.0876 64)" style="overflow:visible;"/>
```

xlink:href is a link, and points to a symbol called HDIconBase. Unlike the SVG standard, Clover does not understand links to groups, only to symbols. Alas and ah, but you need to take this into account when creating your design. That is, transfer the group to the symbol, and make a link to this symbol.
Fills (**fill**) and strokes (**stroke**). I can't say anything specific about strokes. It was made by the founder of the nanosvg project Mikko Memononen, it seems to fully comply with the standard, and even supports dash-dotted lines (dash-line). But let's talk about fills. A simple option is a uniform fill with color. The color is specified by 16-digit RGB constants: fill: "# ffcc00" is orange.
There are also real numbers and percentages. Above the standard, Clover has color names: fill: "white" and many more names, I will not copy-paste a hundred words. Use the SVG standard, and if anything, look in the Clover source.
Another type of fill is gradients.
linearGradient - here 100% compliance with the standard.
radialGradient - unfortunately, the focus point is not supported yet, so the radial gradient in Clover may not look quite as intended in the design.
conicGradient - this is not in the standard (yet?). There is a discussion on the Internet, very interesting drawings made with this method are posted, but the W3C consortium is not yet considering including this in the standard, and accordingly, browsers do not support it. And how can I not support it if my Metal theme is created on conical gradients?! And now the cesium theme!
Another question, how did I draw it then? In Photoshop, it can do it. Adobe did not include this function in Illustrator, probably waiting for standardization. Syntactically the same as radialGradient, only the color changes not by radius, but by angle, considering angle = 0 is stop = "0.0", and angle 360 is stop = "1.0". Well, as many intermediate values as you want. If you need to rotate the initial angle, there is gradientTransform="rotate(45)" and other transformations, as in the standard. For design, you can make a radial gradient, and then use a text editor to turn it into a conical one. The preview, unfortunately, is only in Clover. If anyone finds a second browser that supports cones, let me know.

Dithering. This is also not in the standard, although, in my opinion, it suggests itself. A brief theory, for those who have never heard. If a gradient has a row of points with a brightness of 150, and after them with a brightness of 151, then a step will be visible, but there are no intermediate values (noticeable on the Clovy theme background with circles on early versions of Clover, without dithering support). The method is to output either 150 or 151 randomly instead of the required point with a brightness of 150.3, with a probability of 0.3. With a large number of points, it will sum up to 150.3! The standard does not have a syntax sample for this case, so we developed our own. And it needs the clover namespace. Code:

```
    <radialGradient clover:ditherCoarse="16"  id="GrayRadialBackground_5_"
cx="441.2867" cy="0.7502" r="1.0023" gradientTransform="matrix(5.400000e-14 768
-874.24 4.700000e-14 1338.8547 -338690.875)" gradientUnits="userSpaceOnUse">
        <stop  offset="0" style="stop-color:#7C7C7C"/>
        <stop  offset="1" style="stop-color:#5E5E5E"/>
    </radialGradient>
```

The digital parameter coarse - roughness, determines the length of viewing points, between what and what to choose. For large drawings, as a background, the value 16 was selected. For icons, 1 is enough, and so it looks very smooth and almost imperceptible. By default, the value = 0, which means canceling the method. I will show the illustrations later, however, they were laid out on the insanelymac.com at different times.

Fill with a pattern Here we are very far from full implementation. In the standard, the image will be repeated as much as necessary to fill the filled figure. In Clover, only one option is made: The pattern is a PNG image, and it fills the desired contour in scale to completely fill it. In this case, the raster image will be scaled with anti-aliasing using the dithering algorithm. Why is it necessary? The fact is that many Apple logos are photographs, that is, a raster, and we have options: vectorize the raster (not very nice), draw something completely different, or include the raster in the vector. I did it this way. Illustrator has such a function

1. Include the PNG image in the design. (embed).
2. Translate the figure into a pattern
3. In the properties of the filled figure, specify to fill with a pattern with such-and-such ID. The only problem is that the image does not multiply, as required by the specification, but fills the entire space with scaling to the dimensions of the canvas. Oh yeah, and another limitation

```
<pattern id="pattern_1234"....>
```

According to Clover's rules, a pattern identifier must include the word “pattern”, otherwise Clover has no way of understanding what we're filling the shape with. However, using embedded PNGs doesn't make much sense, in this case it would have been better to just make a PNG theme.

Clips (clipPath)

Unlike the original nanosvg project, I support clips, but the algorithm is not mine, but taken from a pull request. It does not work quite adequately. Here in the cesium theme, Clover is drawn using them, and it turned out almost correctly. There is some difference in the rendered image compared to the preview, but I do not know how to fix it, support for clips. In general, you can use it, but be careful. One support bug is localized: the definition of the clip must be in the same group as the use, since the transformation does not affect the clip outline.

Texts and fonts

The original nanosvg project lacks this feature, but Clover is unthinkable without text, so I did it. Safari has a normal view of images with fonts embedded in the image file, but it is missing from Chrome and Explorer. They display text in their own fonts, of course with design distortion.

Embedding SVG fonts is available in Illustrator, but is missing, for example, in BoxySVG, which can only embed TTF fonts. I don't even remember about other vector editors, for all of them this is a problem one way or another. Clover's vector theme has its own set of rules.
1. You can use several fonts, while raster themes have only one. One of them, the main one, should be included in the theme.svg file itself. You can include everything, but the file will simply swell in size. The rest can be placed in the theme folder, next to the theme.svg file. However, in this case, we will lose the differences in fonts in the preview. That is, Clover will see all fonts, Safari will see only the built-in one, and Chrome will not see any.
2. At the moment, it is possible to define three text styles (font, size and color): for the Options menu, for the Help and About screens, and for the messages at the bottom of the screen, the same style is the heading. Maybe in the future we will expand the use of styles. This is done with a special trick.

```
<g id="MenuRows" class="st0"><text class="st1">Menu</text></g>
<g id="HelpRows" class="st0"><text class="st2">Help</text></g>
<g id="MessageRow" class="st0"><text class="st3">Boot macOS from HDD</text></g>
```

Here classes st1, st2, st3 should define text styles, and class st0 is needed just to hide unnecessary text from the preview (opacity:0;). As you can see, everything is standard here, it's just a different purpose. Here we write the text only to designate the styles. And Clover itself will find what to write with these styles, by group id, that is, for the menu "MenuRows", and for help by F1 "HelpRows".
3. But in general, you can use text in icons as a decorative element. You can use any font, but make it embedded, and maybe embed this font with the "only used glyph" option - that is, not the entire font, which weighs 500 KB on average, but Chinese and all 30 MB, but only the letters used in the icon. Moreover, you can apply transformations to the text: rotation, tilt, etc. You can make a stroke without a fill, the letters will be "white" - I don't remember what it's called, there is no such style in Windows, but Mac has had it since the old days. Unlike the standard, there is no textPath - the placement of text along a line, for example, along the perimeter of a circle. Maybe someday ... The text size is specified in pixels. That is .st8{font-size:16px;}, but this is in the size of the theme design. In real work, the theme will be scaled to the size of the monitor, along with the size of the letters. Thus, for a theme in a 1366x768 design, text with a size of 12px looks normal, on a monitor with a vertical of 2144 it will be almost 36px.
4. For my design I need a font that contains special Unicode symbols. For example . ÿ I found such a font, it takes up 30 MB, where should I put it? I take only individual symbols from there and implement them into my theme. Illustrator, seeing an unfamiliar font, replaces it with its own, and kills everything I did in my theme.... The first rule is that the font used must be installed in the system, not just in the theme. But I don't have the fonts that are installed in the system, in the form of SVG. So... I name my built-in font ArialMT, Helvetica, etc., insert glyphs from the font I need there, Illustrator shows it incorrectly, because it's a different font, but at least it doesn't kill it. Quicklook shows it correctly, and I can manually move the letters to where they should be. I also have inscriptions directly on the icons. I did this simply in Illustrator, with its font, and specified to export the font "used only glyphs". At the same time, when updating the theme, I risk not getting some letter that I haven't used before. So, we should implement text in the original theme design that contains all English letters, numbers and punctuation marks. Something like this

```
<text class="st1">ABCDEFGHIJKLMNOPQRSTUVWXYZ...
```

and so on. Then Illustrator won't forget about all these symbols when importing the theme. And put this text in an invisible group.
5. After you use Illustrator and ask it to implement the SVG font, only the fonts used, or everything in general, then you should check what it has scribbled there. Here, for example, is the Code:

```
<font horiz-adv-x="2048">
<!-- Helvetica is a registered trademark of Linotype AG -->
<!-- Copyright: Copyright 2018 Adobe System Incorporated. All rights reserved.
-->
<font-face font-family="MyriadPro-Regular" units-per-em="2048" underline-
position="-155" underline-thickness="101"/>
<missing-glyph horiz-adv-x="1298" />
<glyph unicode=" " horiz-adv-x="569"/>
<glyph unicode="E" horiz-adv-x="1366" d="M175,1469l1071,0l0,-180l-877,0l0,-
446l811,0l0,-170l-811,0l0,-498l892,0l0,-175l-1086,0z"/>
<glyph unicode="H" d="M161,1469l201,0l0,-607l764,0l0,607l201,0l0,-
1469l-201,0l0,687l-764,0l0,-687l-201,0z"/>
```

Firstly, the font-face element is missing bbox="", and it is strictly necessary, look for what is there must be written by hand.

Secondly, scroll through the entire list so that each glyph has the attribute horiz-adv-x="***", it determines the width of the letter, that is, where you can draw the next one. There is a default value, but it does not always correspond to the actual width of the letter, as a result, such a letter will run into its neighbor. In this example, such an attribute is missing for the letter H, and its actual value is 1479, which is not obvious. You can take it from a letter of approximately the same width, and you can make it a little more or a little less through a series of experiments.

Theme Attributes

Clover needs its own additions compared to the standard, and this is done in its own namespace. To do this,

```
xmlns:clover="https://sourceforge.net/projects/cloverefiboot"
```

is added to the beginning of the file . This is a legal operation. After that, we can add illegal things, but with the clover: prefix, as we did with Dithering above. Another example is conical gradients, this is a deviation from the standard, so we had to introduce an additional attribute, allowed by the standard, but indicating to Clover that it should act non-standardly.

```
      <radialGradient clover:conic="1" id="knopkaUp" cx="14142.7324"
cy="-40300.8711" r="37.5003" gradientTransform="matrix(-0.706 -0.7082 0.7082
-0.706 38525.8438 -18436.5312)" gradientUnits="userSpaceOnUse">
            <stop  offset="0"   style="stop-color:#FFF8D4"/>
            <stop  offset="0.1" style="stop-color:#EEE0B4"/>
            <stop  offset="0.5" style="stop-color:#161616"/>
            <stop  offset="0.9" style="stop-color:#EEE0B4"/>
            <stop  offset="1"   style="stop-color:#FFF8D4"/>
      </radialGradient>
```

Here is a radial gradient, and this is how it will be understood by other programs. But Clover will interpret it as a conical gradient. And now the general settings of the theme

```
<clover:theme

BootCampStyle="0"
SelectionOnTop="0"
SelectionColor="0x80808080"
NonSelectedGrey="0"
VerticalLayout="0"

BackgroundScale="crop"
BackgroundDark="1"
BackgroundSharp="0x80"

Badges="show"
BadgeOffsetX="0x0"
BadgeOffsetY="0xA"
BadgeScale="0x10"
LayoutBannerOffset="10"
```

```
LayoutButtonOffset="0"
CharWidth="16"

AnimeFrames="39"
FrameTime="2000"

Version="100500"
Year="2018"
Author="Me"
Description="My cool vector theme for Clover"/>
```

All these attributes are copied from the raster theme settings, and serve the same purposes. However, not all of those attributes are needed and used.

Conclusion

Vector themes are more promising, and I have no desire to deal with raster ones anymore. There is a night mode, and beautiful fonts, and scaling. It is possible to make dynamic pictures like animation, only on a different level. For this, you need to implement deviations from the SVG standard again. But creating a vector theme is much more labor-intensive, and four have managed it so far: Clovy with the Clovy theme, me with the cesium theme, blackosx with the BGM_SVG theme, although he has not yet decided on the final version. We tested the animation on his BGM_SVG theme, but now for some reason he has removed it, probably waiting for vector animation. The fourth designer coped with the task: pkdesign with the "Purple Swirl" theme. Our ranks have grown!

Cesium theme, daytime view..

Night view

Configuring the hardware

Creating a config.plist file

Actually, Clover does the configuration automatically. But the machine is never perfect, so the user has the right to change various parameters via the config.plist file, or simply in the Options menu when working in the graphical interface. Do you think you can configure the parameters better than the machine? Well, try it!

This is an xml file, but for now it is convenient to view it as text file. You can edit this file with a text editor or a specialized program like PlistEditor or Xcode, if available. Now the Clover package includes the Clover.app program, which has a plist editing function no worse than the programs mentioned. Use it! Clover also includes a version of this config with commented settings, so you can see what settings actually exist, but when used in this form they will not be taken into account. Examples:

```
<key>#SuspendOverride</key>
<false/>
```

The hash # means that this key is completely excluded from the config - commented out.

Then Clover uses the default setting. A more modern option is

```
<key>SuspendOverride?</key>
<false/>
```

The question mark at the end means "ask Clover for the best value". It actually works the same way as the hash comment. The service symbol is moved to the end to be able to sort the entire file and compare it with the second sorted one. The problem statement is as follows: we want to compare two configs. How can this be done? All programs that compare two text files will not be able to compare them, since the same keys can be located in the config.plist file in different places. There is such a sorting method plutil -convert xml1 configImacPro.plist Wow! We got the same config-plist, but with sorted keys. We sort the second config in the same way and now we can compare it with any program like Meld or FileMerge, or diff in the command line. However, there is a problem with commented lines, since sorting occurs by the first symbol. This is why the question mark was introduced, which is at the end of the word and does not interfere with correct sorting!

General rule if you don't know what value to give to some parameter, exclude this parameter from the file altogether, or put # or ?. Don't leave the parameter without a value! And especially don't put a value you don't understand!

The following option for making such a configuration for your computer is offered:
- install the default sample file, it contains only safe parameters;
- boot into the Clover graphical shell and go to the Options menu (there is such a button in the bottom row, or simply by pressing the "O" key from the word Option);
- use the up/down/enter/escape keys to walk through the entire menu and try to understand, what they write there, and why;
- what is clear we correct, what is unclear we leave as is.
- we boot into the system.
If it does not work, we repeat the operation, but having changed parameters, until complete success.

Starting with revisions 5100 we have the Clover.app program, which has the same genconfig function, and this utility is now excluded. A little more manual work for complete perfectionism. Below is a description of the config parameters. All parameters are combined into groups: ACPI, Boot, CPU, Devices, DisableDrivers, GUI, Graphics, KernelAndKextPatches, RtVariables, SystemParameters, SMBIOS***, BootGraphics, Quirks.

Boot

<key>Timeout</key>

```
<integer>5</integer>
```

The bootloader entered the graphical interface and paused for 5 seconds before starting the system by default. If the user presses any key during this time, the time countdown will stop. Options: if 0 sec then the GUI is not called, the system starts immediately, however, if you press the space bar before that then we will go to the GUI. -1 (minus one) then the bootloader enters the menu, does not attempt to start. A pause of 25 seconds in the default config is made so that the user can admire the animation. The countdown and autostart occurs only if the correct DefaultVolume is set, and not the one in the default config. It happens that due to heavy animation the timer ticks slower than 1 Hz, do not worry, it depends on the theme. The timeout does not work if the default system is not defined in NVRAM. Go to the system, to the system panel "Boot Disk", and reboot using it. Next time, the timeout will work! The option with Timeout=0 can be replaced with the option

<key>Fast</key>

```
<true/>
```

In this case, additional loading time is saved by not loading the interface and its elements. That is, there is no chance to enter the GUI. And no chance to fix something in case of an error. Saving a whole second will probably make you rich?! The system will immediately start loading from the section specified in the next parameter

```
<key>DefaultVolume</key>
<string>MacHDD</string>
```

the name of the section, as you named it, as it is displayed in the bootloader log. However, the name can also be set in NVRAM after rebooting from the control panel "Startup Disk". The name set in NVRAM is a priority. There is an option "LastBootedVolume". That is, we will boot from the volume from which we booted last time. If the parameter is not set at all, then only from the control panel.

You can also define a default bootloader

```
<key>DefaultLoader</key>
<string>bootmgfw.efi</string>
```

That is, if there are several bootloaders on one partition, then we will select the required one for booting by default in this way. In this example, we assume to load UEFI Windows by default. If not specified, then boot.efi. Again, "Boot Disk" will change this setting during reboot, thus ensuring automatic Windows startup via the Mac panel. Unfortunately, there is no such service in Windows, you can only return to Mac manually.

```
<key>Legacy</key>
<string>PBR</string>
```

Legacy Boot, which is required to run older versions of Windows and Linux, is very dependent from the hardware, from the construction of the BIOS, therefore several algorithms have been developed, and the choice of algorithm is made in this key. Options:
LegacyBiosDefault – for those UEFI BIOS that have the LegacyBios protocol.
PBRtest, PBR, PBRsata – variants of the PBR boot algorithm, depending on which one you're lucky with.
In general, it was not possible to achieve unconditional work of legacy-boot. Simpler and better already forget about legacy systems and install UEFI system variants. The oldest of them is Windows 7-64, and I personally see no reason to stick with WindowsXP. Does anyone still have a 32-bit only processor? Well, good luck then!

```
<key>LegacyBiosDefaultEntry</key>
<integer>0</integer>
```

Additionally, for UEFI boot, you can specify which hard drive to boot from (not only from the first).

```
<key>Arguments</key>
<string>-v arch=i386</string>
```

These are the arguments that are passed to boot.efi, which in turn passes some of them to the kernel. systems. The specific list of kernel arguments should be looked up in Apple's documentation.
The arguments required by boot.efi itself can be found in the manual for com.apple.Boot.plist. The most well-known are the following

```
Kernel=mach_kernel.amd
slide=0
darkwake=0
```

```
nvda_drv=1
```
For UEFI booting into a 10.8 or 10.9 system, the slide=0 key is required. Starting with revision 1887, it is added automatically when needed. Starting with revision 4369, there is a driver AptioMemoryFix, and with it you don't need to write slide, it means automatic calculation. Starting with revision 3712 (actually later due to bugs), in the Details menu, called by space, you have the opportunity to select the desired arguments from the list with the mouse:

```
  L"arch=i386",       //0
  L"arch=x86_64",     //1
  L"-v ",             //2
  L"-s ",             //3
  L"-x ",             //4
  L"nv_disable=1",    //5
  L"slide=0",         //6
  L"darkwake=0",      //7
  L"-xcpm",           //8
  L"-gux_no_idle",    //9
  L"-gux_nosleep",    //10
  L"-gux_nomsi",      //11
  L"-gux_defer_usb2", //12
  L"keepsyms=1",      //13
  L"debug=0x100",     //14
  L"kextlog=0xffff",  //15
  L"-alcoff",         //16
  L"-shikioff",       //17
  L"nvda_drv=1"       //18
```

P.S. In Clover 4200 and higher, the number of arguments is reduced to purely nuclear ones, otherwise the menu too long, and many of these arguments are not particularly useful to anyone.

<key>Debug</key>

```
<false/>
```

Previously, this key was called Log, which caused confusion about why and how. Setting the value to <true/> will seriously slow down the work, but it gives the opportunity to find out what the problem was after rebooting, because each step will be accompanied by writing the debug.log file to disk. And if you started from a flash drive, then to it. But with a flash drive it will work even slower. The real figure is 10 minutes just to enter the GUI. But if everything hangs for you, you can press Reset, and then look for the file /EFI/CLOVER/misc/debug.log, in which all logs for all boot processes are cumulatively recorded while this parameter is set. Since the revision 5150, a new file is created each time, with a name containing the date and time, example: 2022-06-26_17-47_CLOVERX64.EFI.LOG and if you booted from BOOTX64.EFI, then accordingly 2022-06-26_17-47_BOOTX64 .EFI.LOG Starting with revision 3063, you still won't sit at a black screen — you will see the boot process on the screen! But if you don't like unnecessary messages on the screen, you can insert

<key>NoEarlyProgress</key>

```
<true/>
```

Removes text before loading the bootloader interface, such as "Welcome to Clover"

<key>CustomLogo</key>

```
<true/> OR <false/> OR <string>Apple/Alternate/Theme/None/Path</string> OR
<data>PNG/BMP/ICNS base64 data</data>
```

true — default style
false — disable Apple
logo — apple gray on gray
Alternate — alternative apple white on black
Theme — set by theme

None — no logo, but background is present
Path — path to logo file
<data> - image is encoded as base64 and contains PNG data.

```
<key>XMPDetection</key>
<string>-1</string>
```

The parameter specifies whether XMP should be detected during boot. It depends on the BIOS, and mainly affects the correct detection of the installed memory. In addition, the numerical values 1 or 2 are possible which XMP profile should be used. Perhaps in the future this profile will be used for other purposes.

```
<key>Secure</key>
<true/>
```

“Secure Boot". This invention by Microsoft caused a heated response in the computer world, saying that only Windows 8 would work on new computers, and in the world of Hackintosh they cried "the end of hacking!" But it turned out not to be so sad. Of course, BIOS manufacturers provided for disabling this function. And they also provided for loading certificates. For example, in my case, such BIOS settings do not affect the success of booting.

Apianti decided to do a little more. Let's sign Clover using some "Signing Tool" utility, load the certificate, and let the BIOS work in SecureBoot mode. I don't understand anything about this, so I'll just list what's already been done in Clover, without comment. I hope comments will be added in

the future.

```
<key>Policy</key>

<string>Deny/Allow/Query/Insert/WhiteList/BlackList./User</string>
```

Deny - download only signed files.
Allow - load any
Query - ask the owner
Insert - insert the signature into the database
WhiteList - allow by list
BlackList - exclude by list
User - first check the lists, and then ask the user. The syntax is as follows

```
<key>WhiteList</key>

<array>
    <string>SOMEPATH.efi</string>
     </array>

<key>BlackList</key>

 <array>
    <string>USB(0x1)/HD(0x0,0x1038833...)\EFI\BOOT\BOOTX64.efi</string>
    </array>
```

You may also need a hibernate ignore key, for the simple reason that the image is good, but the technology itself does not work on this computer.

```
<key>NeverHibernate</key>

<false/>
```

Or maybe we are very happy about Hibernate, and don't want to hang around for five seconds waiting for it to happen. Then we write

```
<key>SkipHibernateTimeout</key>

<true/>
```

The second way of hibernation:

```
<key>StrictHibernate</key>

 <true/>
```

This only works if there is a hardware NVRAM, but it is compatible with FileVault2 technology, where the old method does not work. There is news. Lilu+HibernateFixup kexts allow you to save nvram.plist when switching to Hibernate, that is, they partially emulate the work of hardware NVRAM, and thus help to use StrictHibernate with mode 25 on computers with emulated NVRAM.

```
<key>RtcHibernateAware</key>

<true/>
```

Key for safe operation with RTC during hibernation. Author vit9696, questions to him. He claimed that for 10.13.4 systems only this way, but in my 10.13.6 the key is still saved in NVRAM. Nevertheless, it is necessary to set <true> for other reasons, not related to RTC, but with a different wake-up algorithm in 10.13.6.

<key>HibernationFixup</key>

```
<true/>
```

Author: lvs1974, explained in his thread how it works and when. Something like the above situation when there is no hardware NVRAM.

<key>SignatureFixup</key>

```
<true/>
```

When going into hibernation, the system leaves a signature in the image, which is then checked by boot.efi. We wanted to fix it with this key. Probably in vain. It is more correct to leave the default zero, and it works. In my opinion, this key is not needed.

And sometimes the system tries to fall into recovery mode, but it boots from the wrong disk, because we have many systems, this is Hackintosh. To get out of the enchanted ring, we write in the config

<key>NeverDoRecovery</key>

```
<true/>
```

With FileVault2 technology it became possible to use hotkey, but to do this you need to undo the assignments already made in Clover.

<key>DisableCloverHotkeys</key>

```
<false/>
```

Disables all hotkeys in the bootloader menu. A list of all hotkeys can be found by pressing F1 in the bootloader menu.

BootGraphics

<key>DefaultBackgroundColor</key>

```
<string>0xBFBFBF</string>
```

The background color when loading Mac OS X, when an apple with a loading bar appears. Specified in HEX format. This number is the standard Apple gray background.

<key>EFILoginHiDPI</key>

```
<integer>1</integer>
```

Enables HiDPI for Apple with loading bar when loading Mac OS X..

<key>UIScale</key>

```
<integer>1</integer>
```

Scaling the size of the apple with the loading bar, when loading Mac OS X. For example, set it to 2, and you will see an apple double the size.

SystemParameters

<key>CustomUUID</key>

```
<string>511CE200-1000-4000-9999-010203040506</string>
```

Unique identification number of your computer. If you do not set this key, some hardware information will be generated, but if you want full control over what is happening, write your 16-

digit digits. But, for God's sake, do not copy my sample digits! They are not unique anymore, there are plenty of fools who copied them!

```
<key>InjectSystemID</key>
<false/>
```

The same number will be injected in a different way, and in the system properties it will be transformed into something else. The point of this operation is to exactly repeat the UUID generated by Chameleon. To do this, set <true/>, and as CustomUUID use the value that is present with Chameleon in the registry
IODeviceTree:/efi/platform=>system-id.
Then in the profiler we will see a different value, but the same as before with Chameleon. New users no longer know what "as before..." is. They already have Clover. Almost always it is enough to set <true/> in this parameter and no custom. However, if a non-unique number is generated, it will be difficult to understand where the problem is coming from. So it is better to set false.

```
<key>BacklightLevel</key>
<string>0x0101</string>
```

property is injected into the system, and the system knows about its existence. However, the effect is noticeable only on very rare configurations. What is it? Monitor brightness... as the name suggests.
This property is also read from NVRAM, and by default, the value set by the system is used. The value specified in the config or set in the menu will override the default value.
Starting with revision 1865, additional Clover keys were introduced:

```
<key>InjectKexts</key>
<string>true</string>
```

These keys are analyzed by the FSInject.efi driver, if present although kexts are generally loaded from Clover folders without it. But dependencies may be lost. Since revision 5125, kexts are loaded by OpenCor, so FSInject is not needed, and Detect is not needed.

```
<key>NoCaches</key>
<true/>
```

The parameter worked in systems up to 10.7. This is loading only kexts, without caches. In general, it is a strange desire to write this parameter in the config. Probably for someone it did not load differently. But today the parameter is useless, the system in any case loads from the cache.

```
<key>NvidiaWeb</key>
<true/>
```

Sets a flag for loading the Nvidia Web driver, the default value, which you can change in the Clover interface in the Details menu called by the space bar on the system icon. The old method of writing in boot-arg worked only up to Captain. In Sierra and High Sierra, that method does not work, so it is moved to a separate config item.

SMBIOS

This group of parameters is needed to mimic your PC under Mac. Clover will do it automatically, based on the detected CPU model, video card, and mobility feature. However, you may want another choice. Take the MacTracker program and select the Mac model that you like best, and then search the Internet or your friends for all the numbers and serial numbers from this model. There is not much to comment on here. These parameters are not for dummies. Know them

then change them, you will not be able to figure them out at random. Calculate them too is forbidden.

```
<key>ProductName</key>

<string>MacBook1,1</string>
```

SMBIOS.table1->ProductName

can specify only the product name, and Clover will calculate all other parameters corresponding to this model from its own tables. The other parameters can be omitted, but if you want other parameters, not the default, enter them too. New parameters will have priority. However, the list of names familiar to Clover is limited, only 83 models, I will not list them. For other options, fill in all fields manually. If you used a model unknown to Clover, all fields will initially be filled in with data from MacPro3.1, iMac13.1 starting with revision 3900, and then overwritten with your data, if you gave it. Except for BoardVersion, which will automatically repeat the model. If the model is not specified, then Clover will substitute something from this list, look in the menu to see how satisfied you are with this choice. Change at your discretion. It is advisable to enter your own serial numbers. You can take a sample and change one letter in the middle. This usually works. The first three and the last four cannot be changed.

```
<key>SmUUID</key>

<string>00000000-0000-1000-8000-010203040506</string>
```

SMBIOS.table1->Uuid

It seems that it makes sense to write here the MAC address of your network card (the last six pairs of characters). This GUID will also be used if CustomUUID is not specified. It is better not to specify this parameter.

```
<key>Family</key>

<string>iMac</string>
```

Same as **Model**, but without the model number.

```
<key>FirmwareFeatures</key>

<string>0xC0001403</string>
<key>#FirmwareFeaturesMask</key>
<string>0xFFFFFFFF</string>
```

SMBIOS.table128->FirmwareFeatures

These numbers are outside the SMBIOS standard, this is something specific to Apple. Different real Macs can have different numbers, there is no description anywhere, except in the bless source code

```
    && (featureFlags & 0x00000001)) {
  contextprintf(context, kBLLogLevelVerbose,  "Legacy mode suppported\n");
```

Therefore, we need to have an odd number here. In new revisions of Clover, Sherlock made an automatic calculation of the "best" value. I don't know what effect this has. Vit9696 did some work to define these bits, see
https://github.com/acidanthera/EfiPkg/blob/master/Include/IndustryStandard/AppleFeatures.h

```
<key>ExtendedFirmwareFeatures</key>

<string>0x8FE001403</string>
<key>ExtendedFirmwareFeaturesMask</key>
<string>0xFFFFFFFFFF</string>
```

Present since revision 5140.

FirmwareFeatures were 32-bit, and apparently, Apple found this to be not enough, they decided to allocate another 32-bits for this matter, so ExtendedFirmwareFeatures is now a 64-bit replacement.

They are also stored in SMBIOS.table128, but the system actually takes it from NVRAM variable, and if ExtendedFirmwareFeatures is present, then FirmwareFeatures are not asked. And the need for these bits arose in the Monterey system, for its installation a bit is required (thanks to vit9696)

```
#define FW_FEATURE_SUPPORTS_LARGE_BASESYSTEM         0x800000000U // 35
```

```
<key>PlatformFeature</key>

<integer>3</integer>
```

SMBIOS.table133->PlatformFeature

This parameter is found in real Macs and is used by the Captain, however, what it affects is not fully known. Here is some information (thanks to vit9696).

Bit 0x2 - the memory is soldered, non-replaceable.

Bit 0x4 - built-in without connectors, for IQSV.

Bit 0x8 - external processor power management, obviously for virtual machines.

Bit 0x10 - sound when starting the computer. For Hackintosh, this bit is useless, we ourselves determine whether to set the sound (AudioDxe.efi) or not. If the value is not specified, then table 133 will not be created, as on older models the real ones.

```
<key>BoardSerialNumber</key>

<string>C02032101R5DC771H</string>
```

SMBIOS.table2->SerialNumber

Clover supplies one specific parameter. You must substitute your own numbers. It is needed for iCloud and iMessage to work. The length must be 17 letters, capital Latin and numbers. **The number embedded in Clover has most likely been banned for a long time.**

```
<key>BoardType</key>

<integer>10</integer>
```

SMBIOS.table2->BoardType

This parameter was introduced for MacPro, which has not 10 — Motherboard, but 11 — ProcessorBoard, apparently for historical reasons. The meaning is not obvious, but it is noticeable on the System Profiler.

```
<key>BoardVersion</key>

<string>MacBook1,1</string>
```

SMBIOS.table2->BoardVersion

Yes, the model should be specified here too..

```
<key>BiosReleaseDate</key>

<string>05/03/10</string>
```

This is the Boot ROM release date, indicated in the BiosVersion key.

```
<key>Mobile</key>

<true/>
```

Actually, Clover always correctly calculates whether the given platform is mobile (i.e. battery-powered, requiring energy saving) or not. And the parameter is needed if for some reason we want to deceive the system, to indicate that we have no battery, or vice versa.

```
<key>ChassisType</key>

<string>0x10</string>
```

SMBIOS.table3->Type

This parameter serves as an indirect indication of whether we have a mobile platform. Here is a table of SMBIOS standard

```
MiscChassisTypeOther                = 0x01,
MiscChassisTypeUnknown              = 0x02,
MiscChassisTypeDeskTop              = 0x03,
MiscChassisTypeLowProfileDesktop    = 0x04,
MiscChassisTypePizzaBox             = 0x05,
MiscChassisTypeMiniTower            = 0x06,
MiscChassisTypeTower                = 0x07,
MiscChassisTypePortable             = 0x08,
MiscChassisTypeLapTop               = 0x09,
MiscChassisTypeNotebook             = 0x0A,
MiscChassisTypeHandHeld             = 0x0B,
MiscChassisTypeDockingStation       = 0x0C,
MiscChassisTypeAllInOne             = 0x0D,
MiscChassisTypeSubNotebook          = 0x0E,
MiscChassisTypeSpaceSaving          = 0x0F,
MiscChassisTypeLunchBox             = 0x10,
```

Clover selects the value, as set in real Macs, in accordance with the selected you as a model. What this affects, other than mobility, I don't know.

```
<key>ChassisAssetTag</key>

<string>LatitudeD420</string>
```

SMBIOS.table3->AssetTag

This field is never filled in on real Macs, so we can use it for for our needs, for example, we will link it to the HWSensors3 project.

```
<key>SmbiosVersion</key>

<string>0x0300</string>
```

Apple has its own SMBIOS standard, which does not match others, previously it was most similar to the 2.4 standard, and Apple inserted such a number. But in meaning it was more similar to 2.6, so Clover previously put the number 2.6. Now on real Macs you can see the number 3.0, but this is a lie, the standard is still 2.x. However, we can put our own the number doesn't seem to affect anything.

```
<key>BiosVersion</key>

<string>IM131.88Z.F000.B00.1907241303</string>
```

This set of numbers and letters determines whether your hackintosh will ask for a BIOS update, like real. But you don't need that, so you want to put the last values here if you know. And by default Clover will give out its own numbers, which with each revision are updated. You don't have to enter your numbers, just update Clover.

```
<key>EfiVersion</key>

<string>288.0.0.0.0</string>
```

Similar, but I'm not sure if this number affects the need for an update. Shows up as Boot ROM Version. Different models use either BiosVersion or EfiVersion.

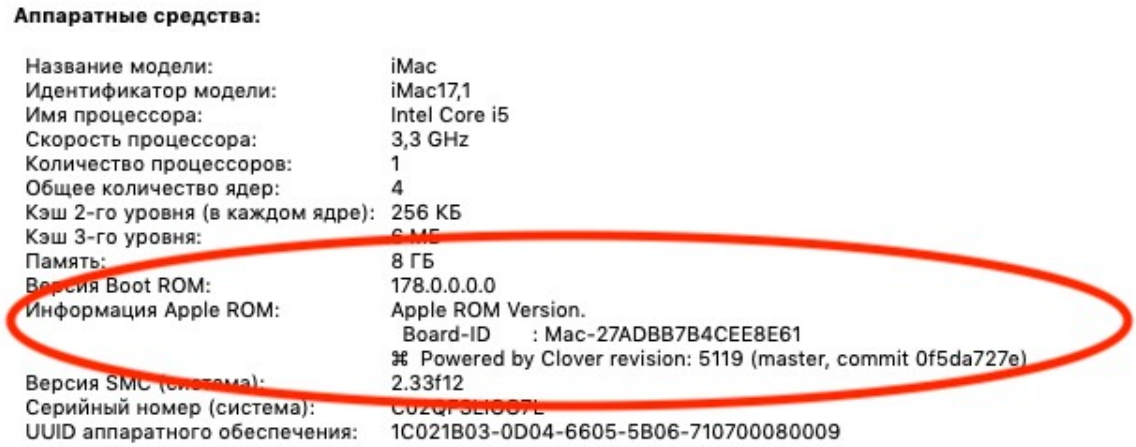
```
<key>BiosVendor</key>

<string>Apple Inc.</string>

<key>BoardManufacturer</key>

<string>Apple Inc.</string>
```

```
<key>LocationInChassis</key>
<string>MLB</string>
<key>MemoryRank</key>
<integer>2</integer>
<key>Version</key>
<string>1.0</string>
<key>Manufacturer</key>
<string>Apple Inc.</string>
```

What the parameters listed above (BiosVendor, BoardManufacturer, FirmwareFeatureMask, LocationInChassis, MemoryRank, Version, Manufacturer) are responsible for can be understood from their names. It is better not to set them manually if you do not know for sure what these changes will lead to. It is better to delete or comment out these parameters and Clover will automatically generate them when the system boots, trust it.

```
<key>NoRomInfo</key>
<false/>
```

Clover can generate SMBIOS.table11 with its firmware numbers. It is very convenient to see these values in About Mac (AppleROM Information).



Strange but this information lost in real Macs may be because of some separators. If you want to look like a real mac owner, set this NoRomInfo key to <true/>.

```
<key>Trust</key>
<true/>
```

The parameter is used to resolve the dispute between SMBIOS and SPD, whose memory parameters are considered more accurate, in addition to the fact that they also perform internal checks. By default, it is true, that is, the SMBIOS (DMI) values are more accurate. If, however, neither with true nor false you can get the "correct" memory display in the system, you have the option to write everything manually (starting with revision 1896)

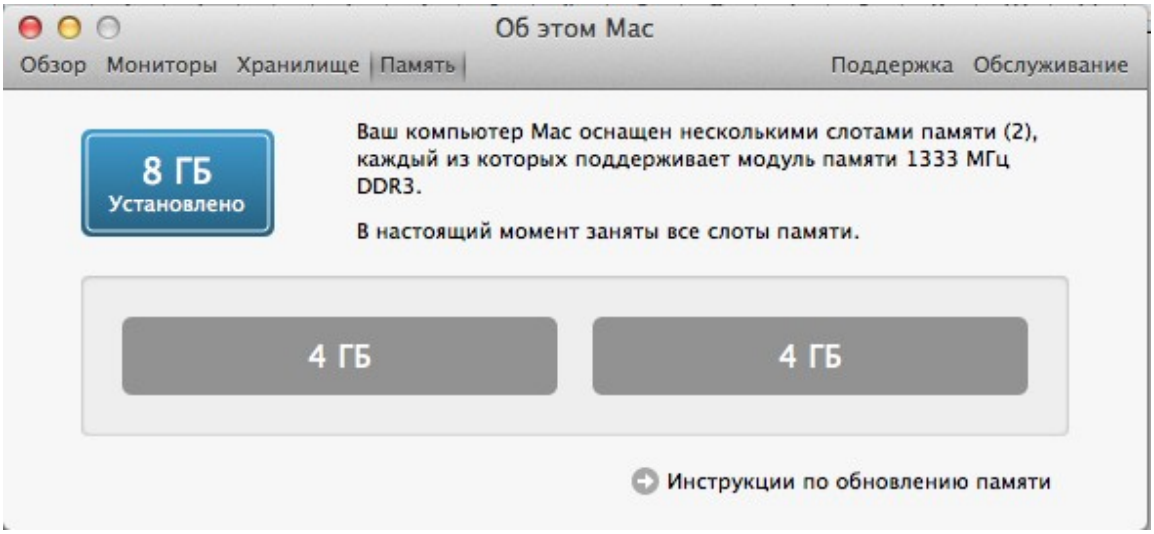
```
<key>Memory</key>
<dict>
```

```
      <key>Channels</key>
      <integer>1/2/3</integer>
      <key>SlotCount</key>
      <integer>24</integer>
      <key>Modules</key>
      <array>
         <dict>
            <key>Slot</key>
            <integer>0</integer>
            <key>Size</key>
            <integer>2048</integer>
            <key>Frequency</key>
            <integer>1600</integer>
            <key>Vendor</key>
            <string>Some Company</string>
            <key>Part</key>
            <string>123456ABCDEF</string>
            <key>Serial</key>
            <string>ABCDEF123456</string>
            <key>Type</key>
            <string>DDR/DDR2/DDR3</string>
         </dict>
         ...
         <dict>
            <key>Slot</key>
            <integer>N</integer>
            <key>Size</key>
            <integer>2048</integer>
            <key>Frequency</key>
            <integer>1600</integer>
            <key>Vendor</key>
            <string>Some Company</string>
            <key>Part</key>
            <string>123456ABCDEF</string>
            <key>Serial</key>
            <string>ABCDEF123456</string>
            <key>Type</key>
            <string>DDR3</string>
         </dict>
      </array>
   </dict>
```

Some explanations:

Channels — the number of memory channels. Very old computers had one channel. Modern computers have two. There are some configurations (Clarkdale, for example) where there are three channels, that is, three-channel memory.

SlotCount — the total number of slots where you can insert memory modules. It is displayed in the About window. Now we draw an array of modules, describing only occupied slots. We do not even mention empty ones. In the Slot key, we write its number from 0.



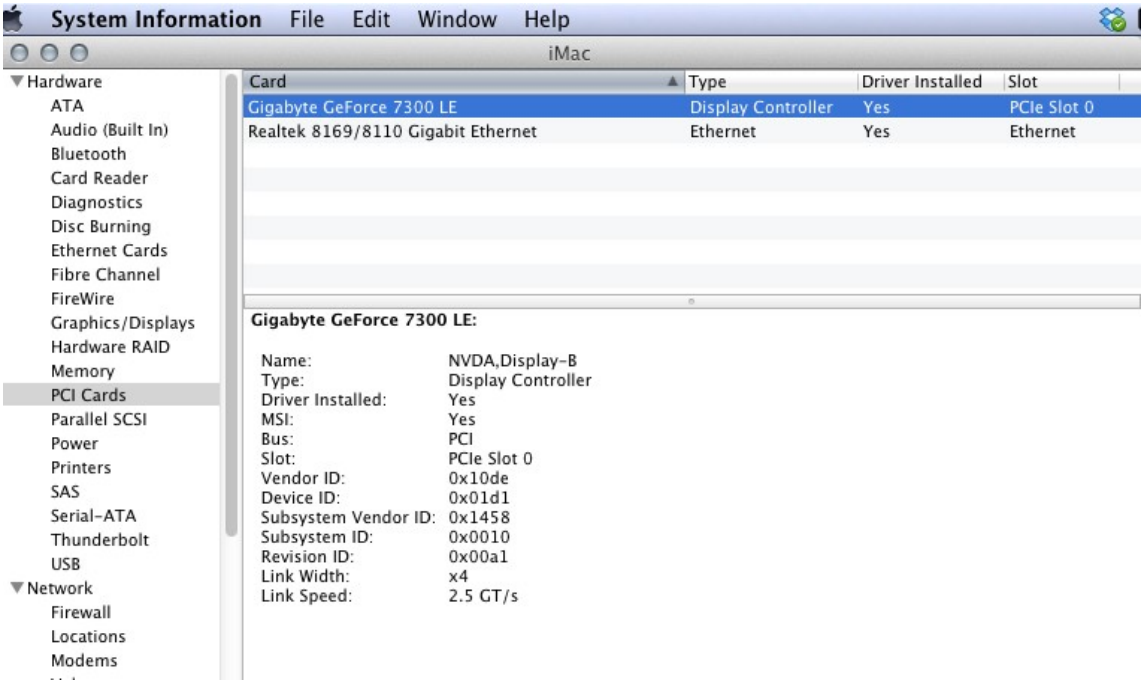
We write the size in megabytes and the speed in megahertz. We do not leave empty fields. In the serial number (Serial) and in the inventory number (Part) only capital letters, numbers, minus signs and periods are allowed.

With this, let me close the issue with the correct display of memory in the system. (And still there was a poodle who said that it is displayed not as he wrote! In fact, he wrote it incorrectly)

<key>Slots</key>

```
<array>
    <dict>
        <key>Device</key>
        <string>Nvidia</string>
        <key>ID</key>
        <integer>2</integer>
        <key>Type</key>
        <integer>16</integer>
        <key>Name</key>
        <string>PCIe Slot 0</string>
    </dict>
```

This registers PCI devices in System Profiler. This is what it looks like:



To fill these properties, write in the config

```
<key>SMBIOS</key>
<dict>
	<key>Slots</key>
	<array>
		<dict>
			<key>Device</key>
			<string>Nvidia</string>
			<key>ID</key>
			<integer>2</integer>
			<key>Type</key>
			<integer>16</integer>
			<key>Name</key>
			<string>PCIe Slot 0</string>
		</dict>
		<dict>
			<key>Device</key>
			<string>LAN</string>
			<key>ID</key>
			<integer>3</integer>
			<key>Type</key>
			<integer>1</integer>
			<key>Name</key>
			<string>Ethernet</string>
		</dict>
	</array>
```

And Clover will form such tables. In order for the corresponding _SUN properties to appear in the DSDT, if they do not exist yet, you need to set the patch mask specifically for these devices. For this example, it is

```
<key>ACPI</key>
<dict>
	<key>DSDT</key>
	<dict>
		<key>Fixes</key>
		<dict>
			<key>FixDisplay_0100</key>
```

```
                        <true/>
                        <key>FixLAN_2000</key>
                        <true/>
                        <key>NewWay_80000000</key>
                        <true/>
                    </dict>
```

If you write these properties manually, they must match the ID

```
                Device (GFX0)
                {
                    Name (_ADR, Zero)  // _ADR: Address
                    Name (_SUN, 0x02)
….
               Device (GIGE)
                {
                    Name (_ADR, Zero)  // _ADR: Address
                    Name (_SUN, 0x03)
```

Avoid ID = 0x00 and 0x01 due to optimizations in Zero and One. Clover may not be able to handle them with such a patch. At the moment, this trick is only possible with ATI, NVidia, LAN, WIFI, Firewire devices . These are predefined names, Clover will find a device that matches this name. If you want a more accurate match, then first look in your OEM SBMIOS, which you can get from the DarwinDumper report, what tables #9 you have, what SUN are bound to which devices. Then correct the DSDT to match these assignments, and add your own. Example:

```
Handle 0x0905, DMI type 9, 17 bytes
0000: 09 11 05 09 01 a6 08 03 03 02 00 04 02 00 00 00
0010: fe

System Slot Information
      Designation: Ethernet
      Type: x1 PCI Express x1
      Current Usage: Available
      Length: Short
      ID: 2
      Characteristics:
            3.3 V is provided
            Hot-plug devices are supported
      Bus Address: 0000:00:1f.6
```

That is, I have an Ethernet controller at address 0000:00:1f.6, and it is supposed to have _SUN=2 (highlighted in green). This is the device in DSDT, but it does not have the _SUN property!

```
        Device (GLAN)
        {
            Name (_ADR, 0x001F0006)  // _ADR: Address
```

We need to edit! Maybe in the future Clover will learn to do it automatically. Slot->Type is the slot type from the list PCI, PCIe x1, PCIe x2,... PCIe x16, which are encrypted for brevity with numbers 0, 1, 2, ...16. In this example, PCIe x1 is encrypted as Slot->Type=1. But since table #9 already exists for this device, it does not need to be written to the config, it is enough to write it in DSDT.

```
        Device (GLAN)
        {
            Name (_ADR, 0x001F0006)  // _ADR: Address
            Name (_SUN, 0x02)
```

CPU

This group of parameters helps with CPU determination when internal algorithms can't cope.

<key>FrequencyMHz</key>

<string>3200</string>

The base frequency of the processor in MHz. Clover usually gets this value by calculating based on the ACPI timer, but if it turns out wrong, you can substitute it through this key. This key affects only the number in the system profiler. Cosmetics! For example, for Hazwells the nominal value is 1800, and the initial speed is 2400. We will work on 2400, and for the profiler we will write 1800.

<key>BusSpeedkHz</key>

<string>133330</string>

This parameter is the base bus frequency, is critical for the system to operate, and is passed from the bootloader to the kernel. If the frequency is incorrect, the kernel will not start at all, if the frequency is slightly off, there may be problems with the clock, with sound, and very strange system behavior. The value in DMI is stored in MHz, and this is not exact, more correctly calculated from the CPU frequency, but you can choose your value more accurately, and write it in this key in kilohertz. For example, in my DMI it is written 100 MHz, and for the clock it became better when I wrote 99790 kHz. One thing. Some manufacturers have a different concept of what BusSpeed is, and what FSBSpeed is, and write in the BIOS a value four times larger. You can figure out the correctness by the range: it should be from 100 to 400 MHz, or by the formula CPU Frequency = Bus Frequency * CPU Multiplier. It is clear that if ASUS writes the bus frequency of 1600 MHz, and the processor multiplier is 8, then the formula will not converge, processors for 12.8 GHz do not exist. In reality, it is necessary to divide by 4. Starting with revision 1060, there is an auto-detection of frequency based on the ADSI timer, and it calculates these values better than specified in the DMI.

<key>UseARTFrequency</key>

<false/>

SkyLake processors have a new base frequency parameter that changes in a smaller step than the bus frequency, the so-called ARTFrequency, its value is usually 24 MHz. Clover can calculate it and pass it to the core, and the Captain will understand and use it. However, in practice, the calculated frequency leads to inaccurate work, so it can simply be disabled, in which case the system core will act in its own way. In new versions of Clover, this figure is rounded, as vit9696 believes, there can be only three values, and they are round, up to 1 MHz. No, not three! So Clover does not use the OpenCore algorithm for this parameter. Clover does it better. And I also doubted about the roundness, I will have to cancel the rounding.

<key>QPI</key>

<string>4800</string>

In the system profiler, this value is called Processor Bus Speed or simply Bus Speed. Chameleon has an algorithm for calculating it for Nehalem family processors (and even that one is incorrect!). Clover has a corrected algorithm based on Intel datasheets. The AppleSmbios kext source code considers two options: either the value is already written in SMBIOS, as the manufacturer wrote it, or BusSpeed*4 is simply calculated. After much debate, this value was moved to the config then write whatever you like (MHz). This does not affect the operation in any way - pure cosmetics. According to the latest information, QPI only makes sense for Nehalem processors, for everyone else it is necessary to have BusSpeed*4. Or nothing at all. If you forcefully write 0, then DMI table 132 will not be generated at all. Someone claims that this is exactly how you need to do it on modern Macs.

<key>Type</key>

<string>0x0201</string>

This parameter was invented by Apple and is used in the "About this Mac" window, internally converting such a constant into a processor designation. Otherwise, "Unknown processor" will be shown. Why couldn't you call CPUID? (because there was still PowerPC). Or look in SMBIOS in table 4? No, Apple has its own worldview, and we have to adapt to which processor is encrypted how. Basically, Clover knows all the ciphers, but since progress does not stand still, the ability to manually change this parameter is left. The correctness of this parameter is controlled in the "About this Mac" window. Again, pure cosmetics. There is information from vit9696 https://github.com/acidanthera/EfiPkg/blob/master/Include/IndustryStandard/AppleSmBios.h

The group of parameters concerning C-state has now been moved back to the ACPI section, ACPI->SSDT. The following keys are defined here.

```
<key>C2</key>
<true/>
```

For modern computers, set this to false.

```
<key>C4</key>
<true/>
```

According to the specification, either C3 or C4. We choose C4. For Ivy we set false.

```
<key>C6</key>
<true/>
```

C6 is known only on mobile computers, however, you can try to enable it on a desktop. On Ivy and Hazewell, set it to true.

I'll note that with these C-states people often complain about bad sound/graphics/sleep. Be more careful, or exclude them altogether.

```
<key>Latency</key>
<integer>250</integer>
```

This is the delay for turning on the C3 state. The critical value is 0x3E8=1000. Less — speedstep turns on, more — it doesn't turn on. On natives it's always 0x03E9, that is, speedstep doesn't work. On Hacks we have to choose what we want, to be like a native, or to turn on power management. A reasonable value in the second case is 0x00FA, as is found on some laptops.
MacPro5.1 = 17
MacPro6.1 = 67
iMac13.2 = 250

```
<key>SavingMode</key>
 <integer>7</integer>
```

Another interesting parameter for speedstep control. It affects the MSR 0x1B0 register and determines the processor behavior:
0 - maximum performance
15 - maximum energy saving.
With my iMac12 model, intermediate states appeared with the last two keys. However, I have no exact evidence of what affects what. However, if in the preboot log written

```
0:162 0:000 Energy PerfBias is not visible:
```

Then this parameter is useless.

```
<key>QEMU</key>
 <true/>
```

When testing Clover in a QEMU virtual machine, I found that it incorrectly emulates an Intel processor. This key was made as a temporary measure, however, it does not fix everything. No miracle has happened yet. But I can load the system in single mode.

<key>TurboDisable</key>

```
<true/>
```

Useful for laptops to prevent them from overheating.

<key>HWPEnable</key>

```
<true/>
```

Starting with revision 3879, Intel Speed Shift technology for Skylake processors has been implemented. Author goodwin_c. If true, then 1 is written to the MSR 0x770 register. One problem, it is not yet clear what to do with this. If the computer is put to sleep and then woken up, the MSR 0x770 value will be reset to 0. And Clover is no longer able to set it back. There is only one option for now, we always set 0, that is, in this item <false/>, and then somewhere in the system we try to set 1 in this register by other means. There is an assumption that the system itself will set this bit, subject to other conditions, for example, the correct model in SMBIOS. It is more correct to manage this with a kext, which will set one even after waking up
https://github.com/headkaze/HWPEnable

<key>HWPValue</key>

```
<string>0x30002a01</string>
```

This value was found to be the most suitable. This value will be written to MSR 0x774. Only if MSR 0x770 is set to 1.
Otherwise, this register is not accessible.

<key>TDP</key>

```
<integer>95</integer>
```

This is Thermal Design Power, taken into account in p-states when generating Processor Power Management
tables.

Graphics

This group of parameters is used to inject video card properties, as Natit.kext does, for example. There are many parameters that are actually injected, but these are mostly constants, some are calculated, some are set in an internal table, and only very individual parameters are entered through the config.

<key>GraphicsInjector</key>
<true/>

Actually enabling this injection function. By the way, it is enabled by default, because injection should work with a clean config - a condition for starting the system. It is worth disabling injection if you know a better way. For some modern cards, such as Nvidia Kepler or Radeon 6xxx, injection is disabled by default, because the native startup works. Incomplete, but you can enter the desktop. In revision 1921+ this parameter is outdated, but supported, now video cards are injected separately, by vendors, because modern computers almost always have built-in Intel, and sometimes there is no need to enable its injection.

<key>Inject</key>

```
<dict>
	<key>Intel</key>
	<false/>
	<key>ATI</key>
	<true/>
	<key>NVidia</key>
```

```
            <false/>
        </dict>
```
Injection of two dozen parameters that are calculated not only by the card model, but also from its internal characteristics, for example, after analyzing its video BIOS. For Nvidia, its NVCAP is calculated, for Intel, dozens of parameters are selected (thanks to Sherlocks), for ATI, parameters depending on connectors. Well, listing all this would take another two hundred pages. Moreover, none of this is needed for modern cards, Apple has made sure that they work out of the box. These injections were used before WEG appeared. Now, when all the work on setting up graphics is done by the WhatEvergreen kext, it is recommended to disable all these injections when using this kext.

<key>VRAM</key>

```
<integer>1024</integer>
```
The amount of video memory in MB. Actually, it is automatically determined, but if you enter the correct value, no one will suffer. In reality, however, I do not remember a single case where this parameter helped someone in any way. If you see 7MB, do not try to change this parameter, it is useless. You need to start the video card. For example, for mobile Radeon there is a trick to use LoadVBios=true and the memory will become correct.

<key>LoadVBios</key>

```
<true/>
```
Loading the video BIOS from a file that should be in the EFI/CLOVER/OEM/xxx/ROM or EFI/CLOVER/ROM folder and have the file name vendor_device.rom, for example 1002_68d8.rom. This sometimes makes sense if you use a patched video BIOS. At Ermak's request, starting with revision 3222, you can use a longer file name that includes the subvendor and subrevision 10de_0f00_1458_3544.rom. He needs this to test different video cards on one computer. There are also problems when the video card does not show the system its video BIOS, although the system requires it, for example in the case of mobile Radeons. In this case, you can set this parameter to Yes, but do not slip any file. Clover will take Video BIOS from legacy memory at address 0xc0000, oddly enough, it is almost always there, and now Clover injects it into the system, and the mobile Radeon turns on! One more clarification. It turns out that the BIOS flashed into the ROM of the card does not match the one formed at address 0xc0000 that is the shadow of the ROM. So, we need it, the shadow, and not the BIOS that we burn with the programmer. In short. For mobile Radeons, we set Yes, although there is no file, for other cards No. History has not recorded other options. And now new times have come. For computers with UEFI-only BIOS, there is no Video BIOS at the legacy address. We put it in the file and wait for new solutions. Or ... does it work like this?

<key>Connectors</key>

```
<array></array>
```
The parameter is reserved for the future, which has not yet arrived. It does not carry any functionality yet.

<key>DualLink</key>

```
<integer>0</integer>
```
By default the value injected is 1, but for some old configurations this parameter=1 results in quadrupling the screen. Setting it to 0 helps, as in the example above.

<key>BootDisplay</key>

```
<integer>1</integer>
```

Specifies which display is the primary one. It is the one that will light up at startup and wake up after sleep. Usually this is number 0, but sometimes the outputs are numbered in the wrong order, look in the ioreg to see how it is for you. Revision 3399.

<key>PatchVBios</key>

```
<true/>
```

Clover makes a correction to the Video BIOS shadow at 0xC0000 so that it supports the video mode that is the maximum for the connected monitor. For example, the monitor's EDID has a mode of 1920x1080, but the Video BIOS does not have one. Clover will register it as the first mode and start using it. If the monitor itself does not generate EDID, it can be injected, as shown below. There were cases when enabling this patch led to panic, a black screen when trying to boot. For the first attempt, disable this option. Or the patch uses the value from the config.plist

```
<key>GUI</key>
<dict>
	<key>ScreenResolution</key>
	<string>1440x900</string>
```

If the automation made a mistake, you can register the Video BIOS patch manually, using the standard Find/Replace algorithm.

<key>PatchVBiosBytes</key>

```
<array>
	<dict>
		<key>Find</key>
		<data>gAeoAqAF</data>
		<key>Replace</key>
		<data>gAeoAjgE</data>
	</dict>
</array>
```

It is possible to make several patches 0,1,2 in one BIOS... For example, success with Nvidia came with four patches. This example is from the Video BIOS ATIRadeon HD6670, replacing the 1920x1440 mode with a more acceptable 1920x1080. With this method, you should choose a mode with the same horizontal. To successfully set the full screen resolution in the Clover interface, and then in the system, if safe mode is used (without video drivers), you must have EDID. A group of parameters has been introduced for this

<key>EDID</key>

```
<dict>
```

This dictionary contains the following items:

<key>Inject</key>

```
<true/>
```

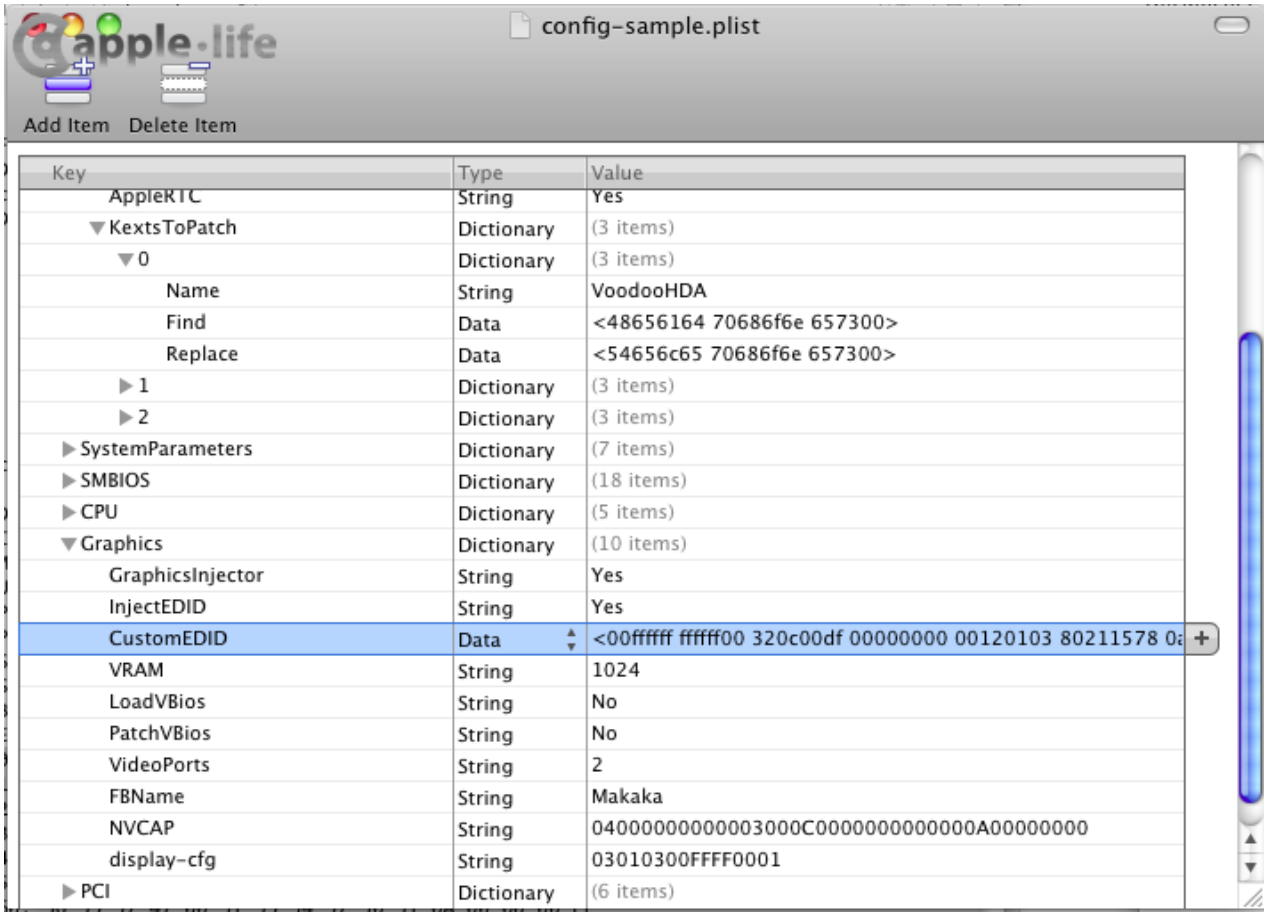
1. There are monitors without DDC, for example laptop panels.
2. There are options when DDC is present, but Apple drivers do not see it.

For the second option we simply set Inject=yes, and Clover will extract the EDID drivers, itself. The need for such an action is noted in the topic about mobile radeons. For UEFI-only computers we consider the first option, since Clover will not extract the Video BIOS. For the first option, we still have to write a new EDID manually. We write it like this:

<key>Custom</key>

```
<data>AP///////wAyDADfAAAAAAASAQOAIRV4CunVmVlTjigm
UFQAAAABAQEBAQEBAQEBAQEBAQEB3iGgcFCEHzAgIFYAS88QAAAY3iGgcFCEHzAgIFYAS88QAAAAAAAA
/gBXNjU3RwAxNTRXUDEKAAAA/gAjMz1IZYSq/wIBCiAgAJo=</data>
```

By the way, the artificially slipped EDID will also be used for automatic patching video bios. See above. Where to get EDID? Well, since it can't be extracted from this computer, we'll take someone else's. The main thing the condition is that there is a correct maximum resolution. In my sample config-plate I put EDID from Dell Inspiron. Matrix 1440x900. The letters in this example are standard XML encoding, but if you look at it through PlistEditor, then we see a more human picture



Another option for making EDID is to use the ViewSonic EDID Editor program (version 3.1.5), which, if desired, can be easily ported to OSX. There are also OSX editors, for example AW_EDID_Editor. But this does not apply to Clover itself. Study the theory. Clover gives you the ability to inject your own EDID, a good, high-quality one. There is also information that Apple drivers check the manufacturer, so there was a patch has been developed so that the automatically extracted EDID is corrected on Apple.

```
<key>ProductID</key>
<string>0x9221</string>
<key>VendorID</key>
<string>0x1006</string>
```

I can't say that this helped anyone in any way. Quite the opposite, I substituted the values, and the brightness stopped adjusting.
New keys

```
<key>HorizontalSyncPulseWidth</key>
<string>0x11</string>
```

is such a parameter in the EDID specification, written in bytes 63 and 65 in the Detailed Timing section. Some hackers have discovered that the parameter affects the well-known problem of eight apples. Look for explanations on the forums, who managed to achieve what by changing this parameter. Yes, it does, and even a lot!

```
<key>VideoInputSignal</key>

<string>0x80</string>
```

The second parameter from the same hackers. This is byte 0x14 in EDID, defines the properties of the connector.
Bit 7 - analog=0 or digital=1.
bits 6 - 1 are defined only for analog signals. bit 0 for digital means that the signal is compatible with the VESA DFP 1.x standard.

```
<key>VideoPorts</key>

<integer>2</integer>
```

Number of video outputs on the card, including TVO and/or HDMI. The selected frame from the Apple list may not match our real card. Affects the number of injected connectors. May help combat false monitors.

```
<key>FBName</key>

<string>Makaka</string>
```

This parameter is specific to ATI Radeon, for which there are three dozen different framebuffers without any pattern to each other. Clover automatically selects the most suitable name from the table for most known cards. However, other users of exactly the same card dispute it, they need a different name. So write in this parameter what you think is most correct. General rule: if you don't know what to write, delete the parameter altogether. But don't write this monkey! I wrote it specifically for the sake of absurdity – no, they copy it into their config anyway! There is an idea that the whole difference in these frames is in the set of connectors, and since you are going to patch them anyway, it makes no difference which one you take as a basis. Unless the frame should match the video card family. For example, Wormy will not work with Radeon 6670. And suddenly it came in handy again! In Monterey 12.3.1, a performance drop in Navi cards was noticed, and the solution turned out to be to inject a named framebuffer, namely

```
RX6900 -> Carswell
RX6800 -> Belknap
RX5700 -> Adder
RX5500 -> Python
RX6600 -> Henbury
```

Performance returned to its previous level. Well, WhateverGreen will have to rest in this case, it is not designed for a named framebuffer.

For RX570 you can use Orinoco, but it doesn't have much effect. Cosmetics. Who knows, maybe it will come in handy someday.

```
<key>RadeonDeInit</key>

<true/>
```

This key works with ATI/AMD Radeon 6xxx and higher cards with legacy BIOS. Or maybe 5xxx, haven't seen any reviews. It corrects the contents of the GPU registers so that the card is properly initialized, and MacOSX drivers work with it as they should. The card turns on at startup, and when waking up from sleep. Thanks to vit9696 and Mieze. It seems to make sense only for cards that do not have GOP in the video BIOS.

```
<key>NVCAP</key>
<string>040000000000003000C000000000000A00000000</string>
```

This is a parameter for NVidia video cards, it configures the types and purposes of video ports. This line contains 40 hexadecimal digits in capital letters. There is no theory here, there is empiricism, and even with contradictory results. There is such a table, but its correctness is disputed.

Yellow = Desktop 1 (primary) | Green = Desktop 2 (secondary)

DVI+VGA

	TV	DVI 2	VGA 2	VGA 1	Bin string	Conversion to hex
Desktop 1	0	0	0	1	1	1
Desktop 2	1	1	1	0	1110	0e
NVCAP	TV + DVI + VGA2			VGA1	04000000 00000100 0e000000 00000007 00000000	

DUAL DVI

	DVI 2	VGA 2	DVI 1	VGA 1	Bin string	Conversion to hex
Desktop 1	0	0	1	1	11	3
Desktop 2	1	1	0	0	1100	0c
NVCAP	DVI2+VGA2		DVI1+VGA1		04000000 00000300 0c000000 00000007 00000000	

DUAL DVI + TV on hardware channel 2 (maybe not supported)

	TV	DVI 2	VGA 2	DVI 1	VGA 1	Bin string	hex
Desktop 1	0	0	0	1	1	11	3
Desktop 2	1	1	1	0	0	11100	1c
NVCAP	DVI2+VGA2+TV			DVI1+VGA1		04000000 00000300 1c000000 00000007 00000000	

DUAL DVI + TV on hardware channel 1 (maybe not supported)

	DVI 2	VGA 2	TV	DVI 1	VGA 1	Bin string	hex
Desktop 1	1	1	0	0	0	11000	18
Desktop 2	0	0	1	1	1	111	7
NVCAP	DVI2+VGA2		DVI1+VGA1+TV			04000000 00001800 07000000 00000007 00000000	

The hardware channel and desktops are intentionnaly swapped so that TV out stays in use for secondary desktop.

DUAL DVI + TV on hardware channel 1 & 2

	TV2	DVI2	VGA2	TV1	DVI1	VGA1	Bin string	to hex
Desktop 1	0	0	0	1	1	1	111	7
Desktop 2	1	1	1	0	0	0	111000	38
NVCAP	TV2+DVI2+VGA2			TV1+DVI1+VGA1			04000000 00000700 38000000 00000007 00000000	

Laptop with VGA + TV out

	TV	DVI 2	VGA 2	LVDS	Bin string	Conversion to hex
Desktop 1	0	0	0	1	1	1
Desktop 2	1	0	1	0	1010	5
NVCAP	External VGA+TV			LCD	04000000 00000100 05000000 00000007 00000000	

Laptop with DVI + TV out

	TV	DVI 2	VGA 2	LVDS	Bin string	Conversion to hex
Desktop 1	0	0	0	1	1	1
Desktop 2	1	1	1	0	1110	0e
NVCAP	External DVI+VGA+TV			LCD	04000000 00000100 0e000000 00000007 00000000	

The first byte is always 04 (05 in MacBook!). The second byte is LID=01 for laptops. You can find other ways to calculate the correct value of this line on forums. And Clover himself tries to calculate it from the BIOS.

<key>display-cfg</key>

```
<string>03010300FFFF0001</string>
```

is also a parameter only for NVidia cards. For details, see the discussions at http://www.insanelymac.com/forum/topic/215236-nvidia-injection/

However, the information provided there is debatable. The real configs can be viewed in the topic http://www.projectosx.com/forum/index.php?showtopic=370
But in general, the default config that Clover creates seems to be the best option. Just don't specify this parameter at all, let Clover do it calculate.

<key>NvidiaGeneric</key>

```
<true/>
```

If true, then instead of the name Gigabyte Geforce 7300LE, the name NVIDIA Geforce 7300LE will be used. The VideoProc program requires that the name must begin with Nvidia, AMD or Intel.

<key>NvidiaSingle</key>

```
<false/>
```

If it is true, then inject only the first card, not the second. For example, the second does not need injection.

<key>NvidiaNoEFI</key>

```
<false/>
```

If true, then adds the NVDA,noEFI property to the Nvidia injection. Explanations from FredWst http://www.insanelymac.com/forum/topic/306156-clover-bugissue-report-and-patch/page-107?p=2443062#entry2443062
Claims that without this there are artifacts on the screen on his GT640.

<key>ig-platform-id</key>

```
<string>0x01620005</string>
```

This parameter is required to launch the Intel HDxxxx video card, the dispute about specific values did not lead to a single rule, so the parameter is simply moved to the config - choose. By the way, Clover itself will offer a certain value. Now I have a result. My Skylake started only with the parameter 0x193b0000, it corresponds to the configuration that has an HDMI output, like mine.

KernelAndKextPatches

This group of parameters is for implementing binary patches on the fly. It should be noted that this is only feasible if loading occurs via kernelcache or via the parameter ForceKextsToLoad. If the kext is not loaded and is not present in the cache, these fixes do not work. Starting with version 5119, patches are made according to internal algorithms that do not depend on the system version. And for your patches, the ability to search by symbols has been made.

<key>Debug</key>

```
<true/>
```

If you want to watch the kext patching in action, this key is for developers.

```
<key>KernelCpu</key>
<true/>
```
Prevents kernel panic on unsupported CPU, specifically Yonah, Atom, Haswell for older systems. Or some Broadwell-E. This patch should be considered obsolete now, and FakeCPUID and other kernel patches should be used instead. It is important to understand that there are other algorithms in the kernel that will not work correctly with unsupported CPU, so do not expect this patch to solve all your problems. It is very doubtful that this will work with Pentium M, Pentium 4 or AMD, for such cases it is better to find a specially made kernel.

```
<key>FakeCPUID</key>
<string>0x010676</string>
```
This patch, introduced with revision 2748, serves as a replacement for KernelCpu. It does not simply block kernel panics, it substitutes the processor ID so that it responds to all calls as supported. In particular, it affects the AppleIntelCPUPowerManagement.kext kext. In this example, it substitutes the Penryn processor ID, which is supported by all versions of OSX from 10.5 up until 10.14, where it was deprecated.

```
<key>AppleIntelCPUPM</key>
<true/>
```
It turns out that the BIOS on ASUS motherboards (how many times has ASUS spoiled our mood?) writes register 0xE2 bit 14 to MSR, and the register becomes ReadOnly, but it is used in the AppleIntelCPUPowerManagement kext, and is used by writing. The authors of this fix could not think of anything better than fixing the kext itself, because the only way to return the E2 register to its former functionality is to reboot. Set Yes if you have a panic about this kext when starting the system. (yes, the E2 register has the WriteOnce property, i.e. you can write to it only once before rebooting). Relevant for Sandy processors and higher. Or reflash the BIOS. And what about other operating systems in this case? They say that this is good for Windows too.

```
<key>AppleRTC</key>
<true/>
```
Deprecated! vit9696 investigated the issue and fixed the RTC operations in Clover, now the recommended key value is `<false/>`, since it affects hibernation. Although, it is a moot point, my hibernation key is still saved in NVRAM.

```
<key>KernelLapic</key>
<false/>
```
There is a problem with lapic on HP laptops, which is solved by running with cpus=1, or now with this patch <true/>. The problem and solution have existed for a long time, since the Chameleon, but new developers have not yet reached HP laptops to understand more seriously what the trick is here.

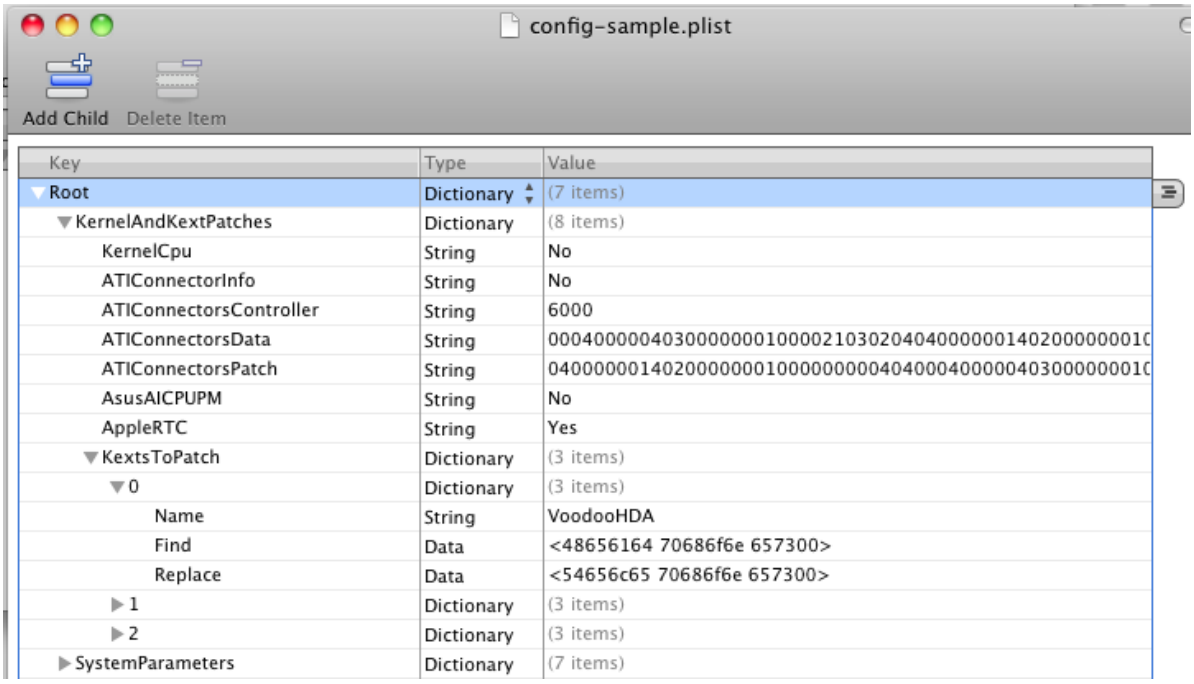
```
<key>KernelPM</key>
<false/>
```
It turns out that starting with the 10.9 system there is some CPUPM control embedded directly into the kernel. This patch prevents kernel panic for those cases when 0xE2 is locked in the BIOS. That is, for processors before IvyBridge, power management was performed by the kext AppleintelCPUPowerManagement.kext, and for newer ones this function is performed by the kernel itself, and it also encounters locked 0xE2. This patch fixes the kernel in these places. And then the 0xE2 register is not accessed at all. In general, reasoning sensibly, it is worth doing this even if the register is not locked, because the PM works like on real Apple computers, and we have completely different rules.

```
<key>KernelXCPM</key>
<false/>
```
In the 10.12+ system, XCPM support for IvyBridge processors has been discontinued. No big deal, you can go the Apple way, and those who are used to XCPM can set this key to <true/>. Also, a number of processors could well be controlled by the kernel, but Apple did not provide for them. Although, in theory, FakeCPUID should improve the situation. And starting with Monterey, this is the only option for Ivy.

```
<key>DellSMBIOSPatch</key>
<true/>
```
It has been noticed that on Dell laptops with a Skylake processor and higher, the UEFI BIOS itself corrupts our ready-made SMBIOS. It is not clear why, but this needs to be dealt with. The patch is tricky, you can't do it manually. It is only needed on Dell laptops with a Skylake processor (and higher?).

```
<key>EightApple</key>
<true/>
```
Sometimes on some hardware with some OS you can see how the apple with the progress bar at some point during loading breaks into 8 apples. Doesn't work in 11.x and higher, because the graphic kexts are loaded later than the bootloader works... WhateverGreen does such a patch automatically, it has access to it. Added in revision 5119.

```
<key>KextsToPatch</key>
<array>
```
In addition to specific patches, you can make a patch of any other kext, the principle is simple: a hexadecimal string of what to search for, and a string of what to replace it with. Starting with revision 5095, you can make patches by mask, see the paragraph Patching with Mask. Example: patch VoodooHDA to replace the name Headphones with Telephones. The condition is that the number of letters must be the same. Or less and supplemented with zeros.

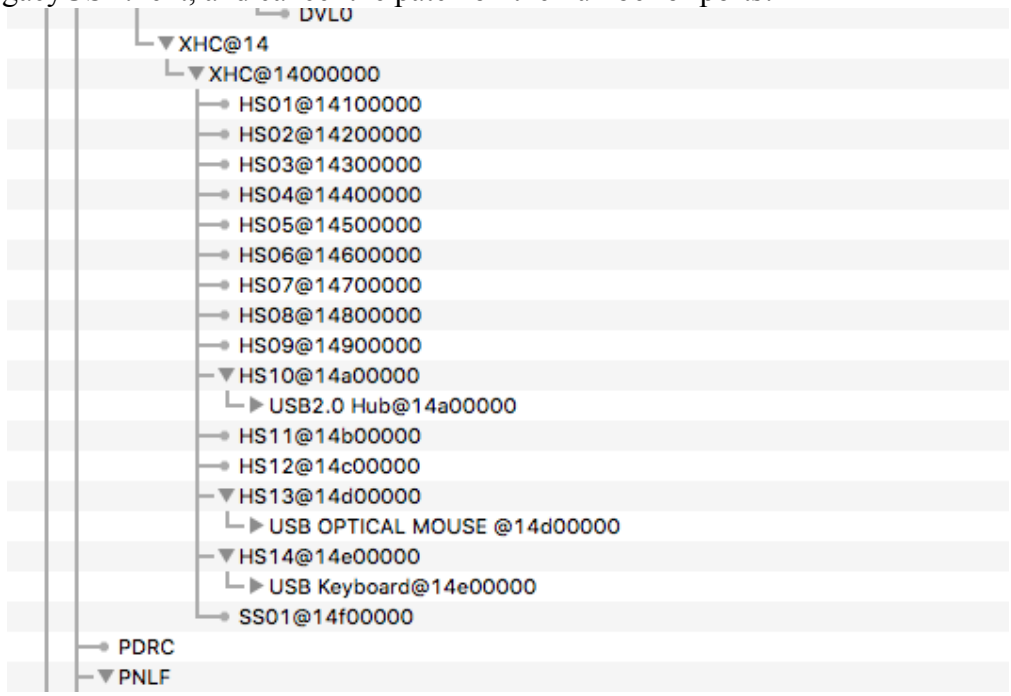


This method is successfully used to enable Trim support for SSD
http://www.applelife.ru/threads/clover.32052/page-539#post-310105

Here is another very useful patch: fighting yellow icons and non-working DVD player (which does not work for external drives): Original topic
http://www.applelife.ru/threads/Меняем-external-на-internal.38111/

```
<dict>
	<key>Name</key>
	<string>AppleAHCIPort</string>
	<key>Find</key>
	<data>RXh0ZXJuYWw=</data>
	<key>Replace</key>
	<data>SW50ZXJuYWw=</data>
</dict>
```

select the MacPro4.1 or 5.1 model, not having memory with ECC. AppleTyMCEDriver patch A simpler method was also noticed **-nehalem_error_disable** disables the AppleTyMCEDriver Thanks to AkimoA. One of the most useful patches is removing the limitation on the number of ports in the USB3 controller. The problem is that the kext numbers all the ports, first as USB2, then as USB3, and the total, according to Apple's concepts, should not exceed 15 ports. I am illustrating the result on my computer. STOP! This cannot be done! The number of ports is written into four bits of the controller, i.e. a number from 0 to 15. If you write more, the extra bit will erase other information, and you will not get any effect anyway.
Make a LegacyUSB.kext, and cancel the patch on the number of ports!



So there is 14 ports HS (usb2), and one SS (usb3), while for the chipset they are 10.
The patch is as follows (depends on the system version)

```
<dict>
	<key>Find</key>
	<data>g32UDw+DlwQ=</data>
	<key>Comment</key>
	<string>USB 3.0 limit High Sierra 10.13.4</string>
	<key>Disabled</key>
	<false/>
	<key>MatchOS</key>
	<string>10.13</string>
	<key>Name</key>
	<string>com.apple.driver.usb.AppleUSBXHCI</string>
	<key>Replace</key>
	<data>g32UGA+DlwQ=</data>
</dict>
```

The sample, by the way, illustrates additional patch keys:

```
<key>Disabled</key>
<true/>
```

You can disable questionable keys in the config file, so that you can later enable them in the Clover interface.

```
<key>Comment</key>
<string>USB 3.0 limit High Sierra 10.13.4</string>
```

The comment is also visible in the Clover interface, but has no effect on the system.

```
<key>MatchOS</key>
<string>10.13</string>
```

It often happens that a given patch is applicable only to a specific version of the system, for another version you need another patch. We write both, and specify the version, including the full 10.13.5 or shortened, as in the example. However, you are unlikely to have two systems 10.13.2 and 10.13.4, so there is no particular sense in the full version, just do not forget to update the patches along with the system update.

Sometimes it is necessary to edit not the binary part of the kext, but its info.plist. In this case, the section looks like this

```
<dict>
      <key>Name</key>
      <string>AppleHDAController</string>
      <key>Comment</key>
      <string>Patch_to_not_load_this_driver</string>
      <key>InfoPlistPatch</key>
      <true/>
      <key>Find</key>
      <string>0x04020000</string>
      <key>Replace</key>
      <string>0x44220000</string>
</dict>
```

There is one complication here. The patch is supposed to be done in the kernel cache, but if we patch the info-sheet so that the kext loads, this kext is not there yet, since it has not loaded yet. Therefore, it needs to be loaded twice. The first time with the cache ignored (the NoCache key), then FSInject will load this kext, and the second time with the cache, where it will be successfully patched. Set ForceKextsToLoad in the config. Only in 10x versions of systems. In revision 3154, and then in 3256, the info-sheet patch was corrected (thanks to user solstice). Now you can include several lines in the search, excluding all invisible characters, such as line feeds and tabs. You now need to specify the search in the <data> format, since service characters such as "<" cannot be specified in text form. The lengths of the search and replace strings may differ, but they must be set to the same length, supplemented with spaces. Example

```
<dict>
      <key>Comment</key>
      <string>Power state 1 - 0</string>
      <key>Name</key>
      <string>AppleIntelHDGraphicsFB</string>
      <key>InfoPlistPatch</key>
      <true/>
      <key>Find</key>

<data>PGtleT5Qb3dlclN0YXRlczwva2V5PjxpbnRlZ2VyPjE8L2ludGVnZXI+</data>
      <key>Replace</key>

<data>PGtleT5Qb3dlclN0YXRlczwva2V5PjxpbnRlZ2VyPjA8L2ludGVnZXI+</data>
</dict>
```

If you have a sample config with many patches, but in a particular case you only use some, then the extra ones can be disabled

```
                        <key>Disable</key>
                        <true/>
```

This was very conveniently done with the Property List Editor, where <true/> is simply represented by a check mark. In Clover revisions 3990+, these patches can be enabled and disabled in the Clover menu by checking this box.

Patching with Mask

Starting with revision 5095, the ability to perform binary patches with a mask has been introduced. This applies to KextPatches, KernelPatches and BootPatches. I will tell you in a separate place, keeping in mind all three points. It looks like this (specific data is unreal, just as a method). So, in addition to the hexadecimal string Find, we can also set a mask MaskFind, a bit mask. If some bit = 1, then we look for an exact match, if =0, then we ignore the difference. And for the Replace string, the MaskReplace mask means that bit=1 — we do the replacement, bit=0 — we leave it as it was. Example:

1. Find all the strings in the specified kext, {also works in the kernel or in boot.efi} clever or Clever. The difference is in the first letter, it differs by bit 0x20. That is, we set the search mask DF FF FF FF FF FF. This means that we achieve a match of all bytes (letters), except for the first one, in which we ignore the uppercase or lowercase letter. The search mask can be shortened, because by default all missing bytes are assumed to be FF. This means that unspecified bytes must necessarily match
2. Replace the third letter in the found words with "o", that is, we get clover or Clover respectively. MaskReplace = 00 00 FF 00 00 00 The string can be shortened on the right, since it is supposed to be filled with zero. In this case, the bytes that we do not replace could not be specified in Replace, but there is a requirement exact length match. To maintain backward compatibility of the new Clover with the old config, it is assumed that the unspecified mask (missing) consists entirely of FFFFFF, i.e. exact match to the search string, and complete replacement of all bytes with the specified ones.

Symbolic patching

Since revision 5119 we have more search and patch capabilities. General syntax

```
    <dict>
                        <key>Comment</key>
                        <string>Symbolic patch example got lapic panic</string>
                        <key>MatchOS</key>
                        <string>All</string>
                        <key>MatchBuild</key>
                        <string>All</string>
                        <key>Disabled</key>
                        <true/>
                        <key>Procedure</key>
                        <string>_lapic_interrupt</string>
                        <key>RangeFind</key>
                        <integer>200</integer>
                        <key>StartPattern</key>
                        <data>ACnHeAAx241H+oM=</data>
                        <key>MaskStart</key>
                        <data>/////wA=</data>
                        <key>Find</key>
                        <data>6AAA//+DAAAAAAAA</data>
                        <key>MaskFind</key>
                        <data>/wAA////AAAAAP//</data>
                        <key>Replace</key>
                        <data>6AAA//8xwJCQkJCQ</data>
                        <key>MaskReplace</key>
                        <data>/wAA////////////</data>
    </dict>
```

MatchOS is set to All, since we consider this patching method independent of the system version. Variants 10.x are all tenth versions, 11.5.x are all subversions of the 11.5 system.
MatchBuild — a specific indication of the system build, such as 21D5025f. You can list several builds separated by commas, or just All. If MatchBuild is missing, All is implied. But in general, the build dependency is not very important, We put
Find/MaskFind, and that should be enough to make it work only in the right builds.
Disabled is true for now, because this is not a real example. In the Clover shell, we can enable/disable patches by checking boxes. The initial state is set by this parameter.
Procedure here we write the name of the procedure we are looking for. The real name may be longer than the specified one, because the comparison is based on the presence of a substring. Be sure that such a substring occurs only in this procedure. For example, instead of _lapic_interrupt you can easily specify lapic_interrupt.
RangeFind is the length of the codes to search for. In general, it is simply the size of this procedure, or less. In this way, we speed up the search without going through all the millions of lines.
StartPattern was invented before the symbolic patch. It is the starting point from which to look for our pattern. If we know the name of the procedure, then StartPattern is hardly needed. Nevertheless, let it be. RangeFind is also applicable to it.
MaskStart is a mask for the starting point, i.e. for StartPattern. And then the Find/
MaskFind and Replace/MaskReplace pairs.

Starting with revision 2814, it became possible to force kexts loading

```
<key>ForceKextsToLoad</key>

<array>
	<string>\System\Library\Extensions\AppleHDA.kext</string>
</array>
```

This overcomes the reluctance of kexts to load. It is necessary for loading IOXXXFamily, which is necessary for loading the main kext, but it depends on this family. For example, IONetworkFamily. Or even the entire \Extra\Extensions folder (revision 2816+). Folders on the main partition are implied, other partitions/volumes/disks are not provided. Note the slash! A driver is required to perform this function FSInject.efi.
Doesn't work after BigSur.

```
<key>ATIConnectorsController</key>

<string>6000</string>
```

To fully launch ATI (AMD) Radeon 5000 and 6000 series cards, it is not enough to inject properties into the registry; you also need to adjust the connectors in the corresponding controller. In this case, we point to the 6000 controller. The following two properties indicate what to find and what to change.

```
<key>ATIConnectorsData</key>

<string>00040000040300000001000021030204040000001402000000010000000004031000000010000000000010000000000001</string>

<key>ATIConnectorsPatch</key>

<string>040000001402000000010000000004040004000004030000000100001102010500000000000000000000000000000000</string>
```

method only works for systems 10.7 and above. In 10.12 the

connectors will be different, so this method should be considered obsolete, although the calculation method is still the same.
I'll tell you in more detail how to get these numbers.
Original article by bcc9 http://www.insanelymac.com/forum/index.php?showtopic=249642 Full recipe by Xmedik in Russian with discussions http://www.applelife.ru/threads/ÿÿÿÿÿ-ati-hd-6xxx-5xxx-4xxx.28890/
I will present it here in brief, taking into account the specifics of Clover.

1. First of all, you need to get your video BIOS. Boot into CloverGUI and press F6. Your BIOS will be saved in the file /EFI/CLOVER/misc/c0000.bin, if, of course, Clover is installed in a partition with the FAT32 file system.

2. Download the radeon_bios_decode program from one of these links. Put the BIOS file c0000.bin in the same folder with this utility. Let's say this is the ~/RadeonPatch folder. Run the following commands in the terminal: cd ~/RadeonPatch ./radeon_bios_decode < c0000.bin

3. On the screen you will receive information about your connectors, which is worth copying/photographing for further use. Here's what I have on my iMac

```
iMac:test slice$ ./radeon_bios_decode <c0000.bin
ATOM BIOS Rom:
    SubsystemVendorID: 0x1458 SubsystemID: 0x2557
    IOBaseAddress: 0xe000
    Filename: R667D32I.F1
    BIOS Bootup Message:
GV-R667D3-2GI/F1

PCI ID: 1002:6758
Connector at index 0
    Type [@offset 44282]: HDMI-A (11)
    Encoder [@offset 44286]: INTERNAL_UNIPHY2 (0x21)
    i2cid [@offset 44356]: 0x92, OSX senseid: 0x3
Connector at index 1
    Type [@offset 44292]: DVI-D (3)
    Encoder [@offset 44296]: INTERNAL_UNIPHY (0x1e)
    i2cid [@offset 44383]: 0x95, OSX senseid: 0x6
Connector at index 2
    Type [@offset 44302]: VGA (1)
    Encoder [@offset 44306]: INTERNAL_KLDSCP_DAC1 (0x15)
    i2cid [@offset 44410]: 0x90, OSX senseid: 0x1
```

4. Download the ati-personality.pl script from one of the links
5. Put it in the same folder and run perl ati-personality.pl -386 >frames.txt in the terminal if you are doing this for a 32-bit system, or perl ati-personality.pl >frames.txt for 64-bit. Attention! In Sierra the texts have changed, so the patch is system-dependent.
6. Now you need to decide on the choice of a suitable framebuffer. Apple offers us a wide choice: birds, fish, and even monkeys. But the real differences there are mainly in the connectors, which we are going to change. If you don't think too much, then the simple selection option is:
5000 series: mobile - Alouatta, desktop - Baboon
6000 series: mobile - Cattail, desktop - Ipomoea
7000 series: mobile — Pondweed, desktop — Futomaki.

For the selected framebuffer, we take the connector printout from our frames.txt file obtained in step 5.

```
0000000 00 04 00 00 04 03 00 00 00 01 00 00 12 04 01 05
0000010 00 08 00 00 04 02 00 00 00 01 00 00 11 02 04 03
```

```
0000020 10 00 00 00 10 00 00 00 00 01 00 00 00 00 00 02
```

The numbers that need to be edited are highlighted in red. The blue numbers are just addresses and should be discarded. The third digit from the end is encoderid, the last digit is senseid. The first 4 digits in each line are the monitor type (or more precisely, the connector type).

```
ConnectorType
            02 00 00 00  LVDS
            04 00 00 00  DVI_DL(Dual Link)
            00 02 00 00  DVI_SL(Single Link)
            10 00 00 00  VGA
            80 00 00 00  S-Video
            00 04 00 00  DP
            00 08 00 00  HDMI
```

senseid we got in step 3 for each of our connectors. encoder can be simply zeroed everywhere. We don't pay attention to the other numbers. We get the following table:

```
0000000 04 00 00 00 04 03 00 00 00 01 00 00 10 00 01 06
0000020 10 00 00 00 10 00 00 00 00 01 00 00 00 00 00 01
0000010 00 08 00 00 04 02 00 00 00 01 00 00 12 00 04 03
```

That is, the first line is DVI-D, the second is VGA, the third is HDMI, and that's all with mine. senseid values.

And another recipe from Sergey_Galan. http://www.applelife.ru/threads/mobility-ati-radeon-hd5650m-hd5470m-hd4570m-hd4650m.29028/page-58#post-379044

The second digit from the end of HotPlugID should follow in order 00, 01, 02. This affects sleep and awakening. (highlighted in red)

```
0000000 04 00 00 00 04 03 00 00 00 01 00 00 10 00 00 06
0000020 10 00 00 00 10 00 00 00 00 01 00 00 00 00 01 01
0000010 00 08 00 00 04 02 00 00 00 01 00 00 12 00 02 03
```

Having discarded the blue numbers, we enter the rest into config.plist without spaces and line breaks. The original table in ATIConnectorsData, after our edits in ATIConnectorsPatch. See the example above in the text. I also saw a situation when the VGA connector was presented among the connectors as DVI-I (DVI-SL). And the patch worked with this usage. Recipe from eierfrucht https://applelife.ru/threads/sony-vaio-vpceb3m1r.522504/page-3#post-537081:

The most interesting moment is the FEATURES bits at the LVDS connector, I advise you to try 08 01, 08 00, 09 01 and 09 00, on one of them everything should start waking up

normally. (Highlighted in orange)

```
0000000 02 00 00 00 40 00 00 00 09 01 00 00 00 00 00 07
0000010 00 04 00 00 04 06 00 00 00 73 00 00 11 02 01 01
```

There, "Senseid LVDS panels set 0x7", because Sony VAIO.

In a new way, you need to do this

```
      <key>ForceKextsToLoad</key>
      <array>

<string>\System\Library\Extensions\AMD6000Controller.kext</string>

<string>\System\Library\Extensions\AMDFramebuffer.kext</string>
      </array>
      <key>KextsToPatch</key>
      <array>
            <dict>
                  <key>Comment</key>
                  <string>ATI Connector patch new way</string>
```

```
                        <key>Disabled</key>
                        <false/>
                        <key>Find</key>

      <data>AAQAAAQDAAAAAQAAIQMCBAQAAAAUAgAAAAEAAAAABAMQAAAAEAAAAAABAAAAAAAB</
data>
                        <key>MatchOS</key>
                        <string>10.9,10.10,10.11</string>
                        <key>Name</key>
                        <string>AMD6000Controller</string>
                        <key>Replace</key>

      <data>BAAAABQCAAAAAQAAAAAEBAAEAAAAEAwAAAAEAABECAQUAAAAAAAAAAAAAAAAAAAAA</
data>
                    </dict>
                </array>
```

For 10.12 duplicate the entire <dict> with other Find/Replace lines, they are longer. We can also set binary patches for boot.efi or for kernel. The method is the same, so I will show it on one example

<key>KernelToPatch</key>

<key>BootPatches</key>

For more information about masks, see **the Patching with Mask section**. I would like to note that this group of patches is almost never used by anyone except the developers themselves, in order to find errors in severe cases that are not obvious from an external inspection. This is how we looked for hibernation problems, this is how we looked for the moment when E2 locks in the kernel.

<key>KextsToBlock</key>

```
        <array>
            <dict>
                <key>Comment</key>
                <string>Allow IOSkywalk Downgrade</string>
                <key>Disabled</key>
                <false/>
                <key>MatchOS</key>
                <string>14.x,15.x,26.x</string>
                <key>Name</key>
                <string>com.apple.iokit.IOSkywalkFamily</string>
            </dict>
```

There is an array of kexts you want to clock. It works only for kexts included in BootKernelCollections.

Devices

A group of parameters for other PCI devices and the bus in general.

<key>Inject</key>

```
      <false/>
```

Set this value to true then the entire internal injection is replaced with the input of a single Properties string, which corresponds to Apple's injection via the APPLE_GETVAR_PROTOCOL protocol with GUID={0x91BD12FE, 0xF6C3, 0x44FB, {0xA5, 0xB7, 0x51, 0x22, 0xAB, 0x30, 0x3A, 0xE0}}; and is used on real Macs. In the old days, hackers called these EFIstrings.

```
<key>Properties</key>

<string>0207364862FA54HG345</string>
```

Deprecated! Starting with revision 4497 we make not a string of 16-digit characters, but a dictionary according to gfxutil rules.

```
    <dict>
      <key>PciRoot(0x0)/Pci(0x14,0x0)</key>
      <dict>
        <key>AAPL,clock-id</key>
        <data>AA==</data>
        <key>AAPL,current-available</key>
        <data>sAQ=</data>
        <key>AAPL,current-extra</key>
        <data>vAI=</data>
        <key>AAPL,current-in-sleep</key>
        <data>6AM=</data>
        <key>built-in</key>
        <data>AA==</data>
        <key>device_type</key>
        <string>XHCI</string>
      </dict>
      <key>PciRoot(0x0)/Pci(0x19,0x0)</key>
      <dict>
        <key>built-in</key>
        <data>AQ==</data>
      </dict>
```

For example, you can use DarwinDumper to see what plist Clover generated by default, that is, to make automatic injections, or as you did it the old way, boot into the system, and in the system call clover-genconfig >config-gen.plist in the terminal. There you look for such a dictionary, and copy it to your config, and turn off all the old injection methods. You can also look at such a dictionary in the DarwinDumper report from a real Mac with a similar configuration. The syntax is the same. Only Clover understands a little more, it understands <string>, <integer>, <true>, <false>, <real>. Today, the clover-genconfig function and the plist editor are included in the Clover.app application. Use it! In principle, the same result is achieved by inserting _DSM methods into DSDT, if it already exists, and if you work on improving it. To each his own. In principle, Properties is better than _DSM, it is native for real users, and works before the kernel starts.

```
<key>PCIRootUID</key>
<integer>0</integer>
```

It turns out that the injection of video card properties depends on what number is in DevicePath=PciRoot(0x0) or PciRoot(0x1). Previously, it was believed that this was a hardware characteristic. However, at the dawn of hackintosh construction, it turned out that this number is simply an identifier written in DSDT. Here:

```
Device (PCI0) {
      Name(_HID, EisaId("PNP0A08"))
      Name(_CID, EisaId("PNP0A03"))
      Name (_ADR, Zero)
      Name (_UID, Zero)
```

_UID=Zero – means 0, if it is equal to One, then 1. Moreover, if this number is changed by force, it will change and will work successfully. So, real Macs always have 0. And accordingly, boot.efi always assumes 0, so it is better if you correct your DSDT, Clover does this by default, and there is no such key in the config.

```
<key>Audio</key>

<dict>
      <key>Inject</key>
```

```
        <string>887</string>
        <key>ResetHDA</key>
        <true/>
        <key>AFGLowPowerState</key>
        <true/>
    </dict>
```

Inject

Injection of sound card properties. True, this only works if the device in DSDT is called HDEF, but if you rename it, you can inject the rest in another way. Also, this effort is not needed when using the VoodooHDA driver. The following value options:

NO - nothing is injected, for example, if you inject properties yourself via Properties

Detect - automatic detection of the installed sound chip in order to use its ID as a layout. Actually, this is nonsense, but very popular. In many cases, it does not interfere, and affects the display of the sound card in the System Profiler.

883 - the layout number in decimal form. This refers to Realtek ALC883. 0x0373 - the same thing in 16-bit form becomes unrecognizable. In fact, these numbers are incorrect, the correct layout, for example, 12 = 0x0C, but, oddly enough, are acceptable. With the advent of AppleALC, this parameter has acquired a new life. Now it is really a layout number, and you need to select it from the list of recommended ones for your audio codec. See the documentation for this kext.

With the new VoodooHDA.kext 3.1.2 you may also choose a layout same way as AppleALC.

ResetHDA — if the sound chip for some reason does not turn on, then this key can help with the initial launch. Also affects Windows. The need was noticed after rebooting from Windows to Mac.

AFGLowPowerState — affects the AppleHDA driver, and should apparently solve the problem with clicks in the speakers when the sound card is put to sleep/wake up. There is little evidence that the patch is effective.

<key>USB</key>

```
<dict>
        <key>Inject</key>
        <true/>
        <key>AddClockID</key>
        <true/>
        <key>FixOwnership</key>
        <true/>
        <key>HighCurrent</key>
        <true/>
</dict>
```

Inject

You can set it to false if for some reason you want to opt out of injecting USB properties, for example if you inject properties yourself via Properties.

FixOwnership

BIOS takes control of USB, and before the kernel starts we must separate USB from BIOS. It seems to be irrelevant for UEFI boot, therefore, by default it is enabled for legacy boot, and disabled for UEFI. Relevant!

AddClockID

If this property is present, the USB controller goes to sleep and does not wake up the computer. If you want to wake up from the USB mouse, set false here. But be prepared that your computer will wake up spontaneously, for example from the built-in camera.

HighCurrent

Increased current on this USB controller, needed for charging the iPad, but I did not make this value the default.

A group of parameters for disguising your devices as native OSX ones. (1971)

```
<key>FakeID</key>

<dict>
     <key>ATI</key>
     <string>0x67501002</string>
     <key>IntelGFX</key>
     <string>0x01268086</string>
     <key>NVidia</key>
     <string>0x0FE210DE</string>
     <key>LAN</key>
     <string>0x436311AB</string>
     <key>SATA</key>
     <string>0x25628086</string>
     <key>WIFI</key>
     <string>0x431214E4</string>
     <key>XHCI</key>
     <string>0x1E318086</string>
     <key>IMEI</key>
     <string>0x1E3A8086</string>
 </dict>
```

In this group of parameters you can set up relabeling your unsupported device as supported. Examples:

- AMDRadeonHD7850 has DeviceID=0x6819, which is not supported by the ATI7000Controller and ATIRadeonX3000 kexts in the 10.8 system. But there is support for DeviceID=0x6818. We make a substitution. In order for it to take effect, this fake must be somehow injected. For video cards, there are two options: either Inject->ATI=true, or DsdtFixMask includes 0x0100 (FixDisplay).

- NVidia GTX660 has DeviceID=0x1183, the card works anyway, but AGPM is not provided for it. We make a substitution for 0x0fe0, and AGPM is enabled. Since for such a card Inject->NVidia=false, then ID substitution can be done only through the DSDT patch with the mask 0x0100 (FixDisplay).

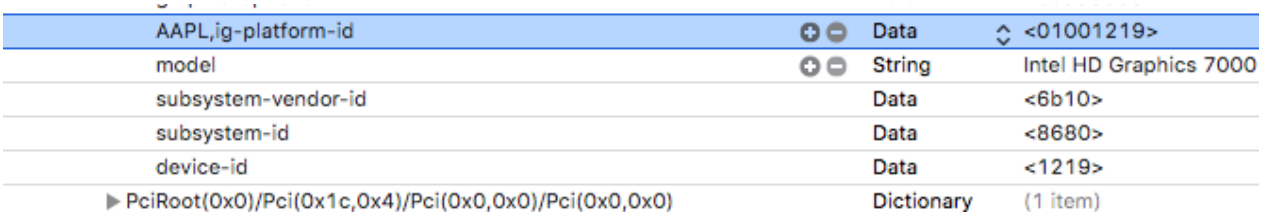
- the WiFi card in the Dell laptop is called Dell Wireless 1595, DeviceID=0x4315, in reality it is Broadcom, which supports chips 4312, 4331, and a number of others. We make a substitution. DSDT patch with mask 0x4000 (FixAirport).

- The common Marvell 80E8056 network card DeviceID=0x4353 simply does not work, but it works with the AppleYukon2 driver if you replace the ID with 0x4363. DSDT patch with mask 0x2000 (FixLan).

- IMEI - this device works with Intel HD3000/4000, however, it is not a fact that your chipset has the correct ID. The substitutions are as follows: SandyBridge = 0x1C3A8086 IvyBridge = 0x1E3A8086 Haswell = 0x8C3A8086 Works with DSDT fix patch AddIMEI_80000 (AddIMEI) This masking works in two cases: during injection or during the DSDT patch. However, if we do not want a full injection as intended by Clover, then we can set the following property:

```
<key>NoDefaultProperties</key>

<true/>
```

In this case, the line for injection is created, but does not yet contain any new properties. For example, such a property would be FakeID. Again, this method of doing FakeID is outdated, it is better to do it through Properties in the following way



You can add other custom properties, such as model, in the following array of dictionaries Deprecated! Use Properties!

```
<key>AddProperties</key>

<array>
	<dict>
		<key>Device</key>
		<string>NVidia</string>
		<key>Key</key>
		<string>AAPL,HasPanel</string>
		<key>Value</key>
		<data>AQAAAA==</data>
	</dict>
...
	</array>
```

The Value can be <data> or a hexadecimal string. You can't just use a string. That is, instead of <string> ABC.... you need to write <string>0x414243.... Convert via PlistEditor or via Xcode. The first Device key determines which device this property will be added to. List of devices:

```
ATI
Nvidia
IntelGFX
LAN
WIFI
Firewire
SATA
IDE
HDA
HDMI
LPC
SmBUS
USB
```

The names should be exactly like this, letter for letter. I think no explanation is needed here. In this way, you can inject different properties for analog sound Device=HDA, and for digital Device=HDMI. Unfortunately, it is not very correct to distinguish Clover by vendor. If Intel, then HDA, if ATI or Nvidia, then HDMI. For example, Hazwell has Intel HDMI sound. This option is not yet provided in Clover. It is envisaged that with Intel graphics, chipset HDA sound will be used on HDMI output. The parameter is used for this

```
<key>UseIntelHDMI</key>

<true/>
```

This parameter affects the injection of sound properties transmitted via HDMI, as well as the DSDT patch. However, as far as I know, the sound drivers of both VoodooHDA and AppleHDA do not fully work with HDMI output. According to new information, VoodooHDA only works with NVIDIA HDMI output, and as for AMD, Apple has created a new driver AppleGFXHDA.kext in 10.13+ systems. Study its capabilities. To distinguish which driver is used for what, add IONameMatch=HDAS to VoodooHDA info.plist, and remove IOPCIClassMatch. Then VoodooHDA will only attach to a device named HDAS, and the system driver will attach to HDMI sound.

<key>HDMIInjection</key>

<false/>

Completely disable injection of HDMI device properties.

Starting with revision 3262, a new method of injecting device properties is introduced, not by name, as it was before, but by their location on the PCI bus. Here is the list that Clover gives in the boot log

```
4:432  0:000  PCI (00|00:00.00) : 8086 2E30 class=060000
4:432  0:000  PCI (00|00:02.00) : 8086 2E32 class=030000
4:432  0:000  Found GFX model=Unknown
4:432  0:000  PCI (00|00:02.01) : 8086 2E33 class=038000
4:432  0:000  PCI (00|00:1B.00) : 8086 27D8 class=040300
4:432  0:000  PCI (00|00:1C.00) : 8086 27D0 class=060400
4:432  0:000  PCI (00|01:00.00) : 10DE 01D1 class=030000
4:432  0:000  Found NVidia model=Gigabyte GeForce 7300 LE
4:432  0:000  PCI (00|00:1D.00) : 8086 27C8 class=0C0300
4:432  0:000  PCI (00|00:1D.01) : 8086 27C9 class=0C0300
4:432  0:000  PCI (00|00:1D.02) : 8086 27CA class=0C0300
4:432  0:000  PCI (00|00:1D.03) : 8086 27CB class=0C0300
4:432  0:000  PCI (00|02:05.00) : 10EC 8167 class=020000
```

Here we see two video cards and four USB UHCI devices. The device type is identified by its class class =040300 - this is an HDA standard sound device. In this case, it is located at the address 00:1B.00, and the network card of class 020000 is located at the address 02:05.00

Bus= 02

Device = 05

Function = 00

Use this value for property injection Deprecated! Use Properties!

<key>Arbitrary</key>

```
<array>
	<dict>
		<key>PciAddr</key>
		<string>02:05.00</string>
		<key>Comment</key>
		<string>Realtek LAN 8167</string>
		<key>CustomProperties</key>
		<array>
			<dict>
				<key>Disabled</key>
				<true/>
				<key>Key</key>
				<string>model</string>
				<key>Value</key>
				<string>Realtek 8169 Gigabit Ethernet
Controller</string>
			</dict>
			<dict>
				<key>Key</key>
				<string>built-in</string>
				<key>Value</key>
				<data>AQAAAA==</data>
			</dict>
		</array>
		</dict>
		<dict>
			<key>PciAddr</key>
			<string>01:00.00</string>
			<key>Comment</key>
```

```
                        <string>Nvidia Geforce card in PCIe slot</string>
                        <key>CustomProperties</key>
                        <array>
                                <dict>
                                        <key>Key</key>
                                        <string>model</string>
                                        <key>Value</key>
                                        <string>Gigabyte GeForce 7300 LE</string>
                                </dict>
                                <dict>
                                        <key>Disabled</key>
                                        <true/>
                                        <key>Key</key>
                                        <string>AAPL,boot-device</string>
                                        <key>Value</key>
                                        <data>AQAAAA==</data>
                                </dict>
                        </array>
                </dict>
        </array>
```

Thus, the Arbitrary section is an array of dictionaries, each of which corresponds to one device with a given address, and to describe each device, an array of CustomProperties consisting of Key-Value pairs is used. Also, a specific property can be disabled with the Disabled key. You can enable or disable a property dynamically in the Clover menu. Key must be a string <string> Value can be a string, number or data <string>, <integer>, <data>

<key>ForceHPET</key>

```
<true/>
```

It turns out that there are still computers where NRET is disabled by default, and there is no checkbox in the BIOS to enable it. This key will help (revision 2789+)

<key>SetIntelBacklight</key>

```
<false/>
```

The key was introduced in revision 3298. In previous systems, the screen brightness was increased by special kexts IntelBacklight or ACPIBacklight, they do not work in Captain, but it turned out to be very easy to do this in Clover at the system startup stage, and no additional kext is needed, just set <true/>. Trick from Ramalama and Rehabman. And additional keys

<key>SetIntelMaxBacklight</key>

```
<true/>
```

The value from the following key will be written into the chip register:

<key>IntelMaxValue</key>

```
<integer>1808</integer>
```

or the default value that is most suitable for this chip. Sherlock entered values for almost all Intel chips into Clover, based on the analysis of dumps from real ones. I don't know if this is always correct, but Clover will offer default values.

<key>DisableFunctions</key>

```
<string>0x18F6</string>
```

This is a mask that is superimposed on RCBA 0x3418 - sets additional bits, prohibiting some devices in the Intel chipset. For very serious hackers.

<key>LANInjection</key>

```
<false/>
```

By default, the built-in property is injected for the network card. This parameter can be used to disable such injection.

RtVariables

The following two parameters were introduced starting with revision 980 and are intended for permissions to register in the iMessage service. Starting with revision 1129, the parameters are taken from SMBIOS and are not needed here.
MLB = BoardSerialNumber
ROM = last digits of SmUUID or Mac address.

<key>Block</key>

```
<array>
    <dict>
        <key>Comment</key>
        <string>Dell variables</string>
        <key>Disabled</key>
        <false/>
        <key>Name</key>
        <string>*</string>
        <key>Guid</key>
        <string>FF2E9FC7-D16F-434A-A24E-C99519B7EB93</string>
    </dict>
</array>
```

This is an opportunity to block the function SetVariable(Name, Guid....), that is, so that external procedures could not do unnecessary things. Specifically, I had a problem that my Dell Latitude constantly sets two variables with some Guid for some of its own purposes. I blocked them.

<key>MLB</key>

```
<string>XXXXXXXXXX</string>
```

Numbers and letters, 17 characters long, indicating the serial number of the motherboard. There is no pattern. The most reliable is to take the real number and change the middle digits, for example, write ...SLICE... Everyone has their own imagination.

<key>ROM</key>

```
<data>AAAAAAAA</data>
```

Six pairs of 16-digit digits, often matching the MAC address of the network card. However, there are reports that the service works with arbitrary numbers. Starting with revision 3051, you can write <string>UseMacAddr0</string>, and Clover will determine the Mac Address of your network card itself. The procedure does not work for everyone, so check. And most importantly, registration in iMessage implies a paid service, you must specify a real bank card, from which you will be debited $ 1. Those who try to log in for free receive messages like "Call Apple".

In 2015, the 10.11 ElCapitan system was released with new security requirements. SIP = System Integrity Protection. By default, the protection is enabled and does not allow you to upload your own kexts and install their system utilities. To disable it, Clover gives the ability to set new parameters in NVRAM

<key>CsrActiveConfig</key>

```
<string>0x3E7</string>
```

```
<key>BooterConfig</key>

<string>0x28</string>
```

These are bit masks with possible bit values.

```
/* Rootless configuration flags */
#define CSR_ALLOW_UNTRUSTED_KEXTS        (1 << 0)
#define CSR_ALLOW_UNRESTRICTED_FS        (1 << 1)
#define CSR_ALLOW_TASK_FOR_PID           (1 << 2)
#define CSR_ALLOW_KERNEL_DEBUGGER        (1 << 3)
#define CSR_ALLOW_APPLE_INTERNAL         (1 << 4)
#define CSR_ALLOW_DESTRUCTIVE_DTRACE     (1 << 5) /* name deprecated */
#define CSR_ALLOW_UNRESTRICTED_DTRACE    (1 << 5)
#define CSR_ALLOW_UNRESTRICTED_NVRAM     (1 << 6)
#define CSR_ALLOW_DEVICE_CONFIGURATION   (1 << 7)
#define CSR_ALLOW_ANY_RECOVERY_OS        (1 << 8)
#define CSR_ALLOW_UNAPPROVED_KEXTS       (1 << 9)
#define CSR_ALLOW_EXECUTABLE_POLICY_OVERRIDE    (1 << 10)
#define CSR_ALLOW_UNAUTHENTICATED_ROOT          (1 << 11)


/* Bitfields for boot_args->flags */
#define kBootArgsFlagRebootOnPanic    (1 << 0)
#define kBootArgsFlagHiDPI            (1 << 1)
#define kBootArgsFlagBlack            (1 << 2)
#define kBootArgsFlagCSRActiveConfig      (1 << 3)
#define kBootArgsFlagCSRPendingConfig     (1 << 4)
#define kBootArgsFlagCSRBoot          (1 << 5)
#define kBootArgsFlagBlackBg          (1 << 6)
#define kBootArgsFlagLoginUI          (1 << 7)
```

The default values correspond to disabling protection. If you need to enable protection due to some paranoia, set the value to 0 (zero). However, some kexts, such as VoodooHDA, will not work with a value of 0. At least a CSR with bit 0x1 is required - CSR_ALLOW_UNTRUSTED_KEXTS In general, I set the mask to 0x285, that is, CSR_ALLOW_TASK_FOR_PID, CSR_ALLOW_DEVICE_CONFIGURATION and CSR_ALLOW_UNAPPROVED_KEXTS, and there is also information that TotalFinder requires the CSR_ALLOW_UNRESTRICTED_FS bit to work. And in Acidantera also recommends CSR_ALLOW_UNAUTHENTICATED_ROOT, so the mask becomes 0xA87. In no case should you set the bit 0x10 = CSR_ALLOW_APPLE_INTERNAL . If you do, availability Updates will not be offered, and the rest is also questionable.

```
<key>HWTarget</key>

<string>J160</string>
```

This is a new variable since Clover 5140+ (commit ec1f8a6). It was required for **Monterey** since some beta, so that the Mac model with the T2 chip would also receive updates. The next observation is which value should correspond to which model (thanks Gradou)

```
MacBookPro 15,1 (J680AP) 15,2 (J132AP) 15,3 (J780AP) & 15,4 (J213AP)
MacBookPro16,1  (J152FAP) 16,3 (J223AP) & 16,4 (J215AP
MacBookPro16,2  (J214KAP)
MacBookAir8,1   (J140KAP) & 8,2 (J140AAP)
MacBookAir9,1   (J230KAP)
Macmini8,1      (J174AP)
iMac20,1        (J185AP) & 20,2 (J185FAP)
iMacPro1,1      (J137AP)
MacPro7,1       (J160AP)
```

For other models, this parameter does not need to be set. In Ventura and higher, this is useless, updates are only available for SMBIOS iMac19,1 or MacBookPro15,1.

The next three parameters are excluded, because they must be set by the system from the Clover Control Panel to avoid conflict.

```
<key>MountEFI</key>
<string>Yes</string>
```

This parameter tells the startup script that the ESP (EFI System Partition) should be mounted when logging into the system. This parameter is unnecessary or temporary for most people, it is worth specifying No in the config, and Yes in the menu if necessary. Another possible value is disk1, if you have several disks, and each has its own partition EFI.

```
<key>LogEveryBoot</key>
<string>Yes</string>
```

The boot log is needed by developers, but ordinary users can set No. Here, instead of Yes, there can be a number of logs to store in the system.

```
<key>LogLineCount</key>
<string>3000</string>
```

The number of lines in this log, then the old lines are replaced by new ones, so that there is no unlimited growth of this file.

It is really not interesting to save these logs. You can always get the latest log with the command

```
$ bdmesg > ~/Desktop/boot-log.txt
```

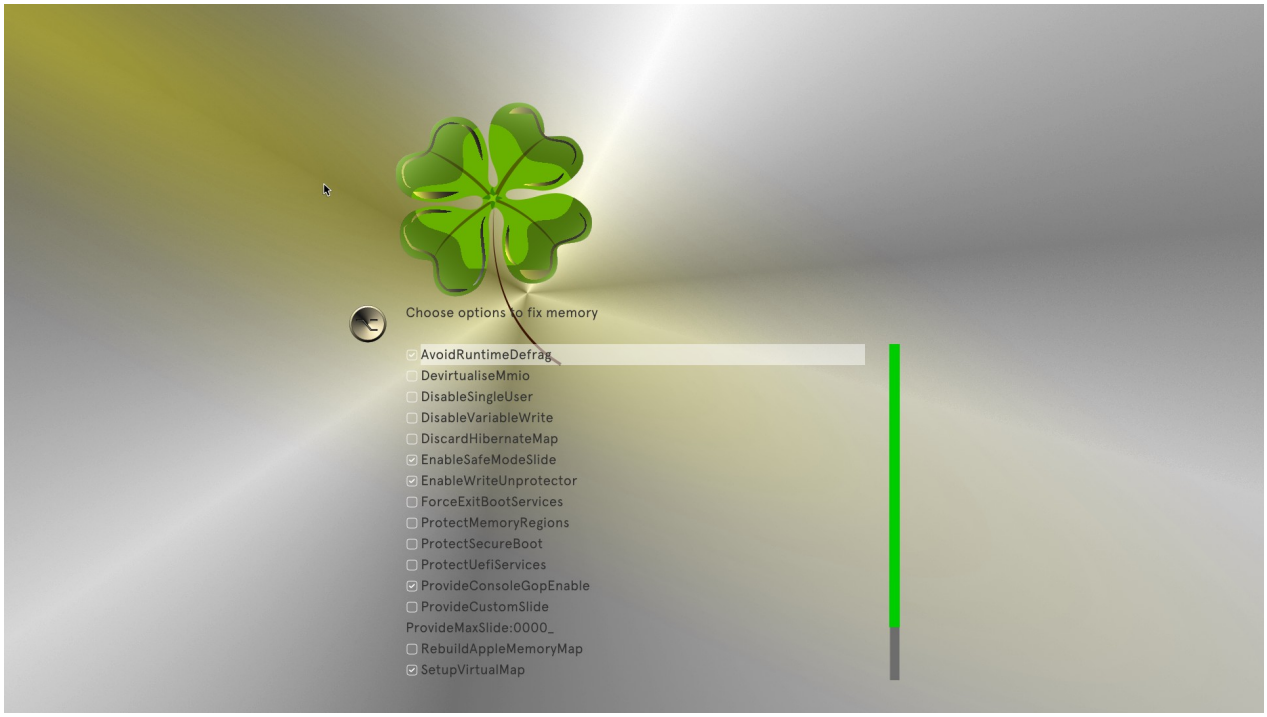
DisableDrivers

```
<key>DisableDrivers</key>
<array>
      <string>CsmVideoDxe</string>
      <string>VBoxExt4</string>
</array>
```

The point of this section is to have different config.plist in different OEM folders, but since the drivers folder is common, we need to somehow distinguish which set of drivers is used on a particular configuration. For example, one needs OsxAptioFixDxe, while another needs EmuVariableDxe. Deprecated! Now all boards use a common set.

Quirks

The history of this section is as follows. The beginning of the development of Clover for UEFI boot by Dmazar was the development of a driver for adjusting the memory that the UEFI BIOS Aptio (American Megatrend) reserves. The fact is that the Allocate function in this BIOS allocates memory in the lower part while to boot macOS you need to have the lower memory free. The conflict affects not only memory there is also boot.efi virtualizing addresses and this affects pointers, functions, and so on. I will not go into detail, this is not my work, it was Dmazar who step by step found all the conflicts and figured out how to resolve them. This became the OsxAptioFixDrv.efi driver. 32-bit and 64-bit versions with all the addressing differences. Note that this problem was not with Legacy Clover, because in this case Allocate from EDK2 is used, and it allocates memory from top to bottom. Legacy Clover works without this driver. For a long time after Dmazar left, no one touched this driver, except, perhaps, for individual one-line additions like Free2000. And then vit9696 took on a major rework of the driver. First of all, he made a change that allowed using native NVRAM on many chipsets (BIOS), with which it did not work before. And then he divided the driver into semantic parts (quirks), which could now be turned on and off at the user's discretion, if the OpenCore bootloader is used. But there was also a programmer ReddestDream, who decided to extract all these developments from OpenCore into a separate driver



OcQuirks.efi, which works together with the driver OpenRuntime.efi, and all the settings are written in the OcQuirks.plist file to use all this with the Clover bootloader.

And now it's my turn. I need to have all the sources in one repo, so that I can do bisection. And since the license for all this allows you to use the sources at your own discretion, and historically everything has returned to its historical homeland, I copied them and modernized them in such a way that instead of a separate .plist file, the same Clover config.plist is used, and these settings can also be changed simply from the Clover GUI. That is, if you can't boot right away, you can try to go into this menu and change some setting yes-no.
Now details about each item. Default values are indicated.

<key>AvoidRuntimeDefrag</key>

```
<true/>
```

Prevents memory defragmentation for runtime services, such as NVRAM support. Recommended for all but Apple and VMware.

<key>DevirtualiseMmio</key>

```
<false/>
```

Removes the runtime attribute from some known MMIO regions. Not recommended for systems older than Sandy Bridge. The list of known regions can still be supplemented in the MmioWhitelist picks section. However, in my observations, it is always disabled. The need for the Z390 chipset is stated.

<key>MmioWhitelist</key>

```
<array/>
```

The list of regions is specified as an array of dictionaries

```
<dict>
	<key>Comment</key>
	<string>Это такой-то регион</string>
	<key>Address</key>
	<string>0xffe00000</string>
	<key>Enabled</key>
	<true/>
```

```
</dict>
```

<key>DisableSingleUser</key>

```
<false/>
```

Disables using command line mode because it is not safe. ;)

<key>DisableVariableWrite</key>

```
<false/>
```

Disables write access to NVRAM from MacOS for security purposes.

<key>DiscardHibernateMap</key>

```
<false/>
```

When waking up from hibernation, use the old memory card. Use only if you are absolutely sure that this is your case. Almost always no.

<key>EnableSafeModeSlide</key>

```
<false/>
```

By default, loading in safe mode gives slide=0. This patch allows you to use a different value, set in the ProvideCustomSlide quirk. I got panic with this quirk on my Skylake, so false is now the default.

<key>ProvideCustomSlide</key>

```
<false/>
```

Whether or not to use automatic calculation of the slide=*** value. The need is visible from the log if there is a message OCABC: Only N/256 slide values are usable!

<key>ProvideMaxSlide</key>

```
<integer>0</integer>
```

A value from 1 to 254, for the case specified above. The value 255 is set by default, and this can lead to memory allocation errors.

<key>EnableWriteUnprotector</key>

```
<true/>
```

Removes the write protection bit in the Runtime services page. Due to the insecurity of this pick, there is another RebuildAppleMemoryMap. That is, for systems older than 2018, we set

```
EnableWriteUnprotector - True
RebuildAppleMemoryMap - False
~~SyncRuntimePermissions — False~~ No! Always True
```

And for newer ones that support MATS, if such an ACPI table is in the BIOS

```
EnableWriteUnprotector - False
RebuildAppleMemoryMap - True
SyncRuntimePermissions - True
```

<key>ForceExitBootServices</key>

```
<false/>
```

Try ExitBootServices again with a new memory map. This is where kernel and kext patches happen. Only use this if you know what you're doing.

<key>ProtectMemoryRegions</key>

```
<true/>
```

Changes attributes of some regions when there is a conflict between the concepts of MacOS and

BIOS. This kirk includes several fixes, including one developed by me personally, which protects the CSM region. In particular, if your video card does not have a UEFI VBIOS, this kirk should be enabled. On the other hand, in the OsxAptioFix3Drv driver it is definitely enabled, and has never caused any complaints.

```
<key>ProtectSecureBoot</key>
<false/>
```

Changes attributes of some regions when there is a conflict between the concepts of MacOS and BIOS. This kirk includes several fixes, including one developed by me personally, which protects the CSM region. In particular, if your video card does not have a UEFI VBIOS, this kirk should be enabled. On the other hand, in the OsxAptioFix3Drv driver it is definitely enabled, and has never caused any complaints.

```
<key>ProtectUefiServices</key>
<false/>
```

Protects encryption keys from being overwritten by the operating system. Secure Boot technology, which we don't really need. The need for this pick is observed on some Insyde BIOSes. The rest don't need it.

```
<key>RebuildAppleMemoryMap</key>
<false/>
```

Generate a memory card compatible with MacOS. The thing is that Apple has slightly different ideas about how and what to do than our UEFI BIOS. The problem, however, is that our memory card matches our hardware. Therefore, we disable the quirk. Other quirks perform the necessary part of this work. This quirk seems to replace EnableWriteUnprotector for systems that support MAT. This quirk also requires SyncRuntimePermissions to be enabled. However, we have it enabled by default. See EnableWriteUnprotector quirk.

```
<key>SetupVirtualMap</key>
<true/>
```

Kirk deals with the order of virtual address assignment and usage. Some BIOSes, such as OVMF, do not support this kirk. Hmm, why do we need this OpenRuntime.efi driver in a system with OVMF?! It works without it, just like legacy Clover. So let's enable it.

```
<key>SignalAppleOS</key>
<false/>
```

Tells BIOS that we are loading the MacOS system, although we are loading Windows. Needed on some MacBooks, but not for us. Maybe needed by OpenCore, but not by Clover.

```
<key>SyncRuntimePermissions</key>
<true/>
```

Updates permission flags in the Runtime area. Of course it should.

The general situation is that for my not-so-new computers these default settings were sufficient. That Rediskin has a different list, and he claims that his list matches thebehavior of AptioMemoryFix. However, this is not true. With his set my computer does not boot, while with the old AptioMemoryFix everything is fine. So in Clover the default set of picks is different from the original OcQuirks (RIP). In Clover 5125, due to the inclusion of patches from OpenCore, new values appeared in this section.

```
<key>FuzzyMatch</key>
<false/>
```
The key provides compatibility with the 10.6 system (Snow Leopard), where the cache checksum is calculated differently. I don't know how it is with OpenCore, but Clover has been loading Snowball for a long time, and together with that same cache.

```
<key>KernelCache</key>
<string>Auto</string>
```
Possible values are (Auto, Cacheless, Mkext, Prelinked). These cache types were pre-10.7 and were usually selected via the Kernel= boot argument, Mkext= and others were present in old Clover.

```
<key>AppleXcpmExtraMsrs</key>
<false/>
```
Support for XCPM on extra CPUs, what the KernelXCPM patch did before.

```
<key>AppleXcpmForceBoost</key>
<false/>
```
This patch writes 0xFF00 to the MSR_IA32_PERF_CONTROL register = 0x199, that is, instead of normal p-states, OpenCore suggests driving the maximum step.

```
<key>DisableIoMapper</key>
<true/>
```
And there is also the option dart=0. And in new systems, Mojave and higher, this technology works, and there is no need to disable it.

```
<key>DisableLinkeditJettison</key>
<true/>
```
Allows Lilu.kext to work faster. It is unknown how. Well, and like keepsyms=1 is no longer needed.

```
<key>DisableRtcChecksum</key>
<false/>
```
The essence of the patch is that RTC in Mac is used differently than in PC, and the checksum will be bad. Simply put, we block its calculation. Clover has an analogue AppleRTC=YES, and we use it.

```
<key>DummyPowerManagement</key>
<false/>
```
Blocks AppleIntelCpuPowerManagement as an alternative to the NullCPUPM kext. I remind you that the kext is blocked mainly if 0xE2 is locked, and there are other patches against this.

```
<key>ExternalDiskIcons</key>
<false/>
```
The long-known patch against yellow icons is designed as a quirk

```
<key>IncreasePciBarSize</key>
<false/>
```
Increases the memory value for the bar in the IOPCIFamily kext from 1 to 4GB. I don't know who came up with this patch and for what cases. OpenCore does not recommend using it at all.

```
<key>PowerTimeoutKernelPanic</key>
<false/>
```
This patch, for Catalina only, prevents kernel panic if some device does not light up for a long time. Why revert it?! Let it panic instead of stupidly hanging. If you ever encounter such a situation, try this patch.

```
<key>ThirdPartyDrives</key>
<false/>
```
A long-known patch, is in the Clover config, Enable Trim on Non-Apple. The essence is in replacing the word Apple with an empty string. There is also a comment that Catalina has internal capabilities to do the same.

```
<key>TscSyncTimeout</key>
<integer>750000</integer>
```
This quirk was needed, in particular, for the Xeon 2650 v2 processor with 8 cores and 16 threads, on a Chinese board. It turns on all the cores, and synchronizes them as they are turned on. Without this quirk, the system does not work. It is also not enough, you still need a kext and ACPI edits, but this quirk is still needed under all other conditions.

```
<key>XhciPortLimit</key>
<false/>
```
It is a known problem that Apple limits the number of ports of the XHCI, USB2+USB3 controller to 15, and despite the fact that new chipsets provide more ports, Apple is in no hurry to increase this limit. Binary patches have existed for a long time, but they depend on the system version. To what extent OpenCore managed to overcome the dependence on the version, I do not know. Practice shows that all uncrashable patches crash. I made Legacy_USB.kext for myself, removed unnecessary USB2 ports, and made room for USB3. This kext worked in 10.13, and it works in 13.0. And yes, with this kext, the port limit does not need to be changed.
Warning! This quirk is considered incorrect and leads to driver errors. Never install it!

```
<key>ResizeAppleGpuBars</key>
<integer>-1</integer>
```
This new kirk only concerns owners of the RX6800 video card in the BIOS which has the item Resizable Pci Bar, which improves the card's performance in Windows, but in Mac, alas, this parameter can lead to a system crash. That's why we enable the parameter in BIOS, and for Mac we disable it with this quirk, value -1. Acidanthera recommends values from 0 to 10, but I don't see for what cases this is, unless I mean that OpenCore also affects Windows.

```
<key>ProvideCurrentCpuInfo</key>
<false/>
```
This is a slightly more complex patch than just FakeCPUID, it is performed at the level of Opencore libraries, and is used for more complex AlderLake processors and newer. But now, thanks to Hnanoto, who has compiled several CPU patches into a single system for very new Intel CPUs of 12-14 generations, and Ryzen Zen3 — Zen5. I can't really imagine what he did there, but everything seems to be safe. New quirk for this

```
<key>AutoModernCPUQuirks</key>
<false/>
```

AutoModernCPUQuirks: Auto-detect and apply quirks for modern CPUs. Supports Intel 12th-14th Gen and AMD Zen3-Zen5 processors.
Requires ENABLE_MODERN_CPU_QUIRKS build flag. Already done in official Clover release.

ACPI

A group of parameters that regulate the correction of various ACPI tables. And it's not just that Mac has its own requirements, but also different versions of the ADSI specification, and the elementary laziness of manufacturers, and simply that the motherboard BIOS does not have information about the installed cards and the CPU (is it hard to determine dynamically? Clover does it!).

Parameters for the FADT table

```
<key>ResetAddress</key>
<string>0x64</string>
<key>ResetValue</key>
<string>0xFE</string>
```

These two parameters serve for one very valuable fix – restart correction. These values should be in the FADT table, but for some reason they are not always there, moreover, the table itself is sometimes shorter than necessary, so much shorter that these values were discarded. By default, the value is already present in the FACP, however, if there is nothing there, then the pair is used0x64/0xFE,what does restart through meanPS2 controller. Practice has shown that this does not work for everyone, another possible pair of values 0x0CF9/0x06, which means restart via PCI bus. This pair is also used on the native, but does not always work on hackintoshes. The difference is clear, hackintoshes also have a PS2 controller, which can prevent restart if it is not reset. Another option 0x92/0x01, I don't know, maybe it will help someone.

```
<key>AutoMerge</key>
<false/>
```

Merge any DSDT and SSDT changes from EFI/CLOVER/ACPI/patched into existing ACPI files.
If set to true, it changes the way ACPI/patched handles files. Instead of appending such files to the end of the XSDT (e.g. treating them as an additional table/SSDT), if the signature, index, and OemTableId match an existing OEM table, it will replace that table. With this feature, as with DSDT, you can patch individual SSDTs (or other tables) by simply placing the patched file in ACPI/patched. No need to mess with DropOem or DropTables. And the original order is preserved. The mapping for SSDTs is naming-based, where the naming convention used by the F4 extractor in the bootloader menu is used to identify the SSDT's position in the XSDT. For example, if your ACPI/origin had SSDT-6-SaSsdt.aml and you wanted to patch it, you could simply patch the file as needed and place it in ACPI/patched. Same thing if you put it in ACPI/patched as SSDT-6.aml. Since some OEM ACPI sets don't use unique text in the OEM table-id field, Clover uses both the OEM table-id and the number that is part of the file name to locate the original in XDST. If you stick to the names provided in ACPI/origin, you'll be fine. Added by Rehabman in revisions 4265-4346.

```
<key>HaltEnabler</key>
<true/>
```
And this is a fix for the problem with shutdown/sleep during UEFI boot. The fix is once, before calling boot.efi, so 100% efficiency is not guaranteed. Nevertheless, it is quite safe, at least on Intel chipsets. For me personally, this is a lifesaver!

```
<key>UseSystemIO</key>
<false/>
```
If set to true, the SSDT section will be used to select in generated _CST tables between:
```
Register (FFixedHW,
Register (SystemIO,
```

```
<key>smartUPS</key>
<false/>
```
Actually, this parameter is intended to register the power profile=3 in the FADT table. The logic is as follows:
PM=1 – desktop, mains powered
PM=2 – notebook, powered by mains or battery
PM=3 – server, powered by SmartUPS, about which MacOSX also knows something. Clover will make a choice between 1 and 2 based on the analysis of the mobility bit, but there is also a parameter **Mobile** in the section SMBIOS. For example, we can say that we have a MacMini, and that it is mobile. The value 3 will be substituted if smartUPS=Yes. With an incorrect power profile, sleep and shutdown may not work.

MADT (APIC) Adjustment

```
<key>PatchAPIC</key>
<false/>
```
On some computers, you can boot the system only with cpus=1, or with a special patched kernel (Lapic NMI patch). A simple analysis showed that they have an incorrect MADT table, namely, it lacks NMI sections. This parameter is used to correct such tables on the fly. Nothing bad will happen to a healthy computer. However, I have not seen any reports that it helped anyone with something. Yes, there is a developer Florin9doi who needs this, and who corrected this patch in new versions of Clover. There is a corresponding patch in the KernelAndKextPatches section, also to solve this problem, but by other means. This patch is a real advantage of Clover over OpenCore, where there is nothing like this, except for somehow manually editing the table. And it affected the fact that X299 users remained on Clover.

Other ACPI tables:

```
<key>DropTables</key>
<array>
	<dict>
		<key>Signature</key>
		<string>DMAR</string>
	</dict>
	<dict>
		<key>Signature</key>
		<string>MCFG</string>
	</dict>
	<dict>
```

```
                <key>Signature</key>
                <string>SSDT</string>
                <key>TableId</key>
                <string>CpuPm</string>
                <key>Length</key>
                <string>0x0fe1</string>
            </dict>
        </array>
```

In this array we list the tables we want to drop.
DMAR — because Mac is not friendly with VT-d technology. Or rather, there is a different table. See below about the DMAR table patch, so as not to discard it.
MCFG — because by specifying the model MacBookPro or MacMini we get severe brakes. A more correct method has already been invented.

<key>FixMCFG</key>

```
<true/>
```

In this case, the table is not discarded, but adjusted. The author of the patch is again vit9696. Nevertheless, the method of discarding this table remains in reserve for now. Back to the story about DropTables SSDTs come in different types, and we additionally specify TableIds that we will discard, because we are going to generate our own SSDT tables, built according to Apple rules, not Gigabyte, or, God forgive me, ASUS. You can look in the table header, or in Clover's boot log. Here, for example, is a table that should not be discarded.

```
DefinitionBlock("SSDT-0.aml", "SSDT", 1, "SataRe","SataTable", 0x00001000)
```

In this case, the rule for binary DSDT patches will be applied to the saved tables, that is, these tables will also be modified, which is logical.
If all SSDT tables for some reason have the same TableID, then you can specify the length of the table you want to drop. The length can be specified in hex, as above, or in <integer> as a decimal number.

<key>DisableASPM</key>

```
<false/>
```

This affects the settings of the ACPI system itself, such as the fact that Apple's ASPM control does not work as intended for us. For example, a non-native chipset. In what cases it should be used, and what it affects, I do not remember.

<key>SSDT</key>

<key>DropOem</key>

```
<true/>
```

Since we are going to create or load our SSDT tables dynamically, we need to avoid unnecessary overlapping of interests. This parameter allows us to discardAll native tables in favor of new ones. Or you categorically want to avoid patching SSDT tables. You have this option: put native tables with minor edits in the `EFI/OEM/xxx/ACPI/patched/` folder, and drop the unpatched ones. (ugh!) discard the uncorrected tables. It is better, however, to use the selective Drop method described above.

<key>Generate</key>

```
<dict>
        <key>CStates</key>
        <true/>
```

```
            <key>PStates</key>
            <true/>
        </dict>
```
Here we define that two additional tables will be generated for C-states and for Pstates, according to rules developed by the hack community. For C-states, table taking into account the parameters C2, C4, C6, Latency, specified by the keys
```
            <key>EnableC7</key>
            <true/>
            <key>EnableC6</key>
            <true/>
            <key>EnableC4</key>
            <false/>
            <key>EnableC2</key>
            <false/>
            <key>C3Latency</key>
            <integer>67</integer>
```
The fact that this generation has taken effect is controlled by the kernel log. Without this method, there is an error `ACPI_SMC_PlatformPlugin::pushCPU_CSTData - _CST` evaluation failed. A separate word about C3Latency. This value appears in real Macs, for iMac it is about 200, for MacPro it is about 10. In my opinion, iMacs are regulated by P-states, MacPro — C-states. And it also depends on the chipset, whether your chipset will adequately respond to C- state commands from MacOS. The easiest option is not to write this parameter, everything will work anyway.
For P-states, the table supplements the processor section with the _PPC, _PCT and _PSS methods.
_PCT - Performance Control – control over speedstep management.
_PPC – Performance Present Capabilities – speedstep capabilities, This function returns a single number, which represents the frequency limit. Details below, in the parameter PLimitDict.
_PSS – Performance Supported States – a set of possible processor states – P-states. This array is formed based on the processor data that Clover has already calculated, as well as taking into account user parameters:

<key>PLimitDict</key>

```
<string>1</string>
```
The essence of the parameter is very simple - to limit the maximum frequency of the processor. The value 0 - work to the maximum, 1 - one step less than the maximum, 2 - two steps. Example: Core2Duo T8300 2400MHz works at a maximum frequency of 2000, if limited by two steps. Why? Yes, so that the laptop does not overheat, the capabilities of the CPU there are much greater than the cooling capabilities. Exactly the same parameter is present in platform plates, for example:
`System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns/ACPI_SMC_PlatformPlugin.kext/Contents/Resources/MacBook5_1.plist`
We will discuss these slabs further below.
For some processors, such as Core2Quad, it has been noted that PlimitDict works the other way around, and the best option is =1. It is quite possible that this is simply a bug in DSDT. For example, because they did not want to make a Darwin patch

<key>UnderVoltStep</key>

```
<string>1</string>
```
An additional parameter for reducing the processor temperature by reducing its operating voltage. Possible values are 0, 1, 2, 3 ... the more, the more we cool until the computer hangs. In this place, protection against fools works, Clover will not allow you to set a value outside the permissible range, or rather write whatever you want, and only what is allowed will work. However, even allowed values can give unstable operation. The effect of this parameter is really observed. However, only for Penryn.

<key>NoDynamicExtract</key>

```
<false/>
```

If set to true, this flag will disable ejection of dynamic SSDTs when using F4 in the bootloader menu. Dynamic SSDTs are rarely needed and usually cause confusion (wrongly put them in ACPI/patched). Added by Rehabman in revision 4359.

<key>NoOemTableId</key>

```
<false/>
```

If set to true, the OEM table ID is NOT appended to the end of the file name in the ACPI tables dump when pressing F4 in Clover in ACPI/origin. If set to false, removes trailing spaces from SSDT names when the OEM table ID is added as a suffix. Added by Rehabman in revisions 4265-4346.

<key>DoubleFirstState</key>

```
<false/>
```

It was found that for successful speedstep it is necessary to duplicate the first state in the P-states table. After introducing other parameters, the necessity of this became questionable. Also, this is only true for IvyBridge, for the rest it is absolutely canceled.

<key>MinMultiplier</key>

```
<integer>7</integer>
```

Minimum processor multiplier. It reports itself as 16, and prefers to work at 1600, however, for speedstep you should set the states in the table down to 800 or even 700. Empirical.

<key>MaxMultiplier</key>

```
<integer>30</integer>
```

Introduced by analogy with the minimum, but it seems to be in vain. It is not worth entering. However, it somehow affects the number of P-states, so you can experiment, however, you should not do this without special need.

<key>Generate</key>

```
<dict>
	<key>CStates</key>
	<true/>
	<key>PluginType</key>
	<false/>
	<key>APLF</key>
	<false/>
	<key>APSN</key>
	<false/>
	<key>PStates</key>
	<true/>
</dict>
```

In the new Clover, this group of parameters is combined into one section, and PluginType is now just true or false. Because there are no other options. The APLF and APSN parameters seem to affect the speedstep, but for those who know what they are for. Note that since APSN/APLF are part of Generate→PStates, they act if Generate→PStates=true, while PluginType is independent and acts regardless of the choice of Generate→PStates.

<key>PluginType</key>

```
<integer>0</integer>
```

For IvyBridge, Haswell (and higher?) processors you need to set 1, for the rest 0.

This key, together with the Generate→PluginType key, allows generating an SSDT table containing only PluginType, but not P-States if their generation is disabled. This key is not needed, it was kept for backward compatibility.

Large section on DSDT setup and patches.

```
<key>DSDT</key>
<dict>
```

```
<key>Debug</key>
<false/>
```

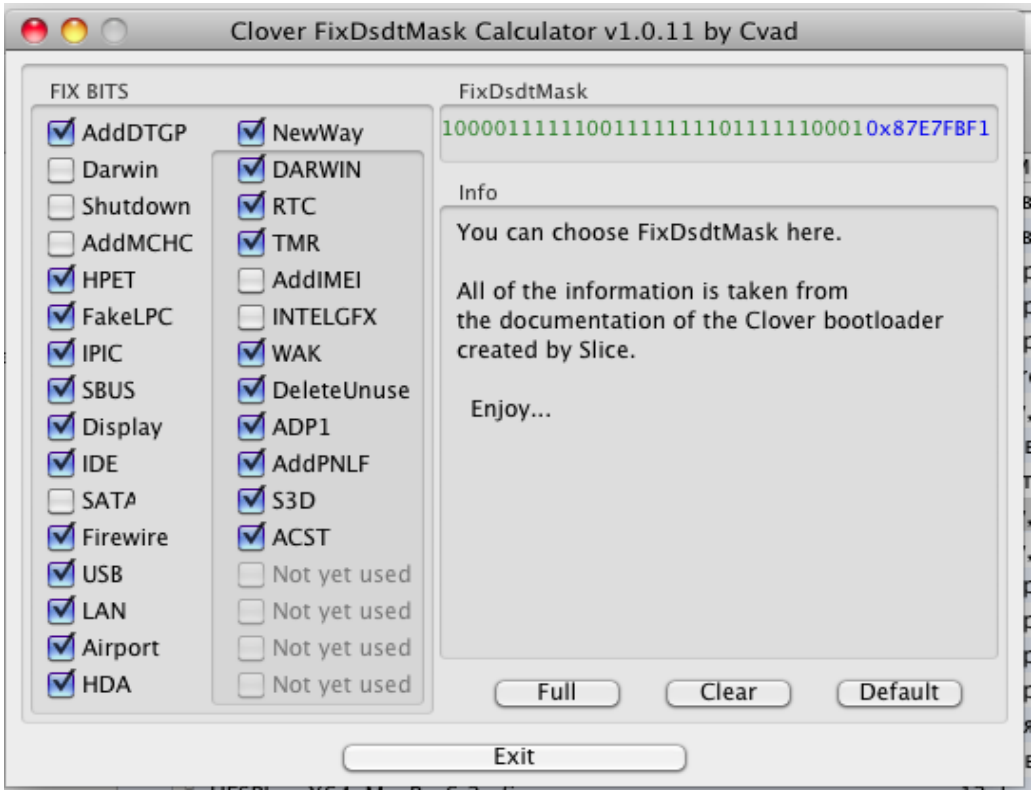
this parameter allows us to see what happens to DSDT during its patching if we cannot boot the system after that. First, the original version /EFI/CLOVER/ACPI/origin/DSDT-or.aml is saved, then the procedure of all patches is performed (by the way, it leaves a lot of messages in debug.log, if it is also connected), and then the file /EFI/CLOVER/ACPI/origin/DSDT-pa0.aml is saved, if such a file already exists from the previous attempt, then the next one by number DSDT-pa1.aml will be created, DSDT-pa2.aml..., they will not overwrite each other. Don't forget to clean the folder at the end of all exercises.

```
<key>Name</key>
<string>DSDT-5.aml</string>
```

You can have several versions of the DSDT file with arbitrary names, for example DSDT-5.aml, or you can write a non-existent name, for example the standard BIOS.aml, and then the DSDT that is in the BIOS will be taken as a basis. The file priorities are as follows:
1. The highest priority is the DSDT.aml file, which is located in the root of the bootable system. The logic is to boot different computers with one flash drive, each of which has its own DSDT.
2. If there is no such file, look for it in the OEM section on the flash drive:
`/EFI/CLOVER/OEM/p8b/ACPI/patched/DSDT.aml`
3. If it is not there, then look in the common folder
`/EFI/CLOVER/ACPI/patched/DSDT.aml`

```
<key>FixMask</key>
<string>0xFFFFFFFF</string>
```

Под этим параметром скрывается сразу 32 патчей для таблицы DSDT, по числу битов в маске.
This parameter hides 32 patches for the DSDT table, according to the number of bits in the mask. To calculate how the sum of bits adds up to a particular mask, you can call the system calculator, convert it to the programmer's view, and switch to 16-bit numbers. And now by clicking the mouse on bits from 0 to 31 we will type the desired mask. There is a more visual option: CloverFixDsdtMaskCalculator by Cvad
http://www.applelife.ru/attachments/cloverfixdsdtmaskcalculator-app-zip.43973/



Starting with revision 2184, patches can (and should) be added bit by bit in the following section

<key>Fixes</key>

```
<dict>
    <key>AddDTGP</key>
    <true/>
    <key>FixDarwin</key>
    <true/>
    <key>FixShutdown</key>
    <true/>
    <key>AddMCHC</key>
    <false/>
    <key>FixHPET</key>
    <true/>
    <key>FakeLPC</key>
    <false/>
    <key>FixIPIC</key>
    <true/>
    <key>FixSBUS</key>
    <true/>
    <key>FixDisplay</key>
    <true/>
    <key>FixIDE</key>
    <false/>
    <key>FixSATA</key>
    <false/>
    <key>FixFirewire</key>
    <true/>
    <key>FixUSB</key>
    <true/>
    <key>FixLAN</key>
    <true/>
    <key>FixAirport</key>
```

```
            <true/>
            <key>FixHDA</key>
            <true/>
            <key>FixDarwin7</key>
            <true/>
            <key>FixRTC</key>
            <true/>
            <key>FixTMR</key>
            <true/>
            <key>AddIMEI</key>
            <true/>
            <key>FixIntelGFX</key>
            <false/>
            <key>FixWAK</key>
            <true/>
            <key>DeleteUnused</key>
            <true/>
            <key>FixADP1</key>
            <true/>
            <key>AddPNLF</key>
            <true/>
            <key>FixS3D</key>
            <true/>
            <key>FixACST</key>
            <true/>
            <key>AddHDMI</key>
            <true/>
            <key>FixRegions</key>
            <true/>
        </dict>
```

If this fix section is present, the fix key by mask will be ignored.
But in order to tell what these fixes mean, we will have to open a new chapter.

More keys to help solve some problems with automatic patching.

```
<key>ReuseFFFF</key>
  <false/>
```

In some cases, an attempt to make a display patch is limited by the presence of a device of the type the original DSDT

```
Device (PEGP)
{
    Name (_ADR, 0xFFFF)
    Name (_SUN, One)
}
```

You can change the address to 0, but it doesn't always work.
<true/> - we try to change the address,
<false/> - we leave and don't try to patch it.
This fix is excluded from the new Clover starting with 5116.

```
<key>DropOEM_DSM</key>
 <dict>
        <key>ATI</key>
        <true/>
        <key>NVidia</key>
        <true/>
        <key>IntelGFX</key>
```

```
            <true/>
            <key>HDA</key>
            <true/>
            <key>HDMI</key>
            <true/>
            <key>LAN</key>
            <true/>
            <key>WIFI</key>
            <true/>
            <key>SATA</key>
            <true/>
            <key>IDE</key>
            <true/>
            <key>USB</key>
            <true/>
            <key>LPC</key>
            <false/>
            <key>SmBUS</key>
            <false/>
            <key>Firewire</key>
            <true/>
        </dict>
```

In some cases, the device we want to automatically patch already has an OEM _DSM method. It cannot be duplicated, so there are two options:
<true/> - the original method will be discarded, and ours will be generated instead,
<false/> - having encountered the original method, we retreat without doing anything.
What's in the original method? It's hardly what we'd like to see, and it's hardly what OSX needs. BIOS manufacturers usually only think about Windows. If you think that there is something important in that method, then inject your properties into this device using strings (see the chapter Devices->Inject). So I would recommend dropping all these OEM DSMs, except for the case when you put your custom DSDT, and apply other fixes to it from among the automatic ones, but you do not want to replace your _DSM methods with automatically generated ones.

In my new computer, DSDT turned out to be insanely bad from the macOS point of view. It has many DSM methods, both in the parent device and in the daughters. The result is panic. DropOEM_DSM is no longer able to help here. The following patch helped
Patches → find :_DSM, replace: ZDSM. See below.

<key>PNLF_UID</key>

```
<string>0x0A<string/>
```

There are several samples of curves/graphs of brightness in the system, they have different UIDs, but if some real user had such a curve, it does not mean that you will have the same brightness with the same processor. It depends on the panel, not on the processor. It would be better to develop a PNLF calibration system, but this is a high-level exercise. For now, we just suggest experimenting with different values, maybe it will be better. Added in revision 5103.

<key>SuspendOverride</key>

```
<false/>
```

The shutdown patch only works on state type 5 - shutdown, however, we may want to extend this patch to states 3 and 4, set SuspendOverride = true. This helped me with entering sleep mode during UEFI boot. Otherwise, the screen would go dark, but the lights and fans would continue to run.

More advanced hackers can make their own DSDT patches using binary-level replacement:

<key>Patches</key>

```
<array>
    <dict>
          <key>Find</key>
          <data>UFhTWAhfQURSAAhfUFJXEgYC</data>
          <key>Replace</key>
          <data>UFhTWAhfQURSAAhfU1VOCgQIX1BSVxIGAg==</data>
          <key>TgtBridge</key>
          <data>UFhTW</data>
    </dict>
    <dict>
          <key>Comment</key>
          <string>Rename oem _DSM to ZDSM</string>
          <key>Disabled</key>
          <false/>
          <key>Find</key>
          <data>
                X0RTTQ==
          </data>
          <key>Replace</key>
          <data>
                WkRTTQ==
          </data>
            <key>Skip</key>
            <integer>0</integer>
            <key>Count</key>
            <integer>0</integer>
    </dict>
</array>
```

TgtBridge specifies that the patch will only work within one block defined by this name
Device (RP02) { .. }
However, this method is not accurate because the blocks are fragmented, it is better to use the RenameDevices method (below) The **Skip** and **Count** keys were introduced starting with revision 5149. The meaning seems obvious. If Skip = 5, then skip the first five matches. If Count = 3, then apply the replacement only three times. The specific numbers are yours, from your developments, if you know what to do. The lengths of the strings may not match, Clover will correctly account for the change in length, with one exception: so that it does not happen inside the If or Else operator. If you need such a change, replace the If-Then-Else operator entirely. Some explanations. Key **Comment** serves not only to remind you that here it is written, it is also used in the Clover menu to create a menu for connecting/disabling these fixes. The initial value is enabled or not is determined by the lines **Disabled**=false. The default value is allowed. If you have someone else's patch set, it is better to first register them as Disabled=true,a nd then in their menu resolve one at a time.

Other ACPI tables

Clover has a service for loading other tables. In particular, it is very common to create a whole library of different SSDT-xxx.aml for different devices and for speedstep. Starting with revision 3088, the rule for loading such tables has changed.
Place the tables in a folder
`/EFI/CLOVER/OEM/xxx/ACPI/patched/`
if there is no such folder, then the common folder is considered
`/EFI/CLOVER/ACPI/patched/`
All files with the extension ".aml" that do not start with a dot "." and do not contain the string "DSDT" in their name will be loaded from this folder, because DSDT of one of the different

variants is loaded using a different algorithm. The boot order is not guaranteed. If we want a strictly defined boot order, we must write it explicitly in the config.

```
<key>ACPI</key>
<dict>
      <key>SortedOrder</key>
      <array>
            <string>SSDT-3.aml</string>
            <string>SSDT-1.aml</string>
            <string>SSDT-2.aml</string>
      </array>
```

And if such an array is present, only these tables will be loaded, and strictly in this order.
Another problem with tables is the name. OEM does not shy away from using either the national alphabet or just the absence of a name, but for Apple this is unacceptable. The name must be 4 characters of the Latin alphabet. The next fix will fix this

<key>FixHeaders</key>

```
<true/>
```

Exactly the same fix was in the DSDT Fixes section, however, since it does not relate to DSDT, it was moved to a separate item in the ACPI section.

<key>RenameDevices</key>

```
<key>RenameDevices</key>
<array>
        <dict>
      <key>_SB.PCI0.RP05.PXSX</key>
      <string>UPSB</string>
        </dict>
        <dict>
      <key>_SB.PCI0.RP05.UPSB.DSB1.NHI0</key>
      <string>UPS0</string>
   </dict>
      <dict>
      <key>_SB.PCI0.RP03.PXSX</key>
      <string>BRG3</string>
       </dict>
</array>
```

Unlike binary ACPI patches, which, by the way, work not only on DSDT, but also on SSDT, both native and downloadable, this method serves as a replacement for a patch like Find->PXSX, Replace->ARPT. But if in the DSDT->Patches section such a replacement works on the entire space, then in the RenameDevices method the algorithm will search only for those devices that are on the specified bridge.

Here is a complex example for replacement `_SB.PCI0.RP02.PSXS → ARPT`

```
Scope(\_SB)
{
  Device (PCI0)
  {
    Device(RP02)
    {
      Device(PSXS)  <- change here
      {
            Method(_ON)
            {
            }
            Method(_OFF)
            {
            }
```

```
    }
    PSXS._ON() <- change here
  }
  Scope(RP02)
  {
     PSXS._OFF() <- change here
  }
  Device(RP03)
  {
     Device(PSXS) <- don't change here
     {
     }
     PSXS._ON() <- don't change here
  }
```

Before version 5130 it was just a dictionary, but now this section is made as an array of dictionaries. The idea is that if one patch depends on another, they must be executed in strict order, and the dictionary does not guarantee the order of application. The order is guaranteed by the array, so I recommend rewriting the existing config to the new syntax, see config-sample.plist.

Editing DMAR and using VT-d

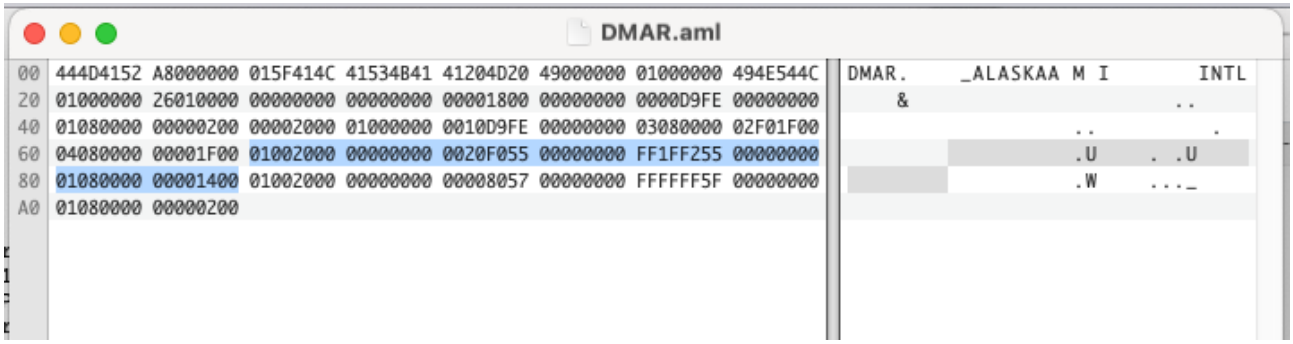
Clover can't do this automatically yet, but I'll describe it here, since it's related to ACPI tables. That is, we set DROP→DMAR, but in the patched folder we place another DMAR-new table, which we made ourselves from the original.
The problem arose from my story. Before Mojave, I always disabled VT-d in the BIOS, because MacOS immediately panicked. From Mojave to Sonoma 14.1, you could live with VT-d enabled, and not even drop the DMAR table. And then there were messages that some network drivers only work if it is enabled. And suddenly I couldn't update to 14.2. The system hung, and without panic, it just got stuck halfway. After a series of experiments, it was found that VT-d should be disabled. Or? Edit the DMAR table, as other hackintoshers have already done. What's in it? There was a timid assumption that all reserve regions should be deleted, however, real Macs have regions in their tables. We move on and find that each region corresponds to a PCI device. Example:

```
[068h 0104   2]                Subtable Type : 0001 [Reserved Memory Region]
[06Ah 0106   2]                       Length : 0020

[06Ch 0108   2]                     Reserved : 0000
[06Eh 0110   2]          PCI Segment Number : 0000
[070h 0112   8]                Base Address : 0000000055F02000
[078h 0120   8]          End Address (limit) : 0000000055F21FFF

[080h 0128   1]            Device Scope Type : 01 [PCI Endpoint Device]
[081h 0129   1]                Entry Length : 08
[082h 0130   2]                     Reserved : 0000
[084h 0132   1]              Enumeration ID : 00
[085h 0133   1]              PCI Bus Number : 00

[086h 0134   2]                        PCI Path : 14,00
```

This PCI Path corresponds to the XHCI controller on my chipset, which can be seen from the Clover preboot log

```
0:137 0:000 PCI (00|00:14.00) : 8086 A12F class=0C0330
```

Device class 0C0330 corresponds to USB3.0. There is no such reserve region in real Macs. But why does it interfere now? I don't know. Just this piece should be completely cut out of the DMAR table, and we are not afraid, macOS will correctly step over the missing piece, you just need to correct the total length of the table in the fifth byte.



So cut since 68 upto 88 bytes (hex numbers), and in A8 change to 88 — table length before and after. The execution is simple, it is performed by the HexFiend editor, there are analogs, and in Windows. There is also an option to edit the DSL, and then compile. There is also a second reserve region

```
PCI Path: 02.00
```

No need to cut it. And now my computer installed 14.2 and then 14.3 with VT-d enabled and everything works.

DSDT Adjustment

DSDT - Differentiated System Description Table - the largest and most complex ADSPI table. The minimum length is 36 bytes, the real length is from 20 kb and even megabytes. This table describes devices and methods of access to them. Access methods can contain arithmetic and logical expressions, and thus represent a program in some programming language, similar to C with its curly brackets. Correcting this table means understanding something in programming. Clover offers a version of automatic correction, but it is necessary to understand that artificial intelligence has not yet been created, and automatic correction of programs is still far from perfect. A person will do better.

Why does it need to be fixed? The entire DSDT patch, since its inception, has been aimed primarily at fixing the HPET device – High Precision Events Timer. The fact is that the OSX system has the AppleIntelCPUPowerManagement kext, which serves to manage the processor power supply (speedstep), and which strictly requires that the system have an HPET with IRQ interrupts. Without this condition, the kext goes into panic. You can work only by disabling or deleting this kext. But there is another option – correct DSDT, and the HPET device will turn on as expected! However, in the 10.9 system the requirements have changed. Nonsense! The patch is still needed, and has the same effect. In new systems, on new hardware (IvyBridge+) it has already lost its relevance, primarily because the kext functionality has gone into the kernel.

This is a **patch №1,** a vital necessity. Does MacOS alone need this NRET? No, of course not, but BIOS manufacturers are just beginning to realize this and write the correct parameters, and it is still rare to find DSDT working without patches.

Moment №2. In DSDT you can see some dependencies on the operating system, "Windows 98", "Windows 2001", "Windows 2006", "Linux", MacOS has the identifier "Darwin", and, as a rule, DSDT is not designed for it. And even if it is designed, then for a version like FreeBSD. MacOSX is a serious ACPI system, i.e. it uses DSDT to the maximum, as it is used by Windows 2001, but not Linux, not Windows 98, and not Windows 2006. The most correct thing is to make mimicry for Windows 2001. And even if you already have "Darwin", make it work like "Windows 2001".On In many BIOSes this corresponds to the value OSYS =0x07D2. But not 0x2410, as it is written in the native. Although there are reports that 7D6 or 7D9 are better for some configurations. You need to look at the algorithm. On my Skylake 2017, the best value was 0x7D9. That's why an additional fix for DSDT FixDarwin7 was made, which mimics Windows 2009, that is, Windows 7. For me, it

corresponds to 0x7D9, and this affects the operation of USB3.0, for some reason they decided to disable it in the BIOS for the Windows XP system. Again, the patch may not be needed if this identification is not used further in DSDT. Have you checked this?

Moment №3. The manufacturer of the motherboard, and with it its BIOS, and its DSDT, cannot foresee which processor will be installed, which video card and other PCI devices. But they should be registered in the DSDT! And vice versa, exclude from the DSDT such devices as the speaker, floppy disk drive, parallel port. There are no drivers for them and they are not needed. It is also often necessary to add or subtract connectors/ports for some devices, for example, for a video card, or for a SATA controller.

DSDT is in the BIOS and is used in the system in the binary code of AML, there is a compiler/decompiler IASL, which translates the codes into a human-readable DSL language. The human editing path is AML->DSL->edit->DSL->AML. And here the moment arises №4. The last compilation becomes impossible due to errors, syntax and logical ones, initially present in OEM DSDT. In the process of editing, they also need to be corrected. Well, and at the same time, semantic errors need to be corrected, because of which, for example, the computer cannot fall asleep, or cannot wake up. And maybe even register new devices. (In general, it is strange, but compilation/decompilation are not strictly inverse operations, they change the table back and forth, or even go there, but not back - intervention is required. From my point of view, this means that the decompiler is written with errors, such are the programmers who worked on it. And non-compliance with standards should be marked as warnings, not errors. If something wrong is happening in the original AML, then you need to understand that the computer is working, which means that this wrong can somehow be interpreted).

Once we have done all this, we can feed the bootloader our patched DSDT by placing it in the /EFI/CLOVER/OEM/xxx/ACPI/patched folder, or, if the OEM name of the computer is not yet known, in the /EFI/CLOVER/ACPI/patched folder, or the bootable system itself has its own version of DSDT, located at the root of the system disk.

Where can I get the original DSDT that needs to be patched? There are options to get it using Windows, Linux or even OSX. If Clover somehow managed to run, then now it itself provides such an opportunity. You need to enter the graphical menu and press the keyF4. If Clover is installed on a FAT32 partition, it will be able to save all OEM ACPI tables, including untouched DSDT and FADT to the EFI/CLOVER/ACPI/origin/ folder. Be patient, if saving is done to a flash drive and there are a lot of tables, the process may take a noticeable time. In the current revision, Clover with this method extracts a set of tables that were previously inaccessible by other methods, including AIDA64. There is also a way to save a pre-patched DSDT version to disk. To do this, in the Clover interface, go to the Options Menu, change the DSDT mask, then exit the menu and pressF5. Clover will save your DSDT, adjusted for the current mask, with a name like DSDT-597.aml, i.e. patched with mask 0xF597. You can make several options to compare later.

Now you can take a DSDT file and edit it... Well, for those who are not strong in the ASL language, Clover offers to make some fixes automatically. I will immediately answer this question: "Why does DSDT still compile with errors after editing by Clover?" Yes, DSDT is a set of descriptions and methods, many of which we do not need. Clover does not touch them, even if we set the maximum mask. Errors can be embedded in those places, untouched, and they remain there. But this does not prevent everything else from working, because DSDT does not work as a single whole, but simply as a set of descriptions and methods.

Let's take a closer look at the fixes. In revision 4300+, the description of fixes has been simplified.

AddDTGP bit(0):

To describe the device properties, in addition to the DeviceProperties option discussed above, there is an option with the _DSM method, prescribed in DSDT. _DSM-Device Specific

Method – a well-known template of this method, which works in MacOSX starting in version 10.5, this method contains an array with a description of the device and a call to a universal method DTGP, which is the same for all devices. This fix simply adds this method so that it can then be used for other fixes. **It has no independent meaning. I saw advice to set the mask 0x31, saying that other fixes are not needed. But then (1) is not needed either!**

FixDarwin bit(1):

Mimicry of Darwin OS under Windows XP. Many problems with sleep and brightness grow from incorrect identification of the system.

FixDarwin7 bit(16)

Similarly, only mimicry for the Windows 7 system. Old DSDTs may not have a check for such a system. You have options.

FixShutdown bit(2):

A condition is added to the _PTS function: if the argument = 5 (shutdown), then no other actions need to be taken. Strange, why? However, there are multiple confirmations of the effectiveness of this patch for ASUS boards, maybe for others as well. Some DSDTs already have such a check, in this case such a fix should be disabled. If the config has SuspendOverride=true, then this fix will be extended to arguments 3 and 4. That is, going to sleep (Suspend). On the other hand, if HaltEnabler=true is set, then this patch is probably no longer needed.

AddMCHC bit(3):

Such a device of class 0x060000 is usually absent in DSDT, but for some chipsets this device is serviceable, and therefore it must be registered in order to correctly distribute the PCI bus power management. The question of the need for a patch is decided experimentally. Another experience, this device was needed on my mother with the Z77 chipset, otherwise the kernel panics at the initial stage of startup. And vice versa, on the G41M (ICH7) chipset this fix causes panic. Unfortunately, there is no general rule.

FixHPET bit(4):

As already said, this is the main fix, necessary.
Thus, the minimum required mask of the DSDT patch looks like 0x0010

FakeLPC bit(5):

Replaces the DeviceID of the LPC controller so that the AppleLPC kext can attach to it. It is needed for those cases when the chipset is not provided for OSX (for example, ICH9). However, the native list of Intel and NForce chipsets is so large that the need for such a patch is very rare. The system checks whether the AppleLPC kext is loaded, if not, the patch is needed. Although, this is not a fact either. It happens that the kext itself is unloaded from memory as unnecessary, although the chipset is supported. There is an option to inject the device-id through Properties instead of this patch.

FixIPIC bit(6):

Removes interrupt from IPIC device. This fix affects the operation of the Power key (pop-up window with options Reset, Sleep, Shutdown).

FixSBUS bit(7):

Adds SMBusController to the device tree, thus removing the warning about its absence from the system log. And also creates the correct bus power management wiring, this also affects sleep.

FixDisplay bit(8):

Produces a number of patches for the video card. Injects properties, and the devices themselves, if they are not there. Adds custom properties. If the FakeID parameter is specified, it will be injected via the _DSM method. (revision 1974+)
Also injects the "hda-gfx" property to connect the HDMI output of the video card to the built-in sound when using the AppleHDA driver. It is outdated on new computers, but does not interfere. This is not necessary with VoodooHDA. Patches for all video cards, only for non-Intel. For integrated Intel another fix: FixIntelGfx

FixIDE bit(9):

In the 10.6.1 system, a panic appeared on the AppleIntelPIIXATA kext. Two options for solving the problem are to use a corrected kext or to correct the device in the DSDT. And for more modern systems? Let it be, if there is such a controller.

FixSATA bit(10):

Fixes some problems with SATA, and removes the yellowness of the disk icons in the system by mimicry under ICH6. In general, a controversial method, but without this fix, my DVDs do not play, and the DVD drive should not be removable. That is, simply replacing the icon is not an option! There is an alternative, solved by adding a fix with the AppleAHCIport.kext kext. See the chapter on patching kexts. And, accordingly, this bit can be left unset! One of the few bits that I recommend not to set.

FixFirewire bit(11):

Adds the "fwhub" property to the Firewire controller if it exists. If not, nothing will happen. You can set it if you don't know whether you need it or not.

FixUSB bit(12):

Attempts to solve numerous problems with USB. For the XHCI controller, when using the native or patched IOUSBFamily, such a DSDT patch is irreplaceable. The Apple driver specifically uses ACPI, and the DSDT prescription must be correct. The prescription in DSDT does not conflict with strings.

FixLAN bit(13):

Injecting the "built-in" property for the network card is necessary for proper operation. The card model is also injected for cosmetics.

FixAirport bit(14):

Similar to LAN, in addition, the device itself is created, if it is not yet registered in DSDT. For some known models, the DeviceID is replaced with a supported one. And the airport is enabled without other patches. In old systems.

FixHDA bit(15):

Correction of sound card description in DSDT so that native AppleHDA driver works. Renaming AZAL -> HDEF is performed, layout-id and PinConfiguration are injected.

FixMutex

This patch finds all Mutex objects and changes the SyncLevel to 0. We use this patch because OS X does not support proper Mutex debugging and breaks on any Acquire with a Mutex that has a non-zero SyncLevel. Non-zero SyncLevel Mutex objects are one common cause of ACPI battery method failure. For example, in Lenovo u430 mutexes are declared like this:

```
Mutex (MSMI, 0x07)
```

To make it compatible with OS X you need to change it to:

```
Mutex (MSMI, 0)
```

This is a very controversial patch. Only use it if you fully understand what you are doing. Added by Rehabman in revisions 4265-4346.

FixRTC

Removes interrupt from _RTC device. This is necessary, and it is very strange that someone refuses such a patch. If the original does not have an interrupt, then this patch will not do anything terrible. However, the question arose about the need to edit the region length. To avoid resetting CMOS, the length must be set to 2, but in this case a phrase like ...only single bank... appears in the kernel log.
I don't know what's wrong with this message, it can be excluded if the length is set to 8. But in this case there will be a risk of getting a BIOS reset after sleep. Therefore, an additional key is introduced to resolve this trick:

```
<key>ACPI</key>
<dict>
	<key>DSDT</key>
	<dict>
		<key>Rtc8Allowed</key>
		<false/>
	</dict>
```

true - the region length will remain 8 bytes, if it was,
false - will be corrected by 2 bytes, which more reliably prevents the CMOS reset. As vit9696 researched, the region length should still be left at 8, because it is needed to save the hibernation key. And the fix itself is useful. Well, hibernation is not needed on desktops, so you can think about resetting the CMOS.

FixTMR

Similarly removes the interrupt from the _TMR timer. This timer is obsolete and is not used by the Mac. We simply free up one IRQ.

AddIMEI

A necessary patch for SandyBridge and above, which consists of adding the device IMEI to the device tree if it is not there already.

FixIntelGfx

The patch for integrated Intel graphics is separate from other graphics cards, meaning you can install injection for Intel and not install it for Nvidia.

FixWAK

Adds Return to the _WAK method. It must be there, but for some reason DSDTs often do not contain it. Apparently the authors adhered to some other standards. In any case, this fix is completely safe.

DeleteUnused

Removes unused floppy devices. What's there to worry about? In fact, it also removes CRT and DVI devices - an absolutely necessary condition for running IntelX3100 on Dell laptops. Otherwise, a black screen, verified by hundreds of users.

FixADP1

Adjusts the ADP1 device (power supply), which is necessary for the correct sleep of the laptop when it is plugged into the socket, or when it is removed from it.

AddPNLF

Inserts a PNLF (Backlight) device, which is necessary for proper screen brightness control, and, oddly enough, helps solve the sleep problem, including for the desktop.

FixS3D

Similarly, this patch solves the problem with sleep.

FixACST

Some DSDTs have a device or method or variable named ACST, but that name is used by MacOSX 10.8+ to manage c-state. A completely subtle conflict with very obscure behavior. This fix renames all occurrences of that name to something safe.

I can't understand how you can ignore this series of patches?! Don't you want a well-functioning computer?

AddHDMI

Adds the HDAU device to DSDT, corresponding to the HDMI output on the ATI or Nvidia video card. It is clear that since the card was purchased separately from the motherboard, there is simply no such device in the native DSDT. In addition, the hda-gfx=onboard-1 or onboard-2 property is injected into the device, depending on the circumstances:

- 1 ifUseIntelHDMI=false

- 2 if there is an Intel port that occupies port 1.

It's time to start sound on HDMI with the native AppleGFXHDA driver, which is designed for some AMD Radeon cards (Polaris, Vega, Navi) and some Intel. This driver itself finds the necessary device, and does not need a DSDT fix, rather the opposite, we must ensure that another driver does not take its place.

FixRegions

But this is a completely special patch. If the other patches were intended for editing BIOS.aml, for creating a good DSDT from nothing, then this fix is intended for the final adjustment of a well-made custom DSDT.aml, and it is useless for BIOS.aml. The thing is this.

In the DSDT there are regions that have their own addresses, for example:

```
OperationRegion (GNVS, SystemMemory, 0xDE6A5E18, 0x01CD)
```
The problem is that the address of this region is created dynamically by the BIOS, and it may be different from boot to boot. First of all, this was noticed when changing the total amount of memory, then when changing the BIOS settings, and on my computer it even depends on the boot history, for example, on the amount of occupied NVRAM. It is clear that in the custom DSDT.aml this number is fixed, and therefore may not correspond to the truth. The simplest observation is the lack of sleep. After fixing the region, sleep appears, but until the next shift.

This fix corrects all regions in the custom DSDT to the values in BIOS DSDT, and thus the mask

```
<key>Fixes</key>
<dict>
      <key>FixRegions</key>
      <true/>
</dict>
```

is a sufficient mask if you have a well made DSDT with all your necessary fixes.

There is another patch, but it does not concern DSDT, but all ACPI tables in general, and it takes up space here illegally.

FixHeaders

It will check headers not only DSDT, but also all ACPI tables in general, solving the problem of Chinese characters in table names, which MacOS does not transfer, immediately panicking. Whether you have a problem with tables or not, it is safer to enable this fix.

Select patches

How to choose which patches are necessary, which are harmless, and which are dangerous? Well, you will not ruin the computer in any case. All this happens only in RAM, and will be forgotten after reboot. You can test a set of fixes by correcting the mask in the graphical menu, and saving the result by pressing F5 - "Save DSDTxxxx.aml, corrected by the current mask". You can try and boot with the current mask. In order not to interfere with the real, patched DSDT, already present in the system, you can specify in the menu DSDT name: BIOS.aml If such a file is not found, the system will take the OEM DSDT from the BIOS and perform fixes on it, according to the set mask. In case of failure, after rebooting the computer, the current settings will be lost, and the default settings that are working for you will come into effect. Mask 0xFFFFFFFF corresponds to enabling all fixes, and if the OS boots after that, the programmers' work was not wasted. From the description above, you already realized that some fixes are simply unnecessary (for example, WIFI). Starting with revision 1992, work has been done to prevent panic on a double patch, so do not be afraid to set extra bits. I would not recommend using two fixes at the moment: FixSata, bit 0x0400, it is better to use a binary kext patch, and FixShutdown, bit 0x04, because the HaltEnabler=true setting almost always works instead, which works more correctly. Also dangerous is the AddMCHC patch, someone definitely needs it, and someone is categorically contraindicated. To see how DSDT patches affected the result, let's say you couldn't boot, you can write the following key in the config, in the ACPI section:

```
<key>DSDT</key>
<dict>
      <key>Debug</key>
       <true/>
```

In this case, before the system starts, two files will be saved to the disk in the folder
/EFI/CLOVER/ACPI/origin/
DSDT-or.aml
DSDT-pa15.aml
origin — is your DSDT loaded from disk, or taken from BIOS, before applying patches.

patched - after applying patches.
Since you were unable to boot, you will make more and more attempts, and the patched files will be numbered sequentially, without erasing the old information. 15 in this case is the 15th boot attempt, apparently successful, you need to look at what the problem was with the 14th attempt. And still, I recommend avoiding double patching. There may be a situation when double patching occurs because the OEM DSDT already has the _DSM method, when we want to inject our own. So you need to set the bits in the DropOEM_DSM mask. See the chapter "Hardware Configuration" → ACPI → DSDT→ DropOEM_DSM. No, you need to rename all OEM DSMs.

Manual DSDT editing

Prerequisites

You need to have some computer with some operating system to be able to edit text files. Or you have already installed macOS on this computer somehow, and now you want to improve DSDT.

There is a command line compiler iasl for any operating system. Including for Windows, iasl.exe runs in the command line as in Mac, has the same functions, and gives the same results. Editing texts in Windows is inconvenient, notepad does not have syntax highlighting and line numbers, it is better to take Notepad+. Mac has plenty of options, and Xcode, and BBEdit, and others. The new version of the compiler for Mac can be taken here Optimization Dsdt. The Newest Compiler.

For Windows on the website acpica.org Windows Binary Tools You should also stock up on the ACPI language description ACPI_6_2.pdf. It is better to take the new version, since you need to deal with the DSDT that the new BIOS has slipped you.

Creating a blank

First, make yourself a flash drive with Clover, bootable, even without a system. It is important that it is in FAT32 format. Boot on this computer from this flash drive to the Clover interface. Press the "O" key (Latin letter Oy) or select the Options icon in the menu. Go to the ACPI section -> there we find DSDT Name: and enter BIOS.aml. This is exactly the DSDT that you could get in Windows via Aida. Go down the menu and select DSDT fixes → there you can check the boxes with the keyboard or mouse. Check almost all the boxes that do not cause you rejection. For example, Firewire, Airport, IDE are not needed if you know that you do not have such devices. It is better to check everything unfamiliar. Return from this submenu and press the F2, F4, F5 keys in sequence.

You can turn off the subject and take the flash drive to the computer where you will edit the files. The files we are interested in are located on the flash drive in the folders

```
EFI\CLOVER\ACPI\origin
EFI\CLOVER\misc
```

Copy these folders to your working folder on the computer where iasl is. Whether with Mac or Windows. The instructions are the same, except for subtleties like slanted slash.

Decompilation

The resulting DSDT.aml is a binary file, you can't view it with a text editor. I strongly advise against specialized editors maciASL, DSDTEditor and the like, since the goal of the topic is not that

a kind uncle rehab will do everything for you, but to see for yourself what and how is happening.

Launch the command line: cmd.exe in Windows, Terminal.app in Mac, I don't know what it is in
Linux, I think it's called bash.

```
> cd WorkingFolder\origin
```

that is, go to the same copy of the folder that was copied from the flash drive. Put iasl.exe in the same folder if you are in Windows, or install iasl in the system if you are in Mac

```
$ sudo cp ~/Downloads/iasl /usr/local/bin/
```

Now, finally, it is possible to decompile, that is, obtain DSDT in a readable language.

```
$ iasl -da SSDT*.aml DSDT-*.aml
```

The -da option does a magic thing, it searches for symbols in all files at once, so that in the end there should be no unknown symbols. However, this is in the ideal case. In practice, it does not work, we get a lot of errors like AE_ALREADY_EXISTS

```
~~~
Firmware Error (ACPI): Failure creating named object [IO1B], AE_ALREADY_EXISTS
(20200925/dswload-495)
ACPI Error: AE_ALREADY_EXISTS, During name lookup/catalog
(20200925/psobject-372)
Could not parse external ACPI tables, AE_ALREADY_EXISTS
~~~
```

Another option turns out to be more successful

```
iasl -e SSDT*.aml -d DSDT-*.aml
```

At the output we have a warning

```
Parsing completed
 Warning - Emitting ASL code "External (BNUM)"
           This is a conflicting declaration with some other declaration within
the ASL code.
           This external declaration may need to be deleted in order to
recompile the dsl file.
```

That is, in the received DSDT.dsl you will need to remove the declaration

```
  External (BNUM, UnknownObj)    // Conflicts with a later declaration
```

Because there is an announcement

```
    Field (GNVS, AnyAcc, Lock, Preserve)
    {
…
        BNUM,   8,
```

That is, the name is declared here, and there is no need to call it external, and even unknown. But there are errors further on

```
    External (_PR_.CFGD, UnknownObj)
```

However, this variable exists and is described in one of the SSDTs. The decompiler is still incomplete. We will have to deal with each of these cases individually.

We decompile the DSDT that Clover patched for us, and thus already have a lot of useful fixes. Similarly, you can decompile the original file

```
$ iasl -da SSDT*.aml DSDT.aml
```

And be able to compare what was and what became. And we got a file with a long name DSDT-1234567.dsl, you will have other numbers and letters. This is the original template, which needs to be renamed to DSDT.dsl, and the original to DSDT-origin.dsl, edited and compiled in an endless iterative process:

1. Edit with a text editor.
2. Compile, get DSDT.aml

compilation command

```
$ iasl -ta DSDT.dsl
```

3. Test, that is, put it in the EFI\CLOVER\ACPI\patched folder and start the system.
4. Return to point 1.

Unfortunately, compilation may fail with the following error

Код:

```
Input file SSDT-10x.aml, Length 0x37F (895) bytes
ACPI: SSDT 0x0000000000000000 00037F (v02 PmRef  Cpu0Cst
00003001 INTL 20120913)
Pass 1 parse of [SSDT]
ACPI Error: [C3ST] Namespace lookup failure, AE_ALREADY_EXISTS
(20160729/dswload-462)
ACPI Exception: AE_ALREADY_EXISTS, During name lookup/catalog
(20160729/psobject-310)
Could not parse external ACPI tables, AE_ALREADY_EXISTS
```

This means that the description of the C3ST symbol was found in two different SSDTs, the latest of which is SSDT-10x.

In my case it was similar to SSDT-5x, the only difference being that 5x is ACPI1.0, and 10x is ACPI2.0. And the names are the same! No wonder I have something like this in my kernel log. It's a BIOS error!

There are also two identical tables in the laptop. I checked, they are indeed both present in the BIOS, it is not my mistake. What to do? Before compiling, remove the duplicate, and in real work, drop extra tables in the Clover config. In case of duplicates, both will fly away.

What to fix

1. Syntax errors made by the manufacturer of this computer
2. Semantic errors
3. Tricks found on the Internet. Regarding this point, everyone thinks that this is the only thing that needs to be done. No, the first two points are also important.

Syntax errors

Let's watchthe file itself DSDT.dsl and in the file itself we see the first problems

```
External (_SB_.PCI0.PEG0.VID_.LCD_, UnknownObj)
```
This means that somewhere in the text there is a reference to the object, but the object itself is nowhere to be found. Let's search. It is in SSDT-7. It turns out that it was not exported from there. This is the reason why this laptop is offered with thisSSDT include within the generalDSDT. By simply copying the text from the SSDT from the first curly bracket to the last one into the tail of DSDT.dsl before the last bracket.

Second error with missing characters

```
External (HNOT, MethodObj) // Warning: Unknown method, guessing 1 arguments
```

The search shows that the method is mentioned in this construction

Code:

```
      If (CondRefOf (HNOT))
       {
         HNOT (Arg0)
       }
       Else
       {
         Notify (GFX0, 0x80) // Status Change
       }
```

Then everything is fine, if the method is undefined, then it is not executed. I don't understand, how does it compile then? It is better to delete this, leave only Notify...
I start compiling the file I just got (I renamed DSDT-1F2C3B4D.dsl to something simpler DSDT1.dsl as a first attempt)

```
$ iasl -ta DSDT1.dsl
1 errors, 14 warnings, 91 remarks, 109 optimisations
```

The error must be corrected, otherwise there is no result.
Warnings also need to be corrected, thinking about why the compiler complains. This is still a good situation, I have encountered hundreds of errors during the first compilation.

In this case, the error is not critical, in the line
```
Name (_HID, "*pnp0c14")
```
the character string format is invalid, it is scientifically corrected as follows:
```
Name (_HID, EisaId ("PNP0C14") /* Windows Management Instrumentation Device */)
```
It needs to be fixed, otherwise it won't compile, but it doesn't affect the work, it's a Windows feature.

Warnings are actually more critical, here are some examples of corrections.
- what was
+ what did

```
- CreateDWordField (BUF0, \_SB.PCI0._Y0F._LEN, MSLN) // _LEN: Length
+ CreateQWordField (BUF0, \_SB.PCI0._Y0F._LEN, MSLN) // _LEN: Length
```

The hint was in the compilation log
```
ResourceTag larger then field (size mismatch tag 64bit, Field 32 bit)
```
The rule is simple, tag is what is needed, field is what needs to be corrected

```
tag=1 => CreateBitField
tag=8 => CreateByteField
tag=16 => CreateWordField
tag=32 => CreateDWordField
tag=64 => CreateQWordField
```
That is, in the message tag 64bit, it means change `CreateDWordField` to `CreateQWordField`. It needs to be fixed so that in real work the sent value does not go to the neighboring field. There is no way to catch such a glitch.

Such a warning
```
Not all control path return a value
```
Logical error, the method should return something, but it turns out that in some cases it will return nothing. And? The Mac, of course, will not crash, but do not expect adequacy either. What should I write there? Return(Zero) or Return(Local0)? To calm the compiler down, this will do, but in general you need to look at the logic. A similar glitch
```
Reserved method should not return a value
```
The code is as follows
```
      Method (_SRS, 1, Serialized)  // _SRS: Set Resource
Settings
      {
        Return (BUF2) /* \_SB_.PCI0.A_CC.BUF2 */
      }
```
We open the PDF mentioned above, the ACPI specification, find the _SRS method by searching, and read what it should do. Returning values is not provided there.
Therefore, we rework it as follows

```
      Method (_SRS, 1, Serialized)  // _SRS: Set Resource
Settings
      {
        BUF2 = Arg0
      }
```
This seems more logical.

Well, and the well-known warning
```
Name (_T_0, Zero)
Use of compiler reserved name _T_0
```

The new iasl compiler understands this construct perfectly, and you don't need to do anything. The old compiler required renaming to TT_0

Semantic errors

Sorry, but you have to be at least a bit of a programmer to catch them. There are no clues to be found.
In the originalDSDT we see
```
External (CFGD, IntObj)
```
And we find this variable inSSDT CpuPm. And here it's time to remember that we are dropping this table, along with this variable! It needs to be copied to DSDT. Like others that might be useful.

```
+         Name (CFGD, 0x0066F6FF)
+         Name (PDC0, 0x80000000)
+         Name (PDC1, 0x80000000)
+         Name (PDC2, 0x80000000)
+         Name (PDC3, 0x80000000)
+         Name (PDC4, 0x80000000)
+         Name (PDC5, 0x80000000)
+         Name (PDC6, 0x80000000)
+         Name (PDC7, 0x80000000)
+         Name (SDTL, Zero)
```

A common mistake is OEM _DSM methods.
This is not a manufacturer's error, this is because he wrote it for Windows, and we have a hackintosh.
Example

```
-          Method (_DSM, 4, Serialized)  // _DSM: Device-Specific
Method
                           {
-                              Name (_T_0, Zero)  // _T_x: Emitted by
ASL Compiler
                              If (Arg0 == ToUUID ("a5fc708f-8775-4ba6-
bd0c-ba90a1ec72f8"))
                              {
                                  While (One)
                                  {
```

See the UUID? It's from the Windows registry, there's nothing like that on Mac, and nothing will be executed.
This would be half the trouble, but if for Windows it is normal to have _DSM for both the device and its bridge, then in Mac it causes a crash. Kill all other _DSM! Simple option

```
-   Method (_DSM, 4, Serialized)  // _DSM: Device-Specific Method
+   Method (ZDSM, 4, Serialized)
```

The method itself has been preserved, but no one will turn to it anymore, and there will be no panic.
Here is such a curious piece

```
            OperationRegion (DXHC, SystemMemory, 0xFED1F418, 0x04)
                XHCD,   1
             }

            If ((OSYS < 0x07D6) && (OSYS > 0x03E8))
             {
                 XHCD = One
                 Notify (XHC, Zero) // Bus Check
             }
```

Looking closely at the address, I realized that it was Function Disable Bit.
The point of the operation is to prohibit USB3 for systems lower than Windows Vista. In my opinion, this piece should be cut out altogether.

Now this is already a bummer of the writers

```
If ((OSYS > 0x07D0) || (OSYS < 0x07D6))
```

The ranges add up and cover any value at all. Rather, there
should be

```
If ((OSYS > 0x07D0) && (OSYS < 0x07D6))
```

Then the ranges intersect.
But then you need to look at what's in If and what's in Else.

I got it that it is more correct in Then, so the original construction

```
                    If ((OSYS > 0x07D0) || (OSYS < 0x07D6))
                     {
                         Notify (PCI0, Arg1)
                     }
                     Else
                     {
                        Notify (GFX0, Arg1)
                     }
```

I reduced it to just one operator.

```
Notify (PCI0, Arg1)
```

The thing is that this If checks if the system is WindowsXP and does what Then does for systems like Windows7,8,10, Else does. What's the difference? The new systems run Optimus. INmacOS we need notification of the first type, for the entire bus. And we need to look at it in the same way in other places DSDT.

A short detective story on the topic, maybe it will give someone the right idea. So, the task. The laptop sleeps and wakes up if it is not plugged into a socket. But then the battery runs out. But if it is plugged in, then immediately, a second after it falls asleep, it wakes up. What's the matter? It is precisely in the DSDT! So, let's look for what makes it differentACPI state in the socket and not.

```
       Device (AC)
       {
           Name (_HID, "ACPI0003" /* Power Source Device */)  //
_HID: Hardware ID
```

The decompiler helpfully told us that this is the power supply.
The _PSR method determines whether this device is powered or resting.

```
            Method (_PSR, 0, NotSerialized)  // _PSR: Power
Source
            {
                Local0 = ECG5 ()
                Local0 &= One
                If (Local0 != PWRS)
                {
                    PWRS = Local0
                    PNOT ()
                }

                Return (Local0)
            }
```

You can read about this in the book ACPIspec.pdf, any year of publication

> *An Integer containing the power source status*
> *0 - Off-line (not on AC power)*
> *1 - On-line*

That is, in my case, some hardware method ECG5() returns a certain number, in which bit 0 means plugged into the socket. And saves it in the PWRS variable if the value there is different. Now we look throughout the code to see where PWRS is mentioned, and find it in the _PTS (Prepare To Sleep) method, the one responsible for falling asleep.

```
   Method (_PTS, 1, NotSerialized)  // _PTS: Prepare To Sleep
    {
        P80D = Zero
        P8XH (Zero, Arg0)
        PTS (Arg0)
```

```
        If (AOAC & One) {}
        If (Arg0 == 0x03)
        {
            If (PWRS == Zero)
            {
                \_SB.PCI0.XHC.PMEB = Zero
                \_SB.PCI0.EHC1.PMEB = Zero
                \_SB.PCI0.EHC2.PMEB = Zero
                If (\_SB.PCI0.XHC.PMST == One)
                {
                    \_SB.PCI0.XHC.PMST = One
                }

                If (\_SB.PCI0.EHC1.PMST == One)
                {
                    \_SB.PCI0.EHC1.PMST = One
                }

                If (\_SB.PCI0.EHC2.PMST == One)
                {
                    \_SB.PCI0.EHC2.PMST = One
                }
            }
```

It's interesting here! Condition: if the device is not plugged into the socket, then do something in USB2 and USB3. And if it is plugged in, then nothing needs to be done. In the log of unexpected awakening we see XHC, EHC2. That is, the creators of this DSDT believe that a laptop can only be plugged into a socket when not in use. To work, pull it out. By removing this condition If (PWRS == Zero) I got a solution to my problem. Now my laptop sleeps and wakes up, and the power adapter does not disturb it.

Good luck in creating a minimally correct DSDT!

Native Speedstep

It is more correct to say Power and Processor Frequency Management (PPM), in English it would be EIST – Enhanced Intel Speedstep Technology, hence the Russian word “Speedstep”.

Actually, this topic is not so much for the bootloader as for setting up HackOS in general, but since Clover takes some steps, we will describe it in a separate chapter. Clover doesn't do everything that needs to be done, and a little manual work is required.

Why is this even necessary? The idea is this: the processor, when idle, operates at minimum frequency with minimum voltage, and under load, the speed and voltage increase. (And why the voltage? Because the pulse front becomes steeper, and therefore gains a level faster, and moves from state 0 to state 1 faster).

UPiChP can be implemented in two ways: with a specialized utility, such as CoolBookController or GenericCPUPM, or by understanding the native speedstep, fortunately MacOS can do this.

The following steps are required:

1. In DSDT, HPET must be corrected, which is successfully done by Clover with mask 0x0010.
2. There must be a correct processor section, which is done by Clover with the GeneratePStates=Yes key (and also DropSsdt)

3. MacModel should be selected as a sample of your SMBIOS, for which EIST technology is provided. It turns out, not for all models. For example, for the MacBook1,1 model, speedstep will not work, but for MacBook5,1 - it will.

Point 3 can be rethought as follows: let the model be more similar in configuration to the real one, but let's fix its platform plate so that the speedstep appears.
Each model has its own playlist, see here
`System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns/ACPI_SMC_PlatformPlugin.kext/Contents/Resources/MacBook5_1.plist`
We look at the similarities and differences between different slabs and correct ours in the right direction.

ConfigArray

```
<key>ConfigArray</key>
<array>
	<dict>
		<key>WWEN</key>
		<true/>
		<key>model</key>
		<string>MacBook4,1</string>
		<key>restart-action</key>
		<dict>
			<key>cpu-p-state</key>
			<integer>0</integer>
		</dict>
	</dict>
</array>
```

This restart-action key means which P-State the CPU should fall to when restarting. Only with this key did the computer sleep and shutdown work!

CtrlLoopArray

```
<key>CtrlLoopArray</key>
<array>
	<dict>
		<key>Description</key>
		<string>SMC_CPU_Control_Loop</string>
…
		<key>PLimitDict</key>
		<dict>
			<key>MacBook4,1</key>
			<integer>0</integer>
		</dict>
```

This PLimitDict key has already been mentioned in the P-states generation. Let us repeat: this is the maximum processor speed limit. 0 – the speed is maximum, 1 – one step below the maximum. If this key is missing here, the processor will be stuck at the minimum frequency.

CStateDict

```
<key>CStateDict</key>
<dict>
	<key>MacBook4,1</key>
	<string>CSD3</string>
	<key>CSD3</key>
	<dict>
		<key>C6</key>
		<dict>
```

```
<key>enable</key>
<true/>
```
Experience shows that it is better to delete this section entirely so that power management works by PState, not by CState. Although, depending on the individual, maybe this option is worth working on. Symptom - the processor is at maximum frequency, does not drop. After deleting the section, the frequency begins to vary. With Skylake processors, a new method of Processor Frequency Management has appeared, in their language it is called SpeedShift. I haven't figured it out yet, preliminary results: In the CPU section we register two keys (inventor - goodwin_c)

```
<key>CPU</key>
<dict>
	<key>HWPEnable</key>
	<true/>
	<key>HWPValue</key>
	<string>0x30002a01</string>
```

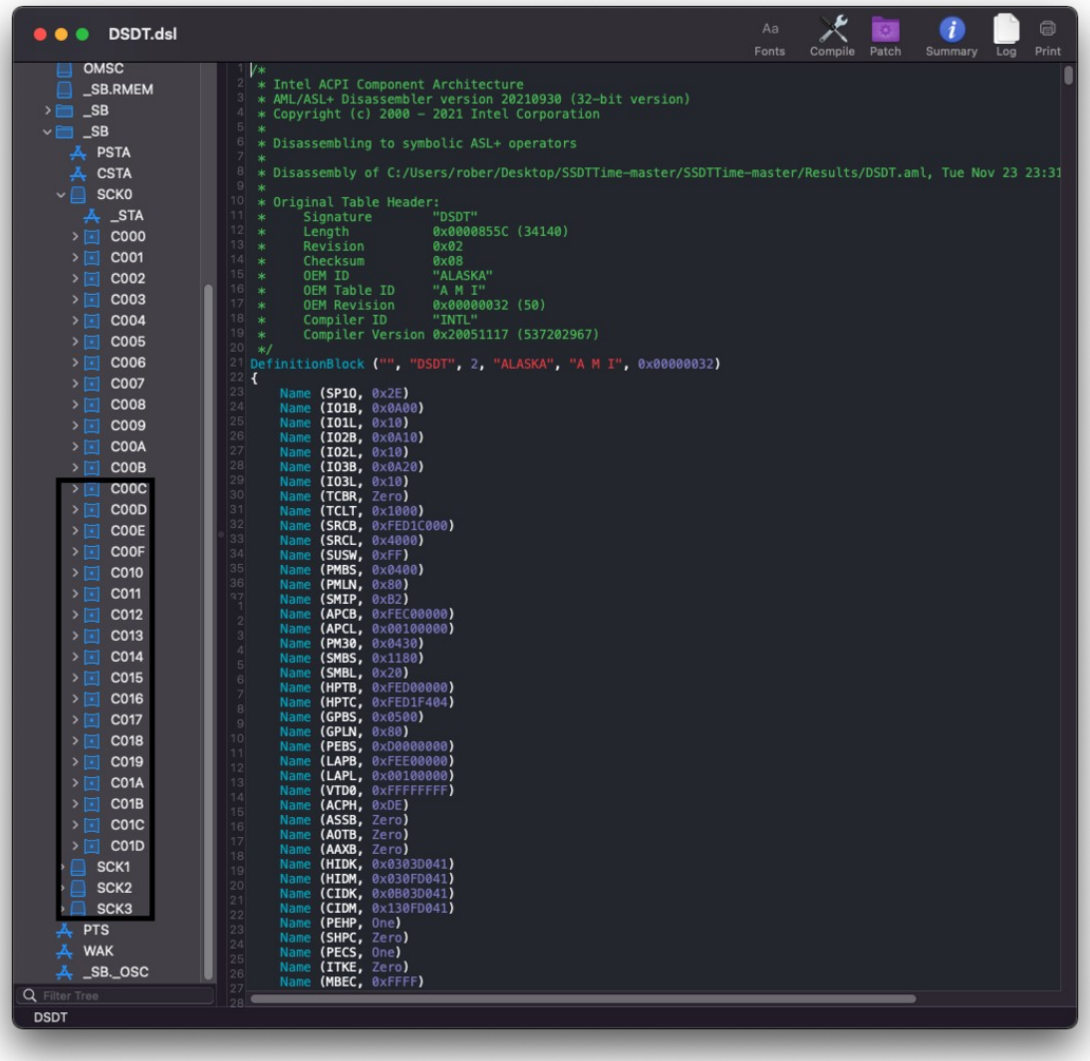
The first key puts 1 into MSR register 0x770. The second key writes the specified value into MSR register 0x774.

Synchronization of processor cores

Previously, somewhere with Merom processors, I think, there was a problem with synchronization of processor cores, leading to panic with words like "TSC synchro..." or "non-monotonic time...". Then the VoodooTSC kext was invented, for which you need to specify the number of threads in Info.plist, the creators were too lazy to calculate it automatically. Vit9696 came and did it CpuCstSync, which does this, but for some reason this kext stopped working in Monterey, so I just went back to the old, proven VoodooTSCSync. Merom is long gone, and somehow all this was forgotten.

The relevance of this action returned after the appearance of Xeon processors on Chinese motherboards, in particular X79, on the market. Yes, the processor itself is not to blame, it is the BIOS that initializes it incorrectly. So maybe we need to initialize it at the bootloader level? Vit9696 did this, the TscSyncTimeout quirk appeared, which needs to be set to a number in microseconds (or nanoseconds?). However, no, it does not help alone. The joint work of this quirk with the above-mentioned kext helps. The recommended value of this quirk is 500000 µs, that is, half a second. Not too much, if this processor has, say, 24 cores? But I came across a statement that any non-zero number is required. For example, 1. I was not able to check it anymore.

However, this is not all that needs to be told to the system for the CPU to work correctly. It is necessary to edit the DSDT processor section. Here I do not understand how this can be achieved by preserving a clean DSDT and adding external SSDT, as Acidanthera advocates. In my opinion, it is necessary to edit the DSDT. Initially, it looked like in the picture. It is necessary to cut out all SCK1,2 groups... except for the initial SCK0, and all unnecessary C0xx cores. Circled in a rectangle in the picture



These sections C000… also should be edited. It was

```
Processor (C000, 0x00, 0x00000410, 0x06)
{
    Name (_HID, "ACPI0007" /* Processor Device */)  // _HID: Hardware ID
    Name (_UID, "PCI0-CP000")  // _UID: Unique ID
```

What needs to be done::

```
Processor (C000, 0x00, 0x00000410, 0x06)
{
    Name (_HID, "ACPI0007" /* Processor Device */)  // _HID: Hardware ID
    Name (_CID, "PCI0-CP000")  // _CID: Compatible ID
    Name (_UID, Zero)  // _UID: Unique ID
```

And so on for all 16 logical cores, that is, 8 cores with hyperthreading.

Well, you should also remember about the ACPI PatchAPIC patch. It is also not superfluous in such cases. This patch is only in Clover, so not all Chinese computers can be launched by OpenCore.

Sleep problem

What about the sleep problem? When all of the above is done, the computer will go to sleep and wake up like an obedient child. The most important thing necessary for this, Clover has already done: corrected FADT and FACS. All that remains is to correct DSDT, start speedstep, use only good kexts, and you will be happy.

Doesn't go to sleep or wake up

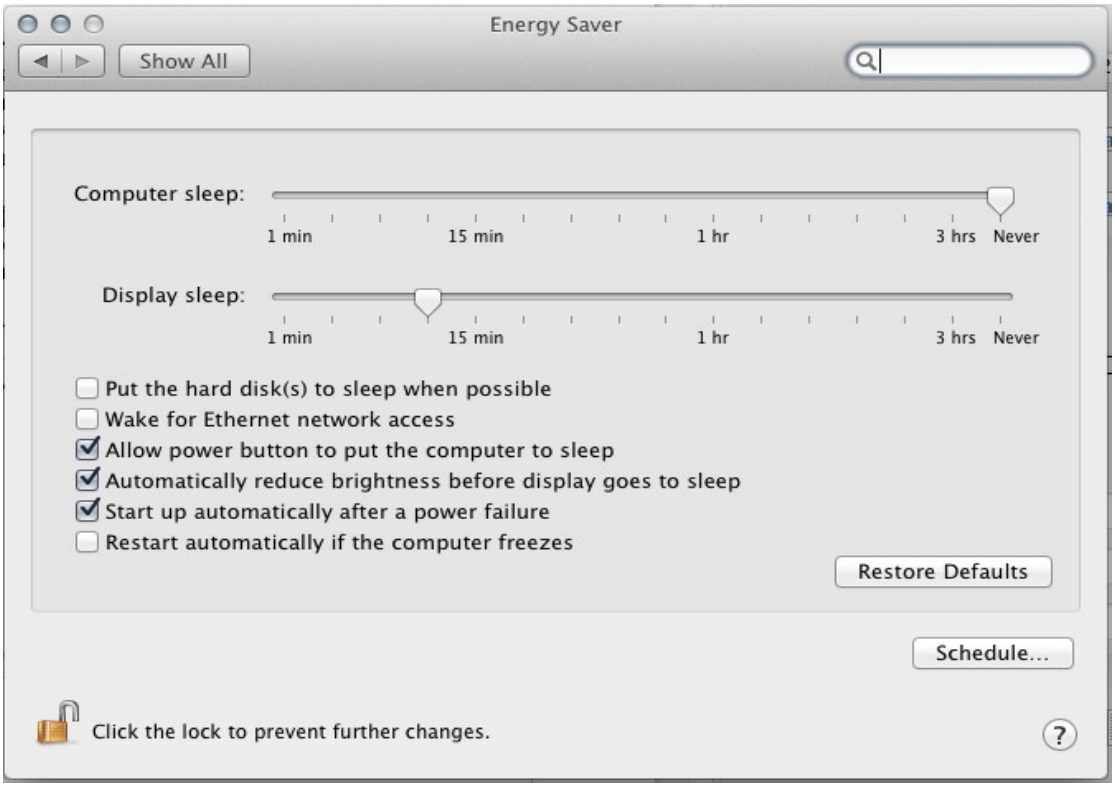
Any device, including an unset PCI, can interfere with a good night's sleep device, or started partially. For example, the sound chip, if AppleHDA is not fully configured. NullCPUPM.kext categorically interferes with sleep. You may not need speedstep, but you should patch NRET so that the native AppleCPUPM starts, and the null is not needed. And those whose processor does not allow using AppleCPUPM can try SleepEnabler and sometimes it helps, or a patched kernel. In DSDT there is a group of _GPE methods with notifications for each device, which need to wake up after sleep. The computer itself woke up, but it may turn out that video/network/sound/mouse forgot to wake up. Watch DSDT, learn the theory of how to do this. There was also a problem with sleep when UEFI booting into the 10.8 system. The problem was solved in revision 1942. Change in the OsxAptioFixDxe driver: sleep/wake up works even in 10.8, even with CsmVideoDxe. Now it's the ProtectMemoryRegions tweak..

The next trick is for UEFI boot

```
<key>ACPI</key>
<dict>
      <key>HaltEnabler</key>
      <true/>
```

This corrects the state of the chipset, incorrectly initialized by the UEFI BIOS. For legacy boot CloverEFI does everything correctly, there was no such problem there. Symptoms are does not go to sleep, the screen goes out, and the fans do not.

And one more trick.



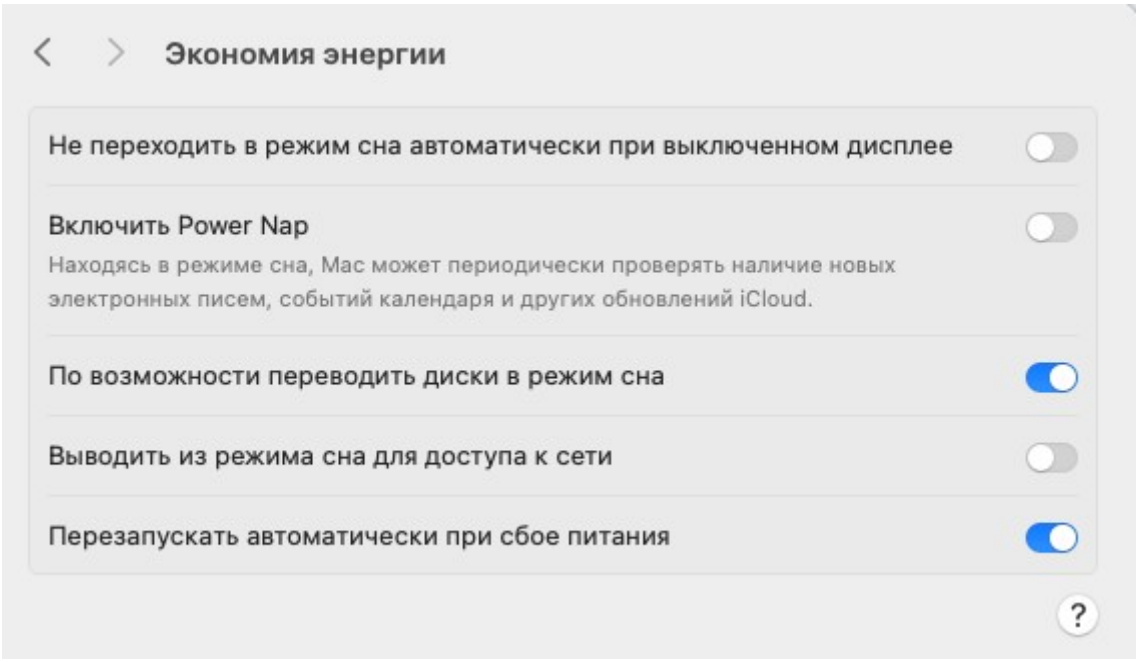
Without checking the box "Start up automatically..." I couldn't get it to wake up after sleep.

Spontaneous awakenings

Firstly, there may be a device like a video camera hanging on the USB bus, which will wake up the computer immediately as soon as it tries to go to sleep. Two options: either prohibit USB devices from waking up the computer in the BIOS (and I personally like to wake up the computer with a mouse click), or register UsbConnector 255 in the legacy keystore, that is, an internal device.

Secondly, Power NAP has appeared in the MacOS settings for some time now then automatically wake up to check for updates, and something else. Let's turn it off!

Thirdly, the checkbox in the screenshot above is “Wake for Ethernet access”, and now it’s “Display from sleep to access the network." I actually thought it was for something else, but no, the computer wakes up regularly with these check marks. Here is the correct picture



You also need to cancel all scheduled wakeups. sudo pmset sched cancelall.

```
sudo pmset sched cancelall
sudo chflags schg
/Library/Preferences/SystemConfiguration/com.apple.AutoWake.plist
```

And in Sequoia the next trick is to create a file

```
/Library/Preferences/FeatureFlags/Domain/powerd.plist
```

with contents

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
      <key>CoreSmartPowerNap</key>
      <dict>
            <key>Enabled</key>
            <false/>
      </dict>
</dict>
</plist>
```

Now my computer sleeps and wakes up with a mouse click, but never on its own!

Hibernate

It is also called deep sleep, but in general it is more like clinical death. The point is that the system is sent to sleep, but it saves its state in the sleepimage file, and simply turns off the computer, so that later, when turned on, it simply restores its state and wakes up. For laptops, this is crucial. During normal sleep, the computer does not turn off completely, and continues to use electricity, although less than in working condition, but it is still noticeable that the battery is completely discharged during some time of sleep. During hibernation, the battery is not used, and is discharged only due to its leaks.

Once upon a time, hibernate worked with Chameleon, but only up to version 10.7.2 (I think), then due to some changes in the system, this method stopped working. In Clover, it was possible to make hibernate, but under the following conditions::

- Loading either CloverEFI (legacy), or InsydeEFI, or Phoenix 2.3.1. The current revision 2915+ has the OsxAptioFix2Drv driver, which allows hibernation with AMI UEFI on a 10.9.1+ system. But the 10.7.5 system does not load at all with this option. And God bless it!
- The system is either 10.7.5 or 10.9.1+. Other systems do not wake up yet..
- Mode 21 or, better, 29 or even 57, although Apple insists on 25.
  `sudo pmset -a hibernatemode 29`
  Starting with Captain, it is impossible to set 29 like this.. The trick is as follows::
  - copy /Library/Preferences/com.apple.PowerManagement.511CE201-1000-4000-9999-120361221216.plist to the Desktop. There are several such files, choose the one that corresponds to the UUID in the System Information. Edit this file to add 29 there.

```
<key>Hibernate Mode</key>
<integer>29</integer>
```

- And copy the edited file back using the terminal
  sudo cp ~/Desktop/com.apple.PowerManagement.511CE201-1000-4000-9999-120361221216.plist /Library/Preferences/
  After this, a reboot is required, only then will the change take effect.
- If you have a real, iron NVRAM working, then you can make mod 25. And then in the config you write

```
<key>Boot</key>
<dict>
 <key>StrictHibernate</key>
 <true/>
```

- For system 10.13 and above you need to enable

```
<key>RtcHibernateAware</key>
<true/>
```

  because the encryption key can be written in CMOS, and because Clover must generate additional NVRAM variables. (We still need to think about the latter).

It works like this:
1. Set the mode to 29 (or 25), if it is not set yet. No need to repeat.
2. Send the computer to sleep either through the menu, or by closing the lid, or by pressing the power button, if configured for this. In a minute, the computer will go completely dark.
3. To wake up, simply turn it on as usual. We see the BIOS splash screen, enter the Clover menu. And here we see that our system is marked `hibernated)`

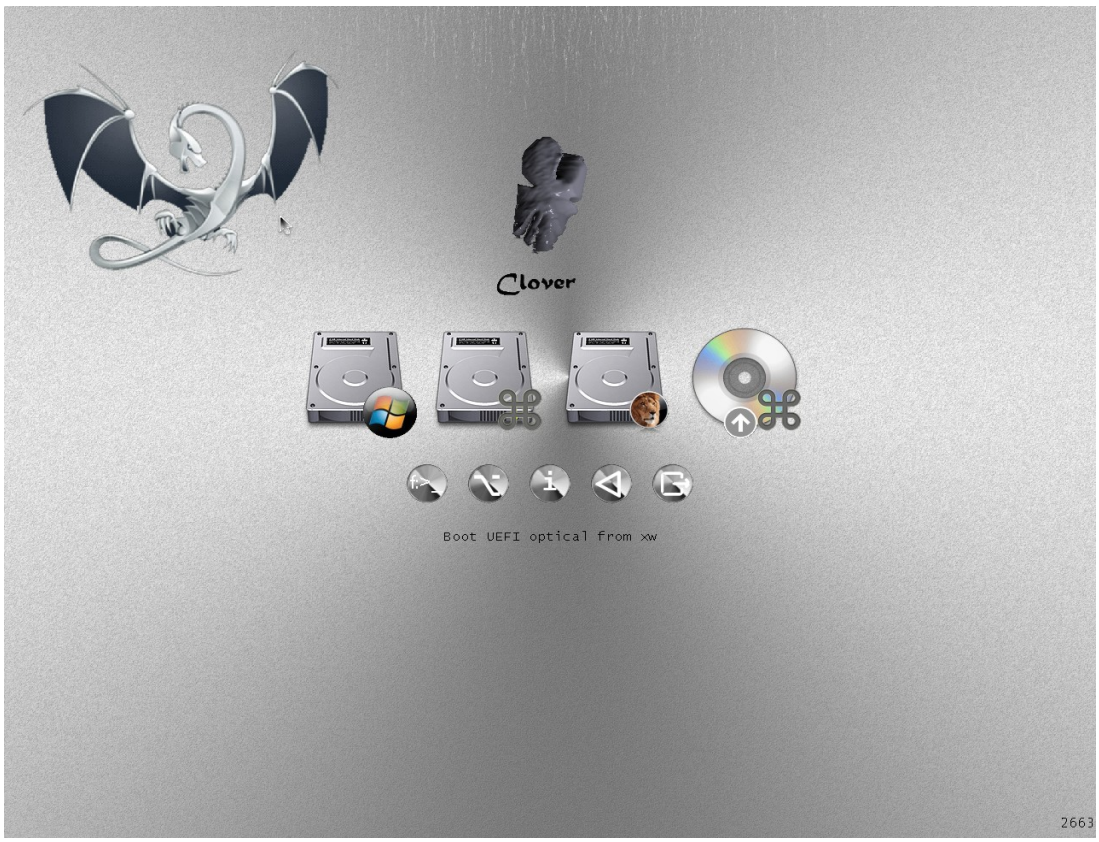
While other systems do not. When you click on this icon, the system boots from the image, a few seconds, progress is visible at the bottom, and the system turns on. This is much faster than a normal system boot, especially for laptops, and especially with a large number of open applications. It should be noted that if the file system of the volume was modified after hibernation, for example, from a system loaded from the second partition, then there will be a serious risk of damaging the file system, because the hibernating system has a cache with a different structure. For the 10.9 system, this difficulty is overcome automatically by comparing the modification date. In the 10.7.5 system, this does not work, monitor the correctness manually. You can cancel waking up from the image by pressing the space bar on this icon and selecting "Cancel hibernation". If the system continues to think that it needs to wake up, you will have to write in the config:

```
<key>Boot</key>
<dict>
        <key>NeverHibernate</key>
        <true/>
```

How to use

First meeting

First, boot into Clover's GUI and try living here for a while, press different keys, move the mouse..



The top row of buttons are the supposed operating systems that can be loaded. In this picture there are two of them, Lion and Windows, which is visible in the pictures. In reality, I remind you, Clover is not a bootloader for operating systems, it is a manager of their own bootloaders. Namely, for Mac the bootloader is /System/Library/CoreServices/boot.efi. For Windows, in this case, `/EFI/microsoft/boot/bootmgfw.efi`

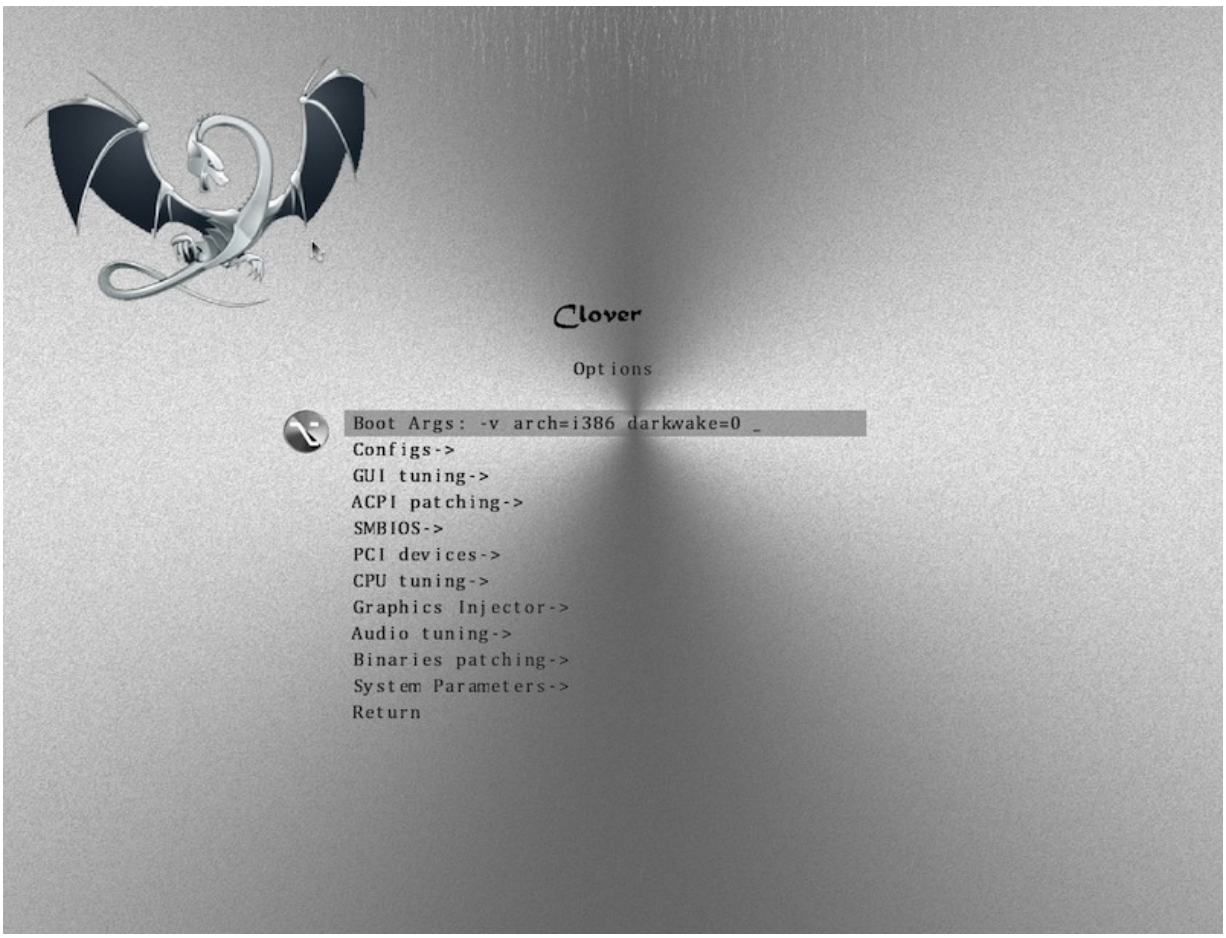
The bottom row of buttons are additional functions: command line (Shell), Options menu, information about the bootloader and environment, restart and exit from Clover. Exit where? Back to the EFI environment, to UEFI BIOS or to CloverEFI, respectively.

It is very useful for initial acquaintance to press F1 (who would have thought?!). If the config specifies

```
<key>GUI</key>
<dict>
	<key>Language</key>
	<string>ru:0</string>
```

then the help will be in Russian. The command line is something like DOS, with the ability to copy and delete files. How and for what - this is beyond the scope of this book. This is Shell.efi with its help.

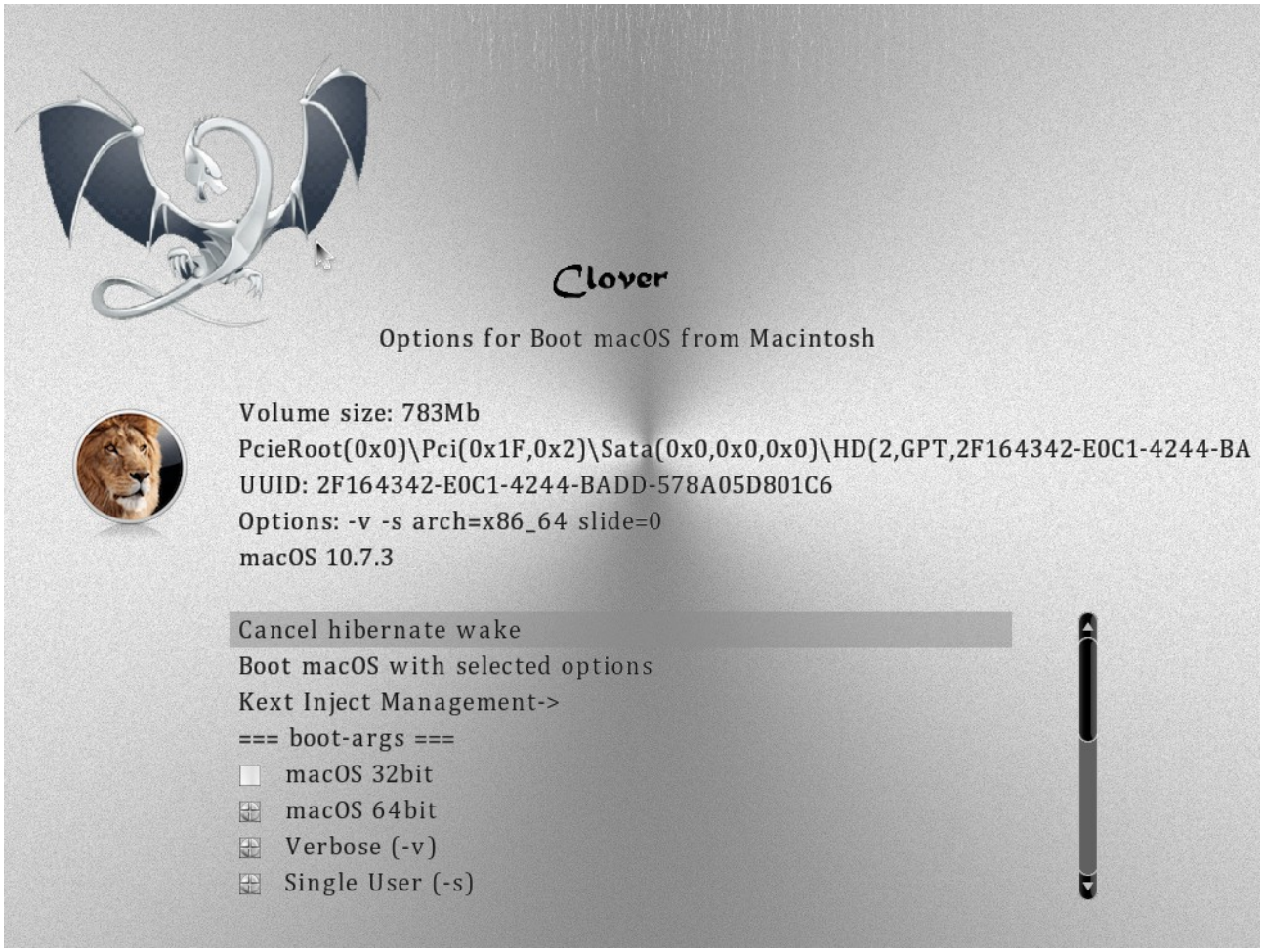
The Options menu allows you to change some settings that will affect the boot process of the system.



Some of them are specified in the config.plist file, but it could well be that they are written incorrectly, and in order not to edit this file yet, the settings can be changed when Clover is already running.

What exactly to change and for what purpose depends on the task, what exactly needs to be obtained. Requests like "I have a Western Digital HDD and Corsair memory, help me set up the config" are very annoying. The config is configured not by hardware, but by the result. If you can't boot right away, try to determine what the problem might be and fix it there.

Several special boot modes can be accessed by pressing the space bar on the bootloader icon. (again, ENTER - boot the system, SPACE - enter the additional boot menu).



In particular, only here can you prevent exiting hibernation if it is undesirable.

Why is Clover so slow to start?

Some can't even wait for the launch, reporting that Clover is not working.
Let's look at this issue more closely.

1. **debug.log is set**

```
<key>Boot</key>
<dict>
	<key>Debug</key>
	<true/>
```

to find out what is not working and why. But keep in mind that it slows down the startup very much. If Clover is installed on USB (for a test, so to speak, we do not believe that Clover is capable of working at all?!), then the startup with debug can last 10 minutes. Yes, really. This happens because the debug log is opened, closed and rewritten on the flash drive on each line. This guarantees that you will get information about the success of the startup even with a forced reset, so if you went to debug, wait! Or for the first try set <false/>. .

2. **Too many disks, partitions and files on them.**

Clover, in order to create a startup menu, has to scan all disks, all partitions, and all files on them to find which systems can be offered for startup. Well, wait! Or cancel the scan,,

```
	<key>GUI</key>
	<dict>
		<key>Scan</key>
		<dict>
			<key>Entries</key>
			<false/>
			<key>Legacy</key>
			<false/>
			<key>Tool</key>
			<false/>
		</dict>
```
and create the menu manually
```
	<key>GUI</key>
	<dict>
		<key>Custom</key>
		<dict>
			<key>Entries</key>
			<array>
				<dict>
```
True, it requires some mental effort to figure out what to write there. Or leave bootloader scanning for now, so that at least you can boot somewhere.
```
		<key>Scan</key>
		<dict>
			<key>Entries</key>
			<true/>
```
3. **A huge Windows or Linux partition,** or more than one. Scanning the Windows partition is performed if there is an NTFS driver. And this partition usually contains a million files, and we are going to look for bootmgr.efi among them. I would recommend installing Windows so that this file is located on the EFI partition, and deleting the NTFS.efi driver altogether, and thus not scanning the Windows partitions. The same with the Linux partition and the VBoxExt2.efi driver.
4. **Too many drivers in the /EFI/CLOVER/drivers/ folder** I foresee a situation when people will start producing their own drivers of this type, and there will be people willing to try them. For now, think about whether you need extra keyboard drivers, mouse drivers, LowMemoryFix... The thing is that if you boot from USB, reading and running all these files can take time.
5. **Unsupported mouse.** Unfortunately, not all mice are supported by the EFI driver that we have. A bad mouse may behave incorrectly on the screen or cause severe lags. Disable it for testing, or even permanently if it is incorrect
```
<key>Mouse</key>
<dict>
	<key>Enabled</key>
	<false/>
	<key>Speed</key>
	<integer>0</integer>
</dict>
```
6. **Slow HFS+ driver.** The official Clover comes with the VboxHFS.efi driver, which is licensed clean and understands links, but works slower than Apple's HFSplus.efi. Download this unofficial but fast driver somewhere and put it in the /EFI/CLOVER/drivers/UEFI/ folder. This also applies to legacy (..drivers/BIOS/) boot.
7. **A monstrously beautiful design theme** has been selected.
The more colors and animations a theme has, the longer it takes to load. Choose a built-in theme, it's the fastest
```
	<key>GUI</key>
	<dict>
		<key>Theme</key>
		<string>embedded</string>
```

8. **Use the newest Clover.**.
No matter what good advisers whisper, the new version is better than the old one, and it has fixed bugs, in particular, those that caused Clover to freeze. And starting with revision 3063, messages about the loading process appeared directly on the screen.

The captions are crooked because the good font has not been loaded yet, and this has caused many unflattering responses "How to remove captions on the screen?". Set timeout=0, and there will be no captions. But for a start they are very useful, many new users realized that Clover does work, just slowly. This chapter is written for them.
Or like this

```
<key>Boot</key>
<dict>
	<key>NoEarlyProgress</key>
	<true/>
```

Next, we will look at some techniques, special patches and working methods, collected according to the principle of reverse dictionary. There is a problem →here is the solution.

Analyzing Debug.log/Preboot.log

Why? Clover will give you information about the hardware that you won't see from the specs on the site, and it will also tell you how it handles configuring before you launch the system. How do you get it? Let me just say that it's the same thing, with a difference in success and speed. To get preboot.log, go to the Clover shell (GUI) and press the F2 key. The preboot.log file will be saved in the Clover folder EFI/CLOVER/misc/. The file ends with the phrase Enter GUI, which is logical. However, if you boot the system, the file will end at the start of the system itself, that is, everything that is possible. If Clover freezes, then debug.log will help. To get it, set in the config

```
<key>Boot</key>
<dict>
	<key>Debug</key>
	<true/>
```

In this case, the file is created line by line, and it is guaranteed that you will find the breakpoint in it. Example:

```
222:890  0:001  DefaultIndex=-1 and MainMenu.EntryCount=7
222:892  0:001  SetScreenResolution: 1366x768 - already set
0:100  0:100  MemLog inited, TSC freq: 2591578780
```

```
0:100  0:000  CPU was calibrated with ACPI PM Timer
```
Here we see that after the message about changing the screen resolution, a new log started. That is, the error should be looked for in the text of Clover after this message, and before the next message, which was not there. Tell these lines to the developer, he will look. Let's now analyze what we see in the log.

```
MemLog inited, TSC freq: 2591583140
```
This is the initial clock frequency of the processor. By default, the processor starts with one core at the maximum non-turbo frequency. If this is not the case, immediately ring the alarm! It will not work. Such situations are rare, and we deal with each one individually with special Clover amendments. The number is not round, and this is correct! It will be round only in utilities that round for ease of display.

```
Now is 14.7.2019,  8:47:7 (GMT)
```
Launch date and time according to Greenwich Mean Time. Local time in Moscow will be +3. Let's distinguish the old log from the new.

```
0:100  0:000  Starting Clover revision: 5120 (master, commit dddceaae3) on
American Megatrends EFI
```
Here and below we find out what version of Clover the user used, and whether he has UEFI boot, as in this case, or legacy (it would be Clover EFI).

```
=== [ Get Smbios ] ====
```
Next comes the decoding of the memory modules, as they were recognized by the BIOS. Usually correctly, so we put in the config
```
      <key>SMBIOS</key>
      <dict>
            <key>Trust</key>
            <true/>
```
And if you don't like the values, then false. Clover will then determine by itself by reading SPD or XMP. In SMBIOS, the values are also taken from reading SPD, according to BIOS algorithms. We have good algorithms, but universal, and in BIOS, specific ones for this motherboard, so it is not a fact which is more accurate..
```
      <key>Boot</key>
      <dict>
            <key>XMPDetection</key>
            <string>-1</string>
```
See the chapter on configuration.

```
Running on: 'Latitude E6430' with board '0H3MT5'
```
And this is information about the name of your computer.

```
=== [ GetCPUProperties ] ====
```
Here is information about the processor, both CPUID and its own name .
```
BrandString = Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz
```
And other important characteristics
```
MSR 0xE2 before patch 1E008404
```
This is a lock, a terrible sin for a Mac. If possible, reflash the BIOS to an unlocked one. If not, you can live quite well by installing patches
```
      <key>KernelAndKextPatches</key>
      <dict>
            <key>KernelPm</key>
            <true/>
            <key>AppleIntelCPUPM</key>
            <true/>
```

Clover should do it automatically, but it's better if you do it manually. I generally recommend always setting it to True, because macOS tends to configure C-state PowerManagement in its own way. But the thing is, we have a Hackintosh, which means that C-states are not controlled by the SMC controller, like on native Macs, but by its own Nuvoton-type microcircuit, which, in fact, does not obey Apple commands. So what do we expect?

```
Turbo: 31/31/31/33
```
These numbers mean that a frequency of 3300 MHz is achievable on one core, and only 3100 on two or more cores. Since all cores are enabled in the Mac, there is no need to make a claim that the processor in the Mac does not overclock to the maximum possible frequency of 3300. However, I have another line
```
 The CPU not supported turbo
```
This is most likely the BIOS that has prohibited the use of turbo frequencies. You need to look at the BIOS settings.

```
=== [ GetDevices ] ===
```
I really like this section, it tells what PCI (PCIe) devices are found in the computer, along with their addresses and DeviceID/VendorID. It is very helpful in finding out if it is possible to find drivers for this device
```
0:100  0:000  PCI (00|00:00.00) : 8086 0154 class=060000  MCH
0:100  0:000  PCI (00|00:01.00) : 8086 0151 class=060400
0:100  0:000  PCI (00|00:02.00) : 8086 0166 class=030000  VideoCard
0:100  0:000   - GFX: Model=Intel HD Graphics 4000 (Intel)
0:100  0:000  PCI (00|00:14.00) : 8086 1E31 class=0C0330  USB3.0
0:100  0:000  PCI (00|00:16.00) : 8086 1E3A class=078000  IMEI
0:100  0:000  PCI (00|00:16.01) : FFFF FFFF class=FFFFFF
0:100  0:000  PCI (00|00:16.03) : 8086 1E3D class=070002  SerialPort
0:100  0:000  PCI (00|00:19.00) : 8086 1502 class=020000  LAN
0:100  0:000   - LAN: 0 Vendor=Intel
0:100  0:000  PCI (00|00:1A.00) : 8086 1E2D class=0C0320  USB2.0
0:100  0:000  PCI (00|00:1B.00) : 8086 1E20 class=040300  HDA
0:100  0:000  PCI (00|00:1C.00) : 8086 1E10 class=060400
0:100  0:000  PCI (00|00:1C.01) : 8086 1E12 class=060400
0:100  0:000  PCI (00|03:00.00) : 14E4 4353 class=028000  WiFi
0:100  0:000   - WIFI: Vendor=Broadcom
0:100  0:000  PCI (00|00:1C.02) : 8086 1E14 class=060400
0:100  0:000  PCI (00|00:1C.03) : 8086 1E16 class=060400
0:100  0:000  PCI (00|00:1C.05) : 8086 1E1A class=060400
0:100  0:000  PCI (00|0C:00.00) : 1217 8221 class=080501  SD-reader
0:100  0:000  PCI (00|00:1D.00) : 8086 1E26 class=0C0320  USB2.0
0:100  0:000  PCI (00|00:1F.00) : 8086 1E55 class=060100  LPC
0:100  0:000  PCI (00|00:1F.02) : 8086 1E03 class=010601  SATA AHCI
0:100  0:000  PCI (00|00:1F.03) : 8086 1E22 class=0C0500  SMBUS
0:100  0:000  PCI (00|00:1F.06) : FFFF FFFF class=FFFFFF
```
Some devices are also commented.
FFFF means that the device is not connected, although it is somehow present. Device class is Video, Audio, USB2.0, USB3.0, LAN, WiFi and so on. All the decodings are known from the PCI specifications. I added the class decodings in the right column, they are not in the log. Class 060400 is a bridge, a slot where you can insert a device. There are no USB devices here. Clover is not old enough to scan them. And here's another point, from someone else's log
```
- GFX: Model = GeForce GTX 760 family CE (Fermi)
```
What do we see?! 760 should be Kepler! Why does Clover say that she is Fermi? Clover is right, he can't be fooled by flashing the BIOS, information about the video card family is taken not from the BIOS, but from the functioning of the video core. In this case, this is the exposure of the Chinese deceiver.

```
0:128  0:027  EFI\CLOVER\#.plist not loaded with name from LoadOptions: Not
Found
0:143  0:015  EFI\CLOVER\config.plist loaded: Success
```
This feature of Clover is almost never used. The thing is that there is a place in the BIOS where you can flash the name of the config.plist file, and thus you can choose which config to load Clover with directly in the BIOS. In this case, such a name was not found, and therefore the standard config.plist is loaded. But in general, you can use it if you remember the 2014 instructions from Dmazar. There is another list below in the log
```
0:346  0:003  === [ Found config plists ] ==============================
0:403  0:057  - config0.plist
0:403  0:000  - config1.plist
0:403  0:000  - config.plist
```
This is already for the Clover menu, to change the config from there. But the Boot, GUI and KernelAndKextPatches sections will not change. Too late!

```
0:143  0:000  === [ GetListOfThemes ] ==================================
0:162  0:018  - [00]: embedded
0:176  0:014  - [00]: random
0:177  0:001  - [00]: metal
0:188  0:010  - [01]: BGM
0:223  0:035  - [02]: CESIUM
0:285  0:061  - [03]: METAL@2X
0:298  0:012  - [04]: Clovy
0:343  0:044  - [05]: BOOTCAMP
```
This is understandable, a list of all installed themes. Unless you make sure that the theme selected in the config is one that actually exists.

```
KextsToPatch: 13 requested
KernelToPatch: 1 requested
```
A small list of what is set in the config for patches of kexts and the kernel. We look at someone else's config and criticize why he made
certain patches.

```
0:749  0:002  === [ LoadDrivers ] ======================================
0:861  0:111  Loading ApfsDriverLoader.efi  status=Success
0:883  0:021   - driver needs connecting
0:885  0:002  Loading AudioDxe.efi  status=Success
0:893  0:007   - driver needs connecting
0:895  0:002  Loading DataHubDxe.efi  status=Success
0:917  0:022  Loading EnglishDxe.efi  status=Success
0:926  0:009  Loading Fat.efi  status=Success
0:935  0:008   - driver needs connecting
0:937  0:002  Loading FSInject.efi  status=Success
0:944  0:007  Loading OsxAptioFix3Drv.efi  status=Success
0:952  0:007  Loading SMCHelper.efi  status=Success
0:959  0:007  Loading VBoxHfs.efi  status=Success
0:966  0:007   - driver needs connecting
0:968  0:002  4 drivers needs connecting ...
0:970  0:002  PlatformDriverOverrideProtocol not found. Installing ... Success
0:974  0:004  APFS driver loaded
0:978  0:003  Searching for invalid DiskIo BY_DRIVER connects: not found, all ok
```
And the next reason for criticism is why the user loads these drivers, and why he does not load others.

```
SetScreenResolution: 1366x768 - already set
```
I ordered this resolution for my screen. And it was successfully set. For those whose desires differ from reality, see the configuration chapter.

```
4:481  0:002  === [ GetMacAddress ] ====================================
4:561  0:080  MAC address of LAN #0= D4:BE:D9:6C:86:CD:
```
Clover can read the MAC address of almost any network card. We use this information to set our own ROM variable value. This does not work on some UEFI BIOS, so we are looking for other ways.

```
=== [ ScanSPD ] ====
```
Checking memory modules if we don't trust BIOS.

```
=== [ GetAcpiTablesList ] ====
```
List of ACPI tables found in BIOS. Useful if you want to drop some
```
- [06]: SSDT  CpuPm len=2850
```
There is also ID = CpuPm and length 2850. You can drop it either way, depending on the uniqueness.

```
=== [ GetUserSettings ] ===
```
some selective information about what exactly is set in the config, especially useful for reading other people's logs.

```
=== [ ScanVolumes ] ===
```
List of volumes with their addresses and UUIDs. A volume is either a partition or a whole disk. Useful to see when "Clover doesn't see my partition!"

```
=== [ InitTheme ] ====
```
Further
information about successful creation of the graphical interface with the selected theme. For example, I see
```
OSicon os_mav not parsed
```
that is, in my selected theme there is no Mavericks icon, just the Mac icon will be used. Here is also information about the startup sound, which depends on the theme.

```
6:511  0:002  === [ Dump SMC keys from NVRAM ] ==========================
6:570  0:059  found AppleSMC protocol
6:584  0:014  Registered 17 SMC keys
```
In most cases, SMC keys do not matter at startup. They are strictly necessary for FileVault2 and for Hibernation. They are provided by the SMChelper.efi driver and the Clover infrastructure. (There is also a VirtualSMC option with its own infrastructure). They seem to be not needed for system installation, but... who knows!

```
=== [ ScanLoader ] ===
```
And here is a list of what can be downloaded from where. And also information if some system is in hibernation.

```
=== [ GetEfiBootDeviceFromNvram ] ====
```
The success of this operation determines whether Clover will autostart on timeout. See the instructions in the corresponding chapter. Success looks like this
```
Boot redirected to Entry 3. 'Boot macOS from HighHD'
```
That is, my system boots from the HighHD disk after a timeout.

```
=== [ StartLoader ] ===
```
We are starting to boot the system
```
GetOSVersion: 10.13.6 (17G7024)
```

To see what system the user is talking about loading.

Next comes information about what exactly Clover does before loading the selected system, what patches, what properties are generated, what kexts are loaded, and the last line Unlock USB2.0 if needed.

```
USB EHCI Ownership for device 1E26 value=1000001
```

In version 5120 the log continues with values from kernel patch routines and kexts. This is for developers.

Running OSX on Unsupported Hardware

Actually, the whole book is about THIS. I will tell you partly here, starting from the question.

Unsupported BIOS. Of course! We are talking about Hackintoshes. And in First of all, this is the data in the DMI, which contains the manufacturer's name (should be Apple inc.), model and serial number, the numbers and letters of which are not random, they mean something, in particular the model and date of manufacture. In the simplest version, since the time of Netkas, everyone was given the model MacPro3.1, and a certain serial number, one for everyone, which worked. Now Clover, having analyzed the hardware, offers dozens of options that are functional. However, it is recommended to generate your own serial numbers, or maybe take a model different from default models.

Unsupported processor. Yes, different versions of MacOS support different CPU kits, and your processor may not be supported. Here is a table for old systems:

CPU name	CPUID	10.4	10.5.8	10.6.3	10.6.8	10.7.2	10.7.5	10.8.5	10.9.5
Yonah	0x0006E6	1	1	1	1	1	1	0	0
Conroe	0x0006F2	1	1	1	1	1	1	1	1
Penryn	0x010676	0	1	1	1	1	1	1	1
Nehalem	0x0106A2	0	1	1	1	1	1	1	1
Atom	0x0106C2	0	0	0	0	0	0	0	0
XeonMP	0x0106D0	0	0	0	1	0	0	0	0
Linnfield	0x0106E0	0	0	1	1	1	1	1	1
Havendale	0x0106F0	0	0	1	1	1	1	1	1
Clarkdale	0x020650	0	0	0	1	1	1	1	1
AtomSandy	0x020660	0	0	0	0	0	0	0	0
Lincroft	0x020670	0	0	0	0	0	0	0	0
SandyBridge	0x0206A0	0	0	0	1	1	1	1	1
Westmere	0x0206C0	0	0	0	1	1	1	1	1
Jaketown	0x0206D0	0	0	0	1	1	1	1	1
NehalemEx	0x0206E0	0	0	1	1	1	1	1	1
WestmereEx	0x0206F0	0	0	0	1	1	1	1	1
Atom2000	0x030660	0	0	0	0	0	0	0	0
IvyBridge	0x0306A0	0	0	0	0	0	1	1	1
Haswell	0x0306C0	0	0	0	0	0	0	1	1
IvyBridgeE5	0x0306E0	0	0	0	0	0	0	0	1
HaswellMB	0x0306F0	0	0	0	0	0	0	1	1
HaswellULT	0x040650	0	0	0	0	0	0	1	1
CrystalWell	0x040660	0	0	0	0	0	0	1	1

and so on

That is, support for Yonah and XeonMP is discontinued; the newer the processor, the newer the system required; Atom was never supported, although it looks like a regular Intel processor. Table is outdated, look at the XNU sources. Skylake, for example, is supported in 10.11.6 and higher.

When you start the system on an unsupported processor, you get a kernel panic. To prevent it, use the KernelCpu=true patch. It simply replaces the panic call with empty statement, and

everything continues to work. How correct is that? Well, at least it works! In new revisions of Clover I made a patch FakeCPUID=0x010676. Or other numbers suitable for your system, and close to your processor (approximately the same generation, for example, Atom should be replaced with Penryn, or even Conroy). The substitution occurs in the kernel at the level of calling the get_cpu_info() procedure and thus will affect those kexts that access the CPU for information, instead of calling CPUID themselves. For example, this is how AppleIntelCPUPowerManagement.kext works, and this patch affects it. Now, with revision 5155, this patch has moved to SMBIOS, which depends on the system. That is, now for one system we make one FakeCPUID, and for another other numbers, or do not do it at all, since the system already supports such a CPU. Example:

```
<key>SMBIOS_mav</key>
<dict>
	<key>FakeCPUID</key>
	<string>0x010676</string>
```

Unsupported video card.

Intel. Supported: GMA950, X3100, HD3000, and higher. Alas, no substitutions help. Each option has its own set of patches, and if the video card is different, then in the best case you will have a picture without the ability to change the resolution, and without any 3D effects. In principle, you can live, but the inability to calibrate the screen color does not suit me, because it is impossible to work even with photos on such a computer. In the 10.14 system, they are supported starting from HD4000. But HD4000 does not support 10 bits / color, but Skylake HD530 already does! In the Monterey system, support for HD4000 has ended, enthusiasts start a separate patch. Another observation. The video card does not work with Celeron and Pentium processors, except in VESA mode and software OpenGL.

Nvidia. 7300-7600 cards are supported only up to system 10.7.5 in 32-bit mode. It is probably useless to talk about older cards. There are some questions about Fermi 4xx/5xx series cards. They should also be classified as partially supported, and only up to system 10.11.6. In the case of Nvidia, also check the AppleGPUPowerManagement kext, it may also contain the ID of your or a similar card. See below. For systems from 10.12 to 11.x, only Kepler, the GK family, works natively. These are GTX 6xx-7xx cards, but not all of them, among them there are also Fermi, then it's a bummer. For Pascal and Maxwell cards , you need a WEB driver, which only exists up to system 10.13.6. There are no web drivers for systems 10.14 and higher, that is, Maxwell and Pascal are completely out of the question. Newer cards are not supported in any way. For Nvidia to work, a trick with MacModel or with BoardID substitution is required (for example, Mac-27ADBB7B4CEE8E61). On the other hand, someone starts old Teslas in Mojave, apparently Apple left a loophole for its old computers. For Monterey, Nvidia Kepler is started with a special patch that includes outdated kexts and frameworks, while this method works.

ATI/ AMD. The huge story. And about how I started Radeon9000IGP, and about dong kext for X1500, and about Callisto kext, and about complex recipes for patching connectors for modern cards. Look in this book. A lot has been done for Radeons, search, read, don't be dummies! Radeons of the HD6000 series and below do not work on systems with 10.13 and higher. They do not have Metal support, and therefore the drivers in the system are incomplete, or do not turn on at all. RX5700 cards work only starting with Catalina. For Mojave, for example, the list of possible cards is: 550-590 and Vega. Older cards on the GCN architecture also work, but not all and not out of the box.

To all this I want to add one more trick. Apple drivers enable the video card depending on the model, but Clover allows you to use one model, and at the same time use the BoardID of another model that came with a similar video card. The list is as follows (the model is for reference, use only the BoardID):

Nvidia Kepler → Mac-27ADBB7B4CEE8E61 iMac14,2

Nvidia Kepler + Intel → Mac-4B7AC7E43945597E MacBookPro9,1 (дисплей на нвидия)
Radeon R9 270X → Mac-42FD25EABCABB274 iMac15,1
Intel Iris Pro → Mac-031B6874CF7F642A iMac14,1
Radeon Vega + Intel → Mac-63001698E7A34814 iMac19,2
Radeon RX570 + Intel → Mac-AA95B1DDAB278B95 iMac19,1
Radeon RX5700 + Intel → Mac-AF89B6D9451A490B iMac20,2
Radeon RX → Mac-7BA5B2D9E42DDD94 iMacPro1,1
In the last option, in particular, you can disable the built-in in the BIOS, and in the system QuickLook will work using Radeon. Another news, some group of developers made a NootedRed driver for Vega cards built into the AMD chipset. And a NootedRX driver for the RX6700 card, which had not worked in any way before.

Now here's WhateverGreen from vit9696 - "driver for all video cards, just install the latest version, as well as the latest version of Lilu, and don't think about anything. In Clover, you need to disable everything related to video cards." But I don't play like that! If you don't want to figure it out, please use it. For others, we'll go through step by step what you need to start a video card. Personally, this driver has never helped me or improved anything.

Sound card. Professional cards usually have drivers for Mac. Chipset codecs of the HDA standard are supported by all with the VoodooHDA kext. The native AppleHDA kext does not support any codec on the market. There used to be ALC885, but it is no longer found. But hackers have developed a method for patching AppleHDA so that it supports almost any necessary Realtek chip (that is, ALCxxx). Clover helps to correct DSDT for this kext, and offers methods for patching the kext on the fly. What exactly to patch and how to read on the forums. HDMI Audio works with DSDT patches provided in Clover, however, it does not work with some AMD cards. But in 10.13+ systems, a new driver AppleGFXHDA.kext has appeared. If you specify that VoodooHDA is attached only to the HDAS device, then AppleGFXHDA will be attached to the HDMI device on the AMD or Intel video card, and there will be sound on the TV. I have

Network card. Firstly, Apple drivers support a whole range of chips. Secondly, programmers have learned to write kexts for network cards, and drivers exist for most known cards. In some cases, it is enough to make a FakeID for a card so that it gets into the list of supported native drivers, but in most cases a separate kext is needed.

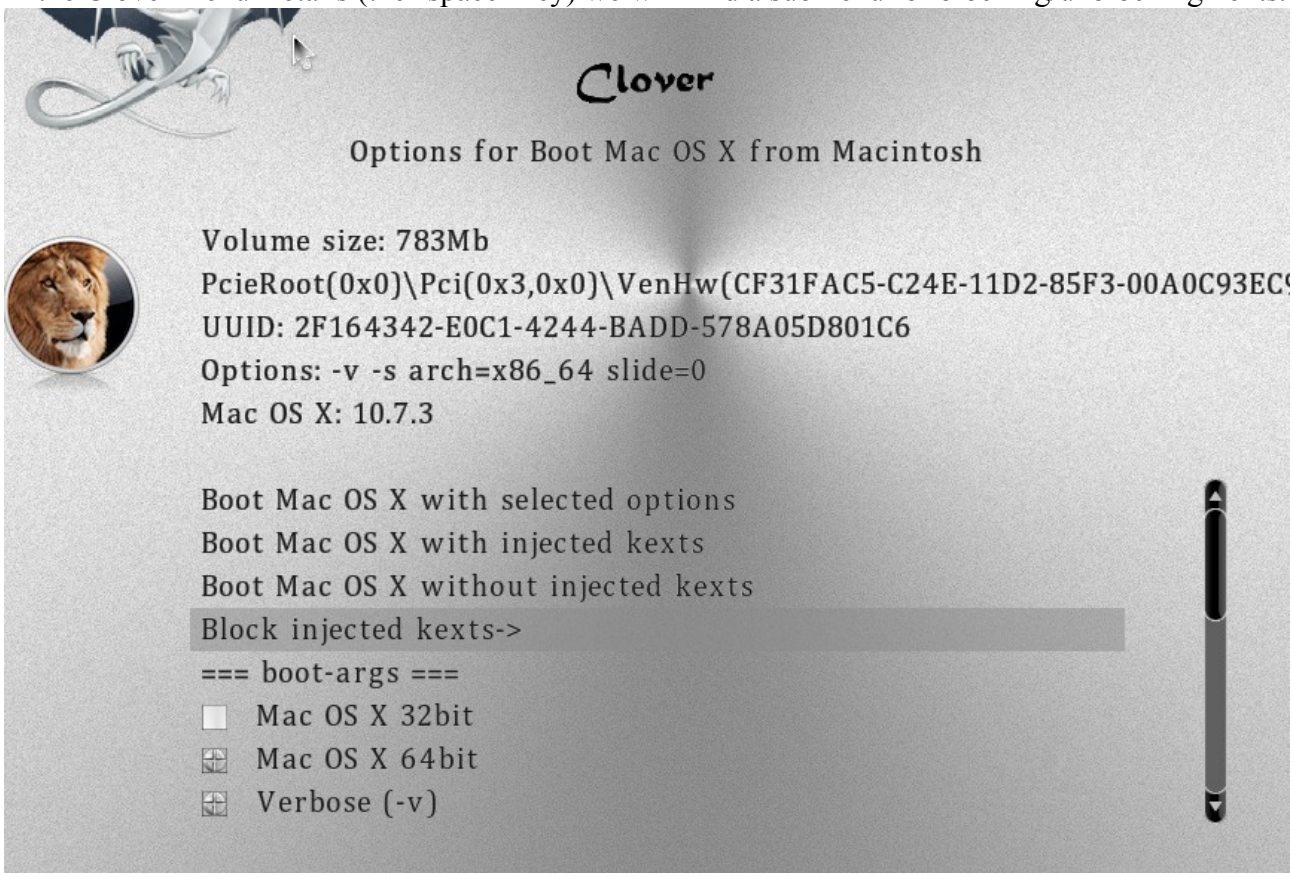
WiFi. But here everything is very sad. Some Broadcom, Atheros and Ralink. Look on the forums for information about each specific model. Intel is not possible at all. Clover can help with FakeID, for example, in my version of replacing Boadcom4315 with the supported 4312. And also Atheros with neighboring numbers. Since 2020, a driver for some Intel WiFi cards has appeared on the network. There is a chance! In the Monterey system, the list of supported cards has again been reduced to Broadcom 94360. In Sonoma, this Broadcom has also fallen off (for now?). And even with the help of OCLP, it no longer starts in the 14.4 system.

Kext blocking

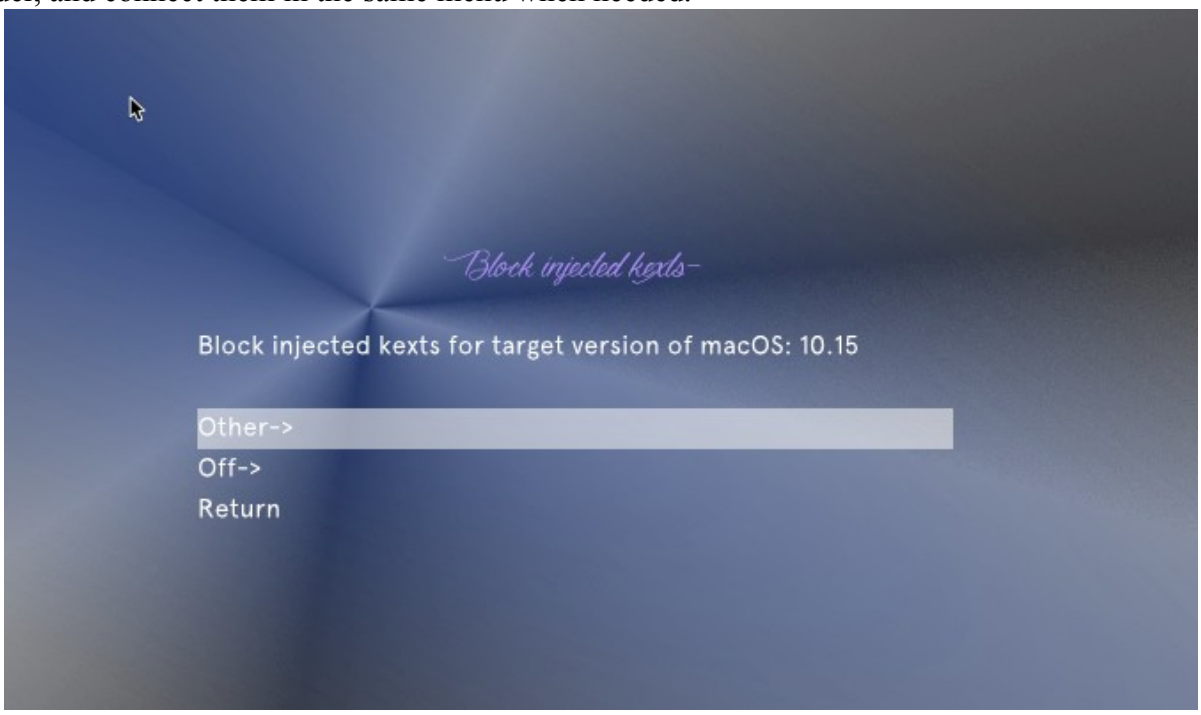
I happened to install the Geenna.kext kext in the SLE system folder. There was a panic on the screen, after rebooting this kext was loaded first, and then panic. So what to do now? It needs to be removed, but there is no other system on this computer yet. For this purpose, an additional function has been introduced in Clover: in the Options Menu, in the third line, enter Block kext: Geenna and calmly load the system in single user mode (space on the system icon). The kext will not have time to load, because it is blocked. In this text mode,

```
fsck -fy
mount -uw /
rm -r -v /S*/L*/Ex*/Geenna.kext
reboot
```

A reboot is necessary here, otherwise the kernel will still load this kext as the next step, and there will be panic again.
The new Clover has a new method: put all your kexts in the EFI/CLOVER/kexts/Other folder. And in the Clover menu Details (the "space" key) we will find a submenu for blocking/unblocking kexts.



Starting with revision 5052, it became possible to store kexts that are not yet needed in the Off folder, and connect them in the same menu when needed.



In this case, kexts in the Other folder are connected by default, and those in the Off folder are disabled, but are visible to Clover for connection.

And now there's a new problem: you need to replace the system's kext with your own, which, of course, is also from Apple, but from a different version of the system. We'll put our own in Other, but how do we block the same native one? A section has been made in Clover 5172 for this purpose. config.plist→KernelAndKextPatches→KextsToBlock,

in which we write an array of kexts to block

```
<key>KextsToBlock</key>
<array>
    <dict>
        <key>Comment</key>
        <string>Allow IOSkywalk Downgrade</string>
        <key>Disabled</key>
        <false/>
        <key>MatchOS</key>
        <string>14.x,15.x,26.x</string>
        <key>Name</key>
        <string>com.apple.iokit.IOSkywalkFamily</string>
    </dict>
    <dict>
        <key>Comment</key>
        <string>Allow Block AppleEthernetRL</string>
        <key>Disabled</key>
        <true/>
        <key>MatchOS</key>
        <string>25.x,26.x</string>
        <key>Name</key>
        <string>com.apple.driver.AppleEthernetRL</string>
    </dict>
</array>
```

Slot name (AAPL,slot-name)

It's mostly cosmetics, although there are claims that it's a must in some in those cases.

Where does the system get the slot name? The old method was to inject it via the _DSM property "AAPL,slot-name", but this is a completely wrong method, because it treats the effect instead of the cause. This property is set by the AppleSMBIOS system kext based on the ACPI property _SUN and DMI tables. That is, _SUN specifies an ID in the range of 0-255, which is used to find the SMBIOS table type 9 with the corresponding ID, where the slot name and its other properties are taken from. See the chapter on filling the config, section SMBIOS->Slots

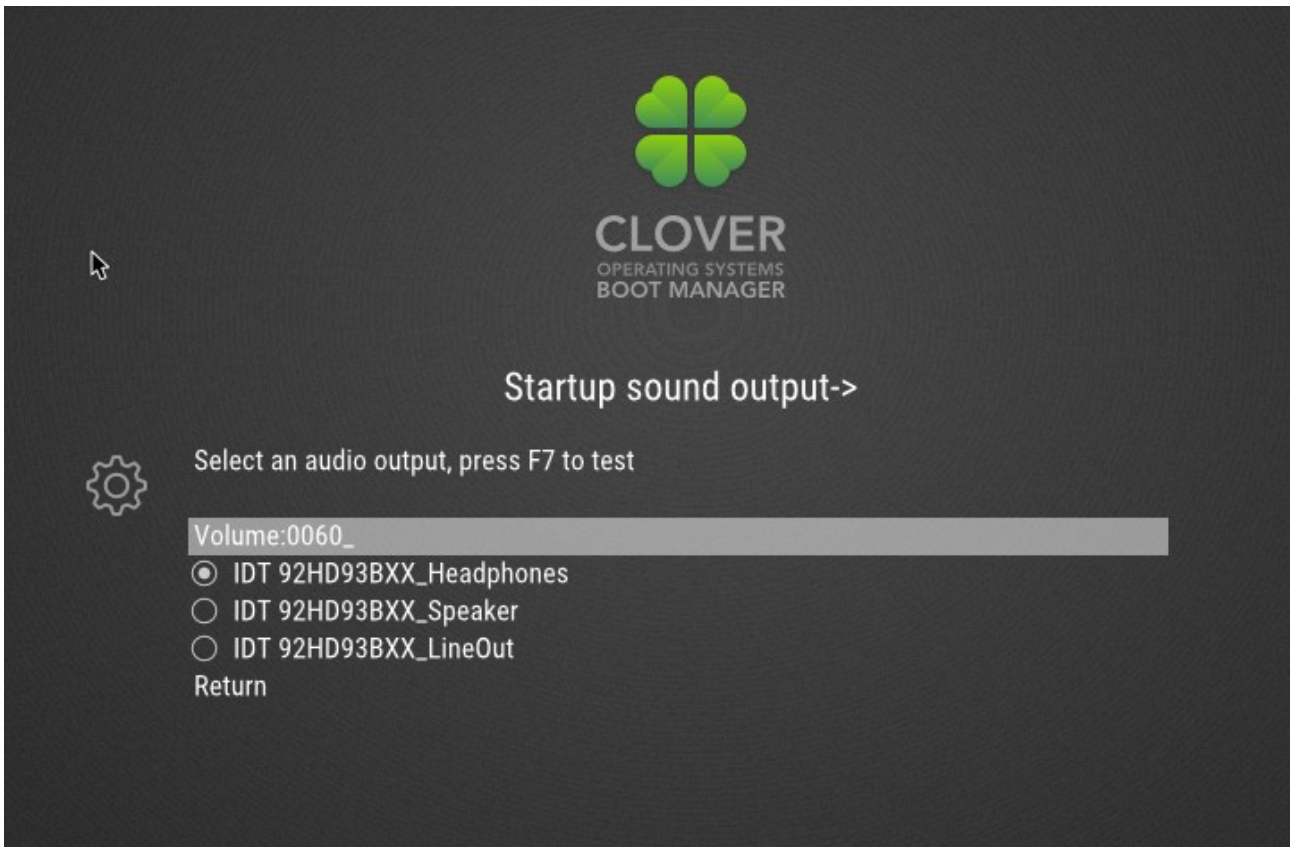
HDMI sound

Everything you need for AppleHDA was investigated by Toleda, but not everyone will want to search his explanations in English. I made DSDT patches, with his participation, to get as close as possible to his result. Basically, there are two options for HDMI devices.
1. On an external ATI or Nvidia video card. In the system, it is listed as a sound device of class HDA = 0x0403, and is serviced by the same sound driver. It is only necessary that both the video card and HDMI have the same property "hda-gfx=onboard-1". Or it may not work! Unsupported device.
2. There is an HDMI connector on the integrated Intel card, but there is no such device, the sound from the chipset HDA is used. In this case, you need to register in the config

```
<key>Devices</key>
<dict>
        <key>UseIntelHDMI</key>
        <true/>
```
In this case, the sound from ATI or NVidia will become "onboard-2". For DSDT, the necessary fixes: FixDisplay_0100, FixHDA_8000, AddHDMI_8000000
3. The option that the built-in is used only for IQSV. Then (iMac18,3) In the HDEF device there is the No-hda-gfx property In IGPU there is nothing (built-in for IQSV) In GFX0 (which is Radeon) there is hda-gfx="onboard-1" And in HDAU it is the same. Clover does not do this automatically yet, use the Properties array. Note that all these additional properties are
needed only for AppleHDA. Driver VoodooHDA does not need external hints. I also noticed that the system can confuse which HDMI port to send the sound to. As a result, we see that the device is there, but there is no real sound. Someone investigate! I do not have the appropriate hardware.
2022. Everything said is out of date! For the Nvidia Kepler video card, the VoodooHDA driver itself knows what to do, and HDMI sound works with it. For AMD Radeon RX4xx cards and higher, VoodooHDA does not work, but the AppleGFXHDA.kext system driver works, which only requires that VoodooHDA does not cling to this device. This can be easily achieved by replacing IOPCIClassMatch in VoodooHDA with IONameMatch=HDAS, for example, and making sure that this is the name of the sound device in DSDT.

Computer startup sound

This is an invention of Goldfish64. He wrote an EFI driver for HDA sound, and made utilities for setting up sound, testing and making a dump of the sound codec.
https://github.com/Goldfish64/AudioPkg
But he inserted the sound at the time of boot.efi startup, intercepting its call by the bootloader. That is, the idea is for it to work not only with Clover, but with any EFI bootloader, without touching its internal codes. But I am more interested in the sound working before entering the Clover interface, or even while I am walking through its menu. The license is open, so I remade it to suit my wishes. All settings and tests can be done by Clover itself, with its graphical menu, and I actually included the driver in the Clover repository so that it does not get lost, and so that in the future it can be improved without asking the author, who may disappear, not today, then tomorrow. So, for the sound to work, you need to:
1. Use Clover revision 4871+. It worked in previous revisions, but with bugs.
2. Put the AudioDxe.efi driver in the EFI/CLOVER/drivers/BIOS or drivers/UEFI folder, respectively. Or both. This driver, supplied with Clover, already differs from the author's original, so far it is not essential, but I would recommend my version.
3. Put sound files with the names sound.wav and sound_night.wav in the themes you use. Thus, the startup sound depends on the selected theme. sound_night.wav is not required, if it is not there, then sound.wav will play at night. These files must be in RIFF/WAV format, 2 channels, 16bit little endian unsigned int, the sampling frequency can be 8, 44.1, 48 kHz, the file size depends on this. The sound itself supports 44 and 48 kHz, and even more. For compactness, I allowed 8 kHz, and Clover converts such a file on the fly to 48 kHz. The quality is inevitably lower, but for this case it is not really needed. But the sound of this format was packed directly into Clover, and it is used for output testing.
4. Go to Clover's interface, “Startup sound output”, and test which output will play



In the first line we adjust the sound volume from 0 to 100. These are percentages, there is no more than 100. The value 0 means that the sound will not be played. That is, it is not that it screams with its mouth closed, but that it does not attempt to meow. By the way, I do not know whether the scale is linear or logarithmic. The following lines are combined from the chip model and its output. If you have several sound cards, as often happens in addition to the built-in one there is also HDMI, then you will see everything in this list, with all their outputs. Choose, press F7, listen. After exiting this menu, the selected setting will be saved in NVRAM, including the emulated one in the variables Clover.SoundVolume, Clover.SoundDevice, Clover.SoundIndex. Here I have a difference from the goldfish, it saves the settings in a private area with its UUID, which is impossible for emulated memory, for legacy Clover. My settings will be visible from the system, can be deleted or modified from the system, and the Clover prefix ensures that there is no conflict with Apple's interests. At this stage, Clover will read the fish settings if it doesn't have its own yet, but in the future it will only use its own. On the next reboot, you will hear the sound before loading the Clover shell, but after the inscription ... scan entries ... These settings are not taken out in the config, there is no need. In any case, you should first test, and therefore write the settings to the NVRAM. But in the config there is one setting left over from the test period. PlayAsync = true. If false, then while the sound is playing, nothing works. It seems to be the same on the real one. If true, then the sound is played in the background, without interfering with everything else. I put a long sound file and listened to the music. Clover's GUI appeared, the music is playing, I went to the menu and chose to boot in Verbose. The music is playing. I pressed "load system" and look at the messages: boot.efi has finished, the music is playing, the kernel has started, the music continues play! And only after a few loaded kexts she shut up, it was probably VoodooHDA, which reinitialized the sound chip. Didn't find any problems in the loaded system. In revision 4862, it was impossible to use asynchronous sound, it hung. In revision 4871, the problem was solved, and now you can use asynchronous sound by default.

NVRAM, iMessage, multiboot

Question about the system's use of non-volatile memory NVRAM using functions GetVariable() and SetVariable() were actually raised by me back in 2010 http://www.projectosx.com/forum/index.php?showtopic=1504
Then I tried to implement work with it in Chameleon in my own branch, but I didn't get any support. Nobody needed it, although my argument about the "Boot Disk" control panel remained irrefutable. Then the gurus explained to me that this is in the DUET bootloader, so, having started the Clover project based on DUET, I first of all set the goal of providing this functionality. These functions are in Chameleon, but they are made very simply "return Unsupported", so so that the system running with Chameleon does not panic and simply does not respond to the call of these functions. This worked for a while, with the exception of the StartupDisk panel. But the iMessage service has already refused to work in this version. No substitution or emulation worked. I bow my head to Meklort, who within a month still came up with a way to make such functionality in Chameleon, using the FakeNVRAM.dylib module and some mother. What is meant by NVRAM health? If the system wants to save some variable until the next reboot, it writes it to NVRAM using the SetVariable(...) function. We can also save our variables using the nvram utility:

```
sudo nvram MyVar=qu-qa-re-ku
```

after reboot this variable should be known to the system using the command readings

```
nvram MyVar
```

How does Clover ensure the functionality of this service?
1. For legacy booting, EmuVariableDxe functions are used. This, of course, is not a real non-volatile memory, because legacy Clover is designed for those computers that do not have such memory at all, as well as their own EFI with the necessary services. This driver simply writes variables to memory, but this memory is available for use by MacOSX in its native interface. When the system is shut down, the rc.shutdown.local script is called, which saves all this memory to the nvram.plist file in the root of the system disk. Clover finds this file at startup and writes all the variables from there back to the RAM, emulating NVRAM. The method is incomplete, because only variables from AppleBootGuid are saved in this way, however, this is enough to select the Start Disk. Starting with revision 5154, this problem has been solved. Made its own utility nvram2, which saves variables with different GUIDs in the file nvram.plist, which may be needed by the system during reboots. You just need to update the Clover scripts.
2. For UEFI boot, we rely on our own service VariableDxe, which is provided in OEM UEFI. In revision 2837, Dmazar corrected the work with this service, so for most users it now works natively. For those for whom it still does not work, there is an emulation driver EmuVariableUEFI, which works similarly to the legacy driver, and also requires scripts and the file nvram.plist. New times have come!

Again, vit9696 corrected the OsxAptioFix driver so that the hardware HBRAM worked, but on the new chipsets 360, 390 and this does not work. The change is presented by the driver OsxAptioFix3Dxe, and vit9696 himself offers a more advanced version of AptioMemoryFix, included in the Clover repository. And now this functionality is in the OpenRuntime.efi driver. (everything is renamed so that people forget about Clover and remember about Open…)
EmuVariable in both cases is not a full-fledged emulation, for example, panic.log is not saved, simply because the script does not have time to work. The boot0082 variable, necessary for hibernation, is also not saved, but we got around this problem in other ways. But the presence of panic.log, a long-time dream of hackintoshers, remains the prerogative of Clover with a real NVRAM. And, again, hibernation in mod 25 requires saving the encryption key online, that is, only with a real NVRAM.

iMessage is an instant messaging system from Apple itself. Since December 2012, the rules for registration and use have changed, and all hackintoshers have been left out of work. It would have worked with Clover if we hadn't made a mistake in the number of digits in September, having

sorted out the iCloud service, we should have left 17, but we left 12. We realized the mistake only in January, and thus the Chameleonites also realized what was going on, only they didn't have NVRAM, without which all this was impossible. Namely, for successful registration of iMessage, it is necessary to write the ROM and MLB variables to NVRAM, unique for each computer, and the computer is identified by its HardwareUUID, which, accordingly, must also be unique. For complete dummies, I made the generation of these properties based on DMI data, but also recommendations to enter the corresponding values in config.plist, for those who understand a little more. It turned out that the iMessage service is paid, and the user needs to register their account in the app store, from which Apple can write off $ 1 to check that the bank account is active. This also implies the need for a unique account. Do not use someone else's ROM, MLB and UUID, and especially someone else's bank card. When everything is yours, the ROM has 12 digits, the MLB has 17 digits, the UUID is non-zero, and all this is unique, the account is linked to a valid account that has money, iMessage will work. And do not listen to any speculation about en0, formatting partitions and the like. I have listed all the conditions.

Boot Disk is a service that allows you to select in the control panel, in what system we want to reboot, press restart, and just leave.

The computer will do everything itself. This service requires that the disk be marked in GPT. So you can switch between 10.9 and 10.7, for example.
Remember the general rule: **dynamic data has priority over static. Data from NVRAM has priority over data from config.plist**

Using multiple configurations

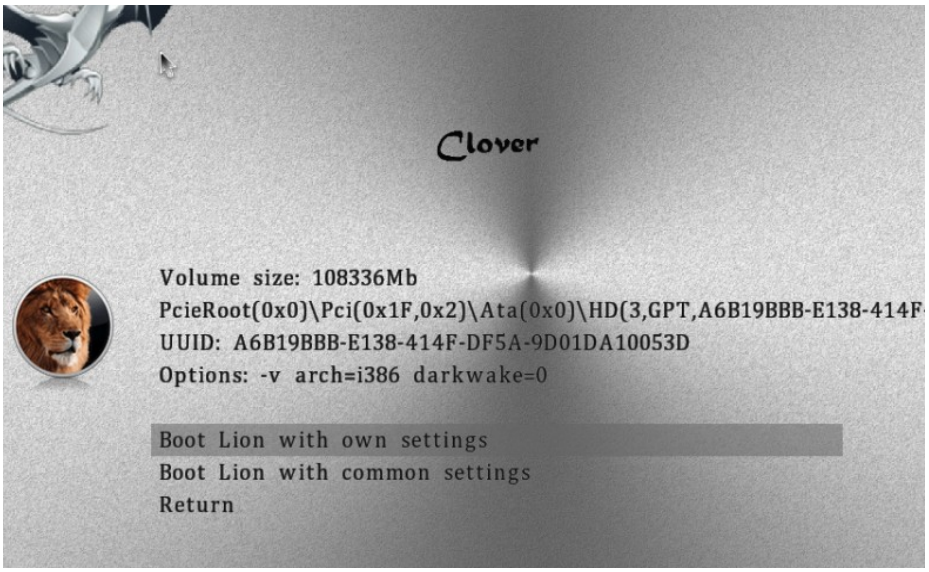
Possible problem: you have several systems, but these systems should boot with a different set of patches written in the config, for example, the definition of the Radeon framebuffer in the new system is different from the old one. But how to do this if there is only one config in Clover? Starting with revision 3266, this option is provided.
Here is such a config

```
<key>GUI</key>
<dict>
	<key>Custom</key>
	<dict>
		<key>Entries</key>
		<array>
```

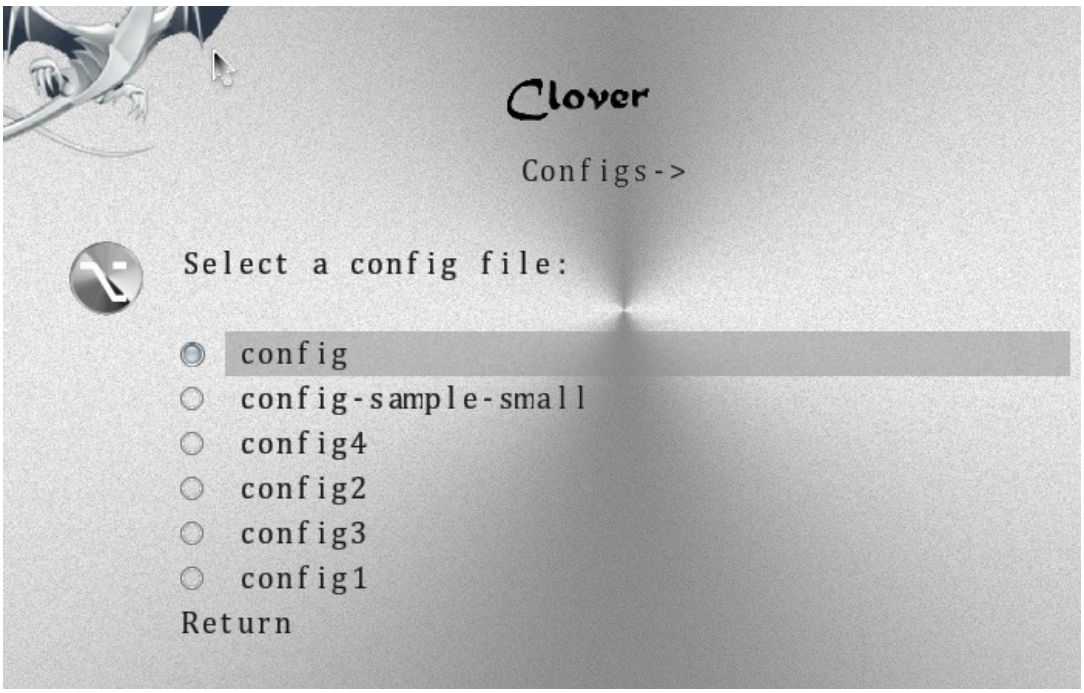
```
                        <dict>
                              <key>FullTitle</key>
                              <string>Lion special</string>
                              <key>Settings</key>
                              <string>config-special</string>
                              <key>Volume</key>

      <string>EE9CCC69-EE7F-358F-B120-BCD07AD78282</string>
                              <key>SubEntries</key>
                              <array>
                                    <dict>
                                          <key>FullTitle</key>
                                          <string>Boot Lion with own
settings</string>
                                          <key>CommonSettings</key>
                                          <false/>
                                    </dict>
                                    <dict>
                                          <key>FullTitle</key>
                                          <string>Boot Lion with common
settings</string>
                                          <key>CommonSettings</key>
                                          <true/>
                                    </dict>
                              </array>
                        </dict>
                        <dict>
                              <key>FullTitle</key>
                              <string>Lion default</string>
                              <key>Volume</key>

      <string>EE9CCC69-EE7F-358F-B120-BCD07AD78282</string>
                              <key>Type</key>
                              <string>OSX</string>
                        </dict>
                  </array>
            </dict>
```

Here is what is described: we have assigned our own main menu items (Entries) with the names "Capitan special" and "Capitan default". The second item, as usual, allows loading the system with a common config.plist, taking into account the changes made in the Options menu of Clover. The first item creates a new icon for the same system, but it will be loaded with a different config-pecial.plist, as specified in the Settings key.

But that's not all. By pressing the spacebar we will enter the launch menu, and here we will find our inputs, registered as SubEntries



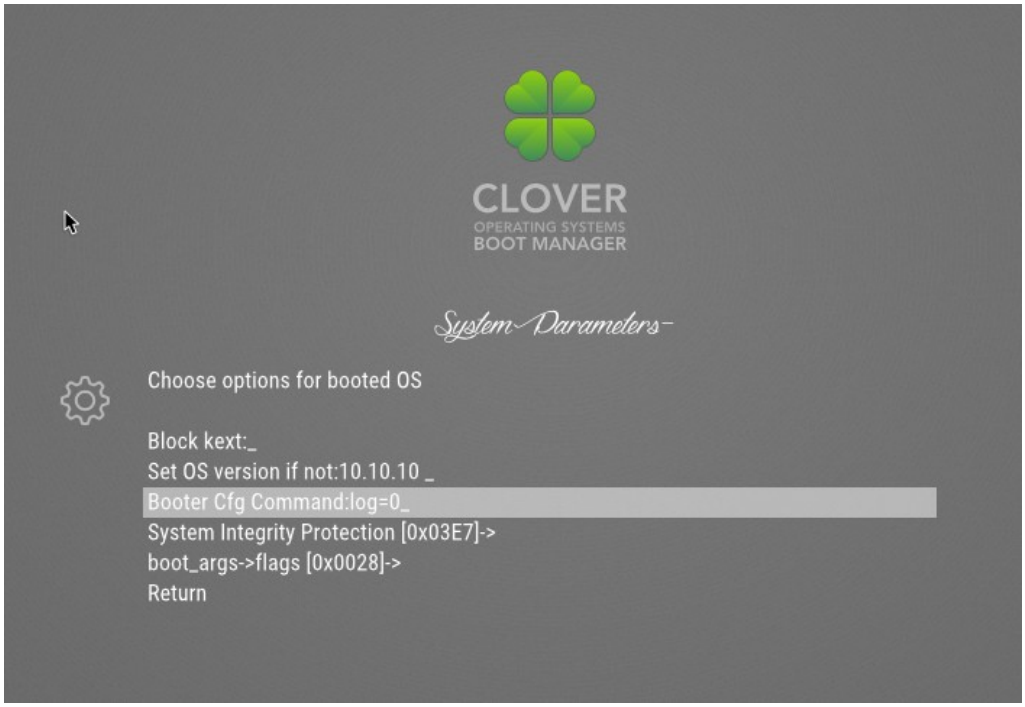
The second item in this menu means rejecting a special config in order to use general.
It is clear that since the special config is connected after Clover has been launched, then the Boot and GUI sections are no longer needed in it, they can only be in the general config. I personally used this opportunity to test new configs for launching the captain, having one verified worker. The worker is special, and the general config is experimental, because that the general config can be changed through the menu, and the special one is used as is.
Now in Clover there is an option to switch configs directly in the menu. Here picture:



The limitation of this method is that the Boot section is not changed. It also does not change KernelAndKextPatches. The new config will take effect after exiting to the main menu. Here the confusion arises, why change the config at all, if not for KextPatches? Alas, it does not work, the reason is somewhere deep in Clover's algorithms. On the other hand, write all the patches in one config, the main one, and in the Clover interface you can enable or disable them with a checkbox. The benefit of different configs is perhaps in different SMBIOS sections. Starting with version 5153, it became possible to have several SMBIOS sections in the config and choose which one you will boot from. Moreover, the selection of the SMBIOS_xxx section can be done automatically depending on the version of the operating system you are booting from. Yes, the tables are filled after the system is selected, so no miracles, just work. For details, see the paragraph on Configuring the Hardware→ SMBIOS.

How to make boot.efi not spam too much on the screen?

Enter this menu



and write log=0 there. Other values are possible, researched by vit9696 log=value,
output direction
1 — AppleLoggingConOutOrErrSet/AppleLoggingConOutOrErrPrint (classical ConOut or StdErr on failure)

2 —AppleLoggingStdErrSet/AppleLoggingStdErrPrint (StdErr or serial?)

4 —AppleLoggingFileSet/AppleLoggingFilePrint (BOOTER.LOG/BOOTER.OLD file on EFI partition)

debug=value
1 — enables print something to BOOTER.LOG (stripped code implies there may be a crash)

2 — enables perf logging to /efi/debug-log in the device three

4 — enables timestamp printing for styled printf calls

level=value – error level
kc-read-size=value - log size
For our
purposes, log=0 is sufficient, which is the default in Clover..

USB setup

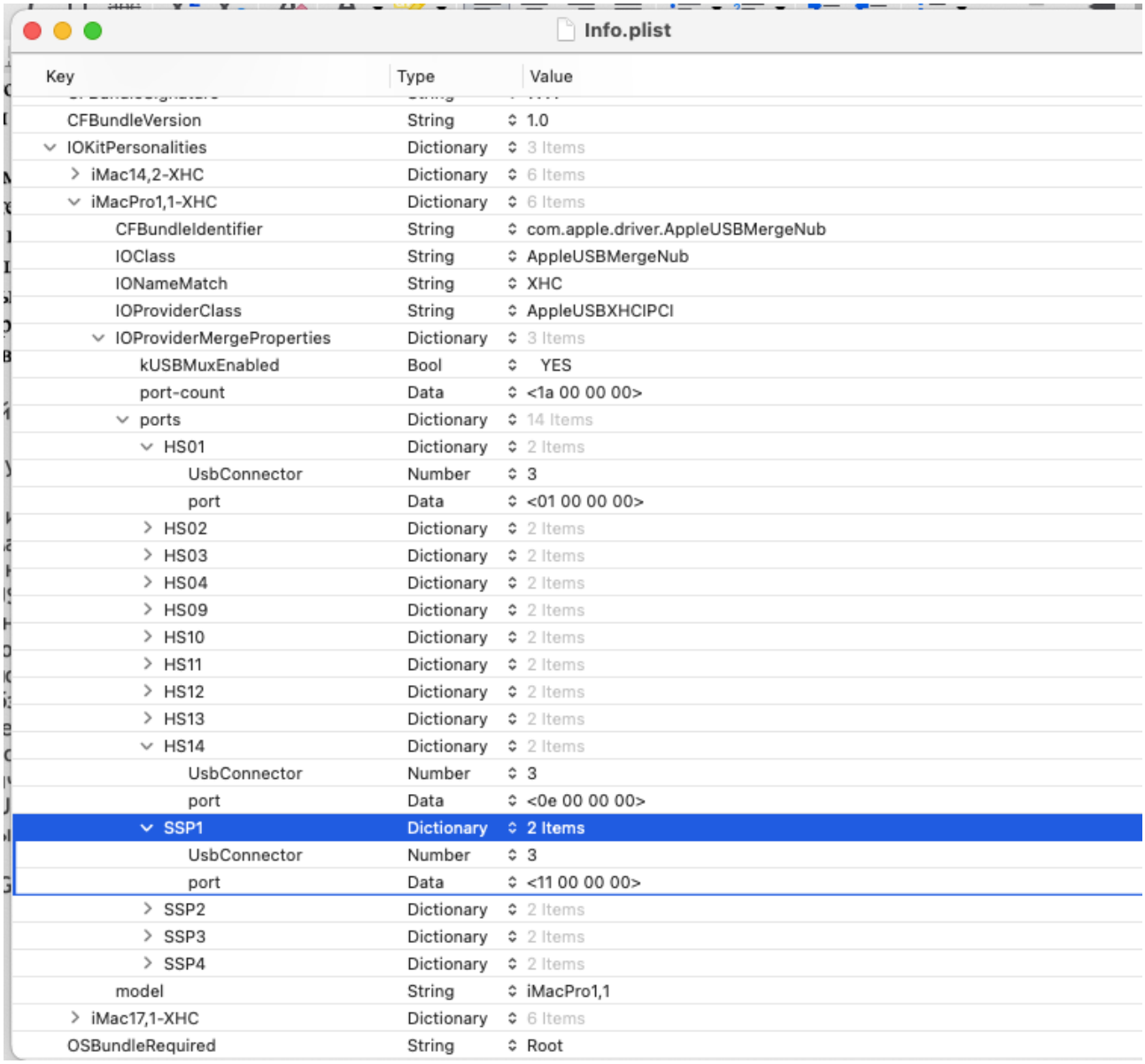
This topic is very popular, most beginners start with "starting USB". In fact, the complexity here is exaggerated. Firstly, in the Devices section there is a USB subsection, see above what is configured there. Secondly, I still recommend not to ignore DSDT fixes. But I will consider some common questions separately.
1. The computer goes to sleep, but then wakes up, and in the wake-up log the reason is USB. Well, this is logical if there is a device there that lives its own life, a video camera, for example. The flag that this port is internal, not external, helps. It is set in DSDT / SSDT or in LegacyUSB.kext, see below.

2. After sleep, the flash drive falls off. Claims to macOS, because some flash drives fall off, others do not. Advice do not leave it there.
3. Not all USB3 ports work. The thing is that macOS has a limit of 15 ports per controller, as described above in the KernelAndKextPatches section. Here I will tell you how to make LegacyUSB.kext, the idea is not mine, here is the linkhttps://applelife.ru/threads/nastrojka-usb-v-10-11-i-novee.627190/page-3#post-537459, the author of the text is Andrey1970. I confirm the operability of the method. I can't give the entire template, so only this:

---

we need to determine which ports are not used in order to delete them and allow those we need to get into the "Top 15". My first 15 ports are HS01-HS15 (servicing USB2 devices) and there was just enough space for them in the IOReg. We take a USB2 device (for example, a USB2 flash drive) and start inserting it into all the ports one by one (USB2 and USB3, and don't forget about the ports on the motherboard that are under the cable, so as not to redo them later, because you might need them someday), while looking in the IOReg at which HS** ports the USB2 device is connected to. We write down the ports. Having done this with all the physical ports and making sure that they are all connected, we delete the unused HS** from the Legacy. Now the "Top 15" includes SSP* ports (servicing USB3 devices), but perhaps there wasn't enough space for all of them. No problem, temporarily remove several HS** ports from the top of the list (make a backup of Legacy), so that the total number of registered ports is no more than 15, and then all SSP* ports will get into IOReg. Next, take a USB3 device and start inserting it into all physical USB3 ports (only into USB3, and forget about the USB3 sockets for the cable), similarly, write down which SSP* the device is attached to. A USB3 device inserted into a USB3 port should only attach to SSP*, and not to HS**. Remove unused SSP* ports from Legosi, restore the temporarily removed, but necessary, HS** ports. Now there is enough space for all ports. For my MB GA-Z87X-UD4H, I needed to register 14 ports in Legacy.

I got it like this, and it works from Captain to Sonoma. I don't have any built-in devices, there you would have to set UsbConnector = 255, just so that this device doesn't wake up the computer. By the way, all this doesn't make sense for IviBridge and older, there are fewer ports there than the limit in MacOS anyway. If you change MacModel from time to time, then pay attention that I have three identical sections, only with different models. I excluded ports HS05-HS08, I don't need them, but I have all USB3 ports working. Here is a screenshot:



Antidortania

Newbies come to the forum and say with a completely serious face that they did everything exactly as Dortania did. Why doesn't it work? I'll tell you about some of the mistakes in this guide, maybe the author will see and correct it. I don't know if it's worth telling in a more public place place.
Mistake #1 — using OpenCore. Modern Clover contains modern OpenCore, what difference are you going to see? However, yes, there is a difference. If the author wants to use OpenCore, then that is his right.

Anti-Hackintosh-Buyers-Guide

CPUs

Much has been said about AMD CPUs, except for the most important thing, hackintosh on them is all- they do, earlier by means of a special kernel, and accordingly no updates were possible. Now there is a set of binary kernel patches that provide automatic kernel patching after an update. They even launch Sonoma. I can't say anything about bugs in the system running on Ryzentosh. Maybe it's true, maybe it's already outdated. I'm not interested in this topic. Is Intel X79/LGA 2011 listed as poorly supported? Why all of a sudden, is it made on it? MacPro6,1! Recommendation: Ivy Bridge through Comet Lake are properly supported in macOS. At all-even older processors work, and with modern patches even Penryn can be used. On the other hand, IvyBridge is also outdated, it will not work in Venture without patches. And if you take into account the integrated graphics, then Hazwell is already problematic. And newer processors, like Alder Lake, are quite functional.

GPUs

I have already written about the compatibility of different video cards with MacOS, I will not repeat myself. My the information differs from Dortania. But here are the specific objections. The vendor claims that you can't use XFX and PowerColor. And also MSI if NAVI, while MSI Polaris and Vega are good. All nonsense! XFX RX570 works like a clock, all outputs (with the latest firmware, at least), speedstep, I don't even know what to complain about. But the opposite is true MSI RX570 has very small hysteresis in fan speed from temperature, the rating is worse than XFX. Well, how should we treat Dortania's advice now? By the way, I also had PowerColor. Normal card, just morally outdated. It is claimed that the RX540 and 550 do not work at all. This is not true, although as advice that it is not worth taking, maybe it is possible to accept. The question of price.

Motherboards

Chipset. Sorry, but it's stupid to list them. Just like it's stupid to list the chipsets that not for sale. If the computer is old, then you do not choose it, and if you choose what to buy, then it will be the newest chipset at the time of purchase, and not at the time of writing this guide.

Audio. Almost nothing has been said. AppleALC requires selecting a layout from 1 to 100, by enumeration, rebooting each time, and accepting the first more or less working one, even if not everything has started. VoodooHDA is often configured automatically, but if you want more, then you need to seriously work on the config, but without hundreds of reboots. Creative and VIA chips are very poorly configured, and with AppleALC they may not work at all. He is right about the microphone, on some boards, and not only on AMD, the microphone can only be launched with VoodooHDA, because it can violate the narrow HDA standard.

Ethernet. In general, I correctly listed modern kexts to support different chips, I just forgot to mention my RealtekSL1000. And other network ones are somehow started, Aquantia, Attansic, Broadcom, Nforce, Marvell… USB. Nothing really said. But avoid VIA VLC! Although, actually supported.

NVRAM. Nonsense. NVRAM is on any UEFI BIOS, including x79, which is this blacklist.

Memory Maps and Protections. Again comes the blacklist of chipsets, and praises OpenCore with his picks. So what? Not take such motherboards? But there are solutions!

Storage

This is about supported and unsupported SSDs. I don't know. Maybe here correct lists.

RAM

In my opinion, if the computer works with this memory, Windows, for example, then MacOS will work too.

Coolers

A whole article, but I don't know why. I went to DNS and bought a cooler for the right processor.

Networking

Actually, we just discussed it. His list is not complete, neither for controllers nor for drivers.

Wireless

Well, why not a word about Broadcom? It's the card and Bluetooth #1 on Mac!

Power Supply, Case, Thermal paste

What? He advises which power supply to choose? A box? Thermal paste? From those that sold in the USA.

FAQ

Frequently Asked Questions.

Q. I want to try the Clover. From what to begin?

A. From reading this book.
P.S. It's strange to write this inside the book, but maybe these FAQs will end up outside its pages.

Q: Which version of Clover is best for my hardware?

A. The last one. Not even discussed.
P.S. Here is the bug log for some revisions, what was finally fixed:
3514: DDR4 support
3471: global bug with va_args usage
3362: SMBIOS bug when lines are duplicated in the original
3358: fixed calculation of the number of cores for many Xeons
Here is the bug log for some revisions, what was finally fixed:
3362: SMBIOS bug when lines are duplicated in the original
3358: fixed calculation of the number of cores for many Xeons
And so on... All this is not counting the amendments to the compilation and installation processes, to cosmetics and design, as well as support for new hardware and new OSes.

Q. It doesn't work.

A. You're a fool yourself.
P.S. Well, what else can I say?

Q. I installed Clover but I get a black screen.

A. The OS loads in eight stages (see Page 6). Please specify at which stage exactly it stops. And in your report, be sure to indicate "Installed using the installer with the following options selected." Then we'll talk. The most common errors: CsmVideoDxe doesn't work with some BIOSes, remove it; PatchVBios=Yes sometimes results in a black screen, try turning it off; Boot->Debug=true. Everything works, but slowly, I don't have the patience to wait. For better diagnostics of what is happening,

```
<key>Boot</key>
<dict>
	<key>Debug</key>
	<true/>
```

in the config.plist file. The boot will be very slow, since /EFI/CLOVER/misc/debug.log will be updated at each step, but after the final freeze you get information about what exactly happened. In reality, when loading from a flash drive, it can take up to ten minutes before entering the GUI. Starting with revision 3063, the screen is no longer black, if CloverGUI has started loading, you will see messages on the screen that will tell you what exactly is happening.

Q. I see 6_ on the screen and nothing else happens.

A. This is the worst case of hardware incompatibility. It is no longer encountered, except with AMD processors. Only a programmer who can insert debug messages into Clover codes and reboot after reboot until the problem is fully identified can diagnose it. Unfortunately, there is nothing to advise ordinary users. Read the chapter on slow Clover, maybe wait? Maybe play with BIOS settings, sometimes it helps. Try using boot7 (Clover BiosBlockIO) instead of the boot file, the difference is in the SATA driver, perhaps Clover gets stuck because of SATA. This is most often the case. Or reinstall the boot1 sector..

Q. The boot only reaches the text analogue of the BIOS with five items, the top one is – Continue>

A. This means that the boot file has loaded successfully and is working, but does not find the CloverX64.efi file. Either it does not see that section, or the device in general - we need to figure it out further, walking through the options of this menu. For example, the file may be missing HFSPlus.efi, and you have Clover installed on the HFS+ partition. It's strange, why would you want to make UEFI boot from the HFS+ partition.

Q. I installed Clover on a flash drive, booted from it, and I don't see my HDD..

A. First, the HDD must be inserted into the Sata0 port. In the future, this may already be Second, I understand that if 5 want to kill it, you have a well-functioning Ham, Chimera, XRS, fixed. in short, BBH (Booter on the letter H), you do not but want to try Clover, then such an action seems natural. But, nevertheless, there are options for installing Clover on a hard drive that do not kill the old bootloader, and in this case, the voiced error will disappear. Also try the boot7 file if you have some unusual SATA/SAS/RAID controller. With UEFI boot, this may also mean the absence of PartitionDxe.efi and HFSPlus.efi files.

Q. When loading UEFI, I don't see a section with MacOS, only Legacy..

A. This means that the /EFI/CLOVER/drivers/UEFI folder does not contain HFSPlus.efi or its legal analogue VboxHFS.efi.

Q. When booted with UEFI, Windows looks like legacy, even though it is EFI..

A. Same thing, the NTFS.efi driver is missing, and Windows is installed so that its bootmgfw.efi is located not in the EFI partition, but in the Windows partition. P.S. These two drivers (NTFS and HFS) are missing from the repository for licensing reasons, you need to find this file somewhere on the Internet. Now there is a legal analogue of GrubNTFS.efi. It is in the Clover installer..

Q. I set the native resolution in the bootloader, but the screen has a black frame.

A. There is no way to fix it. In any case, the Clover developers couldn't come up with anything, and no one will answer this
question. There is one option: if there is a UEFI BIOS, you need to do a UEFI boot and flash the video card to the UEFI Video
BIOS. In the BIOS, make the following settings:
- OS: Windows 8 WHQL
- CSM: Never
- Full screen logo: Disabled
Nothing can be done for legacy boot. If you don't like the mourning frame, make the resolution lower.

Q. When trying to start the OS, it freezes on a black screen

A. At this point, the DSDT is patched with your mask. Yes, ideally it should not hang here. But the problem is that many BIOS manufacturers do not comply with standards, do not know how to program, and do not want to polish their DSDT for the needs of OSX. It is very easy to see that the decompile - compile again operation does not work - the DSDT is crooked. Clover would like to fix all this, but alas, the number of bad options is still beyond even review. Therefore, you are required to select a DSDT fix mask so that the bootloader does not hang, and then so that the OS does not hang, and ideally, so that it also works. This is real. Or refuse the autopatch (mask = 0), and make the DSDT manually. See the chapter on debugging DSDT. And I also strongly recommend using the latest version of Clover, because such bugs are found and fixed from time to time. And there is also a black screen option if E2 is locked: set KernelPM=true

Q. The kernel starts loading, but panics after the tenth line Unable To find driver for this platform \"ACPI\"..

A. This is a missing or incorrect DSDT. If the autopatch does not work, add a manually made DSDT. Pay attention to the autopatch options, as well as the ReuseFFFF and DropOEM_DSM keys.

Q. The system starts to load, but gets stuck on still waiting for root device....

A. In addition to the usual advice for such cases to enable AHCI in the BIOS, or, if there is none, to find the correct driver (meaning kext) for your IDE controller, here is another advice to boot with the WithKexts key (in new NoCaches revisions), then the boot will be slower, and the controller will have time to turn on. By the way, such an error can only occur if Clover and the system are on different devices.

Q. The system boots up to the message: Waiting for DSMOS....

A. FakeSMC is missing. Maybe with Chameleon you had this kext in Extra, and Clover does not see this folder. The folder /EFI/CLOVER/kexts/Others or others is intended for it. Do not forget about the InjectKexts key. It is disabled by default! In revision 5125+ it is no longer needed. At the second stage of installation Clover does not know the system version (it has not yet been determined), so put FakeSMC in the folder /EFI/CLOVER/kexts/Other/
In new versions the InjectKexts key has the value

"Detect", which should automatically cope with this situation, check what is written in your config.

Q. The system passes this message, but nothing changes further, although the hard drive buzzing as if the system is loading.

A. A typical situation when the video card does not turn on. Try GraphicInjector=Yes in the config, or vice versa =No. In the second option, Radeons are launched on the "native startup", which even allows you to work in the system, with a few exceptions, for example, DVDplayer will not work. For a full startup of Radeon, you also need to fix the connectors. For other cases, you can try to boot the system with the -x key, and enter the desktop in VESA mode. Not very good, but it will allow you to fix something. Another option for slowdowns in this place is observed if you select the MacMini or MacBookPro model. The problem is solved by installing the DropMCFG=Yes or FixMCFG key. A modern system has the -amd_no_dgpu_accel key, but it did not help me.

Q. The system boots up to the message: [Bluetooth controller.....

A. Same thing. See the previous point. Bluetooth has nothing to do with it..

Q. The system has loaded, everything is fine, but there are errors in the System Profiler..

A. In general, this is cosmetic, does not affect the functionality. About PCI boards. See the Chapter on AAPL, slot-name About memory. There are two speed values, nominal and actual, and they often do not match. Which one should I show in the profiler? I set the first one and they yelled that it was wrong. I set the second one, they fell silent, other users yelled that it was wrong... See page 47 how to register your memory values in the config.

Conclusion

Clover is certainly far from perfect, but the process of improving programs is never complete. There will be new revisions, there will be new functions, but for now this is it. Clover's biggest drawback is that it tries to be universal. A programmer can make his own version from the source code, suitable specifically for his hardware. For the rest, there is a config with hundreds of settings, and this is too complicated for the average mind, despite the presence of automation, instructions, descriptions and a lot of advice from experts. Chameleon works due to BIOS drivers, and therefore it has a higher chance of running on arbitrary hardware, but no one keeps statistics on what percentage of cases Clover works more correctly. The development of Clover is finished, but the project is not dead, it continues to exist, and more will develop