

Клевер цвета хаки

Автор Slice



ПРЕДИСЛОВИЕ.....	5
ТАКТИКО-ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ.....	6
ЧТО ЕСТЬ ЧТО?.....	7
MBR СЕКТОР	7
PBR СЕКТОР	7
БООТ.....	8
CLOVERIA32.EFI и CLOVERX64.EFI.....	8
ВСПОМОГАТЕЛЬНЫЕ ФАЙЛЫ И ПАПКИ.....	8
ДРАЙВЕРА ЕФИ.....	10
РАЗРАБОТКА.....	11
ИНСТАЛЛЯЦИЯ.....	13
ИСПОЛЬЗОВАНИЕ ИНСТАЛЛЯТОРА.....	13
ВЫБОР ТЕМЫ.....	17
ТЕМЫ ЗАГРУЗЧИКА КЛОВЕР.....	18
НАСТРОЙКА ИНТЕРФЕЙСА: REFI.T.CONF.....	22
УСТАНОВКА ЗАГРУЗЧИКА ВРУЧНУЮ.....	26
OSX.....	26
Linux.....	27
Windows.....	27
КОНФИГУРИРОВАНИЕ АППАРАТНОЙ ЧАСТИ	27
ФАЙЛ CONFIG.PLIST.....	27
SYSTEMPARAMETERS.....	27
<key>boot-args</key>.....	27
<key>prev-lang:kbd</key>.....	28
<key>DefaultBootVolume</key>.....	28
<key>CustomUUID</key>.....	28
<key>InjectSystemID</key>.....	28
<key>LegacyBoot</key>.....	28
<key>BacklightLevel</key>.....	29
<key>iCloudFix</key>.....	29
SMBIOS.....	29
<key>FirmwareFeatures</key>.....	29
<key>BoardType</key>.....	29
<key>Mobile</key>.....	29
CPU.....	30
<key>Turbo</key>.....	30
<key>CpuFrequencyMHz</key>.....	30
<key>BusSpeedkHz</key>.....	30
<key>QPI</key>.....	30
<key>ProcessorType</key>.....	31
GRAPHICS.....	32
<key>GraphicsInjector</key>.....	32
<key>VRAM</key>.....	32
<key>LoadVBios</key>.....	32
<key>PatchVBios</key>.....	32
<key>InjectEDID</key>.....	32
<key>CustomEDID</key>.....	32
<key>VideoPorts</key>.....	33
<key>FBName</key>.....	33
<key>NVCAP</key>.....	33
<key>display-cfg</key>.....	34

KERNELANDKEXTPATCHES.....	35
<key>KernelCpu</key>.....	35
<key>AsusAICPUPM</key>.....	35
<key>AppleRTC</key>.....	35
<key>KextsToPatch</key>.....	35
<key>ATISConnectorsController</key>.....	36
PCI.....	38
<key>StringInjector</key>.....	38
<key>DeviceProperties</key>.....	38
<key>PCIRootUID</key>.....	39
<key>HDAInjection</key>.....	39
<key>USBInjection</key>.....	40
<key>LpcTune</key>.....	40
POINTER.....	40
<key>Speed</key>.....	40
<key>DoubleClickTime</key>.....	40
VOLUMES.....	40
<key>HideAllUbuntu</key>.....	40
<key>HideVolumes</key>.....	40
ACPI.....	41
<key>DropOemSSDT</key>.....	41
<key>GenerateCStates</key>.....	41
<key>C3Latency</key>.....	41
<key>GeneratePStates</key>.....	41
<key>PLimitDict</key>.....	42
<key>UnderVoltStep</key>.....	42
<key>ResetAddress</key>.....	42
<key>ResetValue</key>.....	42
<key>smartUPS</key>.....	42
<key>PatchAPIC</key>.....	43
<key>FixDsdtMask</key>.....	43
КОРРЕКТИРОВКА DSDT.....	45
FIX_DTGP BIT(0).....	46
FIX_WARNING BIT(1).....	46
FIX_SHUTDOWN BIT(2).....	46
FIX_MCHC BIT(3).....	46
FIX_HPET BIT(4).....	46
FIX_LPC BIT(5).....	47
FIX_IPIC BIT(6).....	47
FIX_SBUS BIT(7).....	47
FIX_DISPLAY BIT(8).....	47
FIX_IDE BIT(9).....	47
FIX_SATA BIT(10).....	47
FIX_FIREWIRE BIT(11).....	47
FIX_USB BIT(12).....	47
FIX_LAN BIT(13).....	47
FIX_WIFI BIT(14).....	47
FIX_HDA BIT(15).....	48
Послесловие.....	48
НАТИВНЫЙ СПИДСТЕП.....	48
CONFIGARRAY.....	49
CTRLLOOPARRAY.....	49
CSTATEDICT.....	49
ПРОБЛЕМА СНА.....	50
Клевер цвета хаки.....	3
Москва, 2012.....	

ЧАВО.....	50
<i>В. Хочу попробовать Кловер, с чего начать?</i>	<i>50</i>
<i>В. Не работает.....</i>	<i>50</i>
<i>В. Установил Кловер, но получаю черный экран.....</i>	<i>50</i>
<i>В. Вижу на экране 7_ и больше ничего не происходит.....</i>	<i>51</i>
<i>В. Происходит загрузка только до текстового аналога БИОСа с пятью пунктами, верхний – Continue>.....</i>	<i>51</i>
<i>В. Установил Кловер на флешку, загрузился с нее, и не вижу своего HDD.....</i>	<i>51</i>
<i>В. При УЕФИ-загрузке не вижу раздела с МакОСью, только легаси.....</i>	<i>51</i>
<i>В. При УЕФИ-загрузке Виндоус выглядит как легаси, хотя он EFI.....</i>	<i>51</i>
<i>В. При попытке запуска ОСи зависает после синей строки с надписью rEFIt.....</i>	<i>51</i>
<i>В. Ядро начинает грузиться, но паникует после десятой строки.....</i>	<i>51</i>
<i>В. Система начинает грузиться, но стопорится на still waiting for root device.....</i>	<i>52</i>
<i>В. Система грузится до сообщения: Waiting for DSMOS.....</i>	<i>52</i>
<i>В. Система проходит это сообщение, но дальше ничего не меняется, хотя винчестер жужжит, как будто система грузится.....</i>	<i>52</i>
<i>В. Система загрузилась, все хорошо, но в Систем Профайлере ошибки.....</i>	<i>52</i>
ЗАКЛЮЧЕНИЕ.....	52

Предисловие

О чем идет речь? Да уж разумеется не о цветочке, растущему на лугу на радость коровам. Речь идет о программном обеспечении, о загрузчике нового типа, который позволяет на обычном компьютере запустить необычную операционную систему – Mac OSX. Apple этого делать не разрешает, в первую очередь мотивируя тем, что “мы не можем обеспечить работоспособность на компьютерах, произведенными не компанией Apple”. Что ж, ставим систему на свой страх и риск. Ну и не стоит получать какую-то коммерческую выгоду из этого, во избежание других юридических сложностей. Не-эппловский компьютер с установленной системой MacOSX называется Хакинтош, происхождение слова понятно.

Чтобы запустить Хакинтош, нужен специальный загрузчик, их много разных, но по своей основе можно разделить на два класса: FakeEFI и RealEFI.

FakeEFI изобретен David Elliot много лет назад, и действует по-простому принципу: сделаем вид, что у нас EFI уже отработала, оставим в памяти следы его деятельности (boot-args и все дерево таблиц), оставим в памяти EfiRuntime в упрощенном виде “Неподдерживается”, и запустим ядро mach_kernel. Так работает Хамелеон, и работает успешно, но за небольшими исключениями типа панели “Загрузочный диск”. Не исключено, что со временем Эппл даст нам и другие проблемы, связанные с отсутствием Рантайм Сервисов.

Real EFI должен был бы быть прошит вместо БИОСа, но для тех, у кого материнская плата на основе БИОС, придуман загружаемый EFI. Эта система, загрузка EFI на машине с BIOS придумана Intel, и сейчас находится в активной разработке с открытыми исходными кодами на сайте tianocore.org. Собственно этот загрузчик называется DUET. Да вот беда. EFI-то он загружает, а вот загрузка операционной системы МакОС там не предусмотрена. Требуется следующий шаг, приспособить DUET под требования МакОС.

Название Clover данный загрузчик получил от одного из первооснователей проекта kabyul’a, который увидел сходство клавиши “Command”, существующей только на Маках, с четырехлистным клевером.



***Четырехлистный клéвер** — одиночное растение клевера, обладающее по крайней мере одним четырехпластинчатым листом, в отличие от обычных трёхпластинчатых. В западной традиции существует поверье, что такое растение приносит удачу нашедшему, в особенности если оно найдено случайно^[1]. По легенде каждая из пластинок четырёхпластинчатого листа представляет что-то конкретное: первая — надежду, вторая — веру, третья — любовь, а четвёртая — удачу^[2].*

В русском варианте мы называем загрузчик «Кловер».

Разработка проекта идет на форумах

<http://www.projectosx.com/forum/index.php?showtopic=2304&st=0>

<http://www.applelife.ru/threads/clover.32052/>

1 Интересно, а как можно найти клевер случайно? Регулярно щипать травку на лугу?

Тактико-технические характеристики

EFI – Extensible Firmware Interface – расширяемый интерфейс доступа к аппаратно-зависимым функциям. В отличие от БИОС, который занимает 64кб и написан в 16-битных кодах, ЕФИ занимает от 4Мб, написан в 32 или 64-битных кодах, и позиционируется как аппаратно-независимый, хотя... конечно, чудес не бывает, и 100% совместимости с любой платформой добиться невозможно.

Clover – это EFI загрузчик операционных систем, для компьютеров уже имеющих UEFI BIOS, и для компьютеров, не имеющих такового. При этом сами операционные системы могут поддерживать EFI- загрузку (OSX, Windows 7-64EFI, Linux), либо нет (Windows XP), в этом случае предусмотрен legacy-boot – возврат к старой схеме BIOS-загрузки через бутовые сектора.

EFI – это не только начальный этап загрузки ОС, она создает также таблицы и сервисы, которые доступны для использования в ОС, и её работоспособность зависит от корректности ЕФИ. Нельзя на встроенном UEFI загрузить OSX, также, как нельзя загрузить OSX из чистого Дюета. CloverEFI и CloverGUI выполняют немалую работу по корректировке встроенных таблиц для возможности запуска OSX.

- таблица SMBIOS (DMI) заполняется данными, эмулирующими реальные компьютеры Apple Macintosh – условие запуска OSX. Серийные номера выдуманные, но подходящие.
- ACPI таблицы, содержащиеся в ROM компьютера, как правило содержат ошибки и недостатки, чаще всего из-за лени производителей: в таблице APIC неправильное число ядер ЦПУ, отсутствуют данные NMI, в таблице FACP отсутствует регистр Reset, неправильный профиль питания, в таблицах SSDT отсутствуют данные для EIST, а уж про DSDT вообще длинный разговор. Clover пытается все это поправить.
- OSX также стремится получить от загрузчика данные о дополнительных устройствах, таких как видеокарта, сетевая, звуковая и т.д. через механизм EFI-string. Clover формирует такую информацию.
- для компьютеров на основе BIOS характерно использование USB на начальной стадии загрузки в режиме Легаси, что становится неприемлемым при передаче управления в ОС. Загрузчик осуществляет переключение режима работы USB.
- также OSX обменивается с EFI информацией через специальную память NVRAM, доступ к которой осуществляется через RuntimeServices, отсутствующие в легаси-загрузчиках. Кловвер предоставляет такой обмен информацией, причем двухсторонний, что дает правильную работу Firewire и возможность использование панели управления "Стартовый Диск" для автоматической перезагрузки в другую систему.
- перед началом работы CPU должен быть правильно проинициализирован, но, поскольку системные платы изготавливают универсальными, на целый круг разных ЦПУ, во внутренних таблицах отсутствуют данные по процессору, либо представлены какие-то универсальные, некорректные в частном случае. Кловвер осуществляет полный детект установленного ЦПУ и делает необходимые коррекции в таблицах, и в самом ЦПУ. Как один из результатов - включается режим Турбо.

Все это осуществляется автоматически, и новичок может использовать Кловвер даже ничего не понимая в указанных проблемах. Ну а продвинутому пользователю Кловвер предоставляет возможности по изменению вручную множества параметров. Их рассмотрение и представляет собой цель этой книги.

Что есть что?

При включении или при перезагрузке компьютера загрузка операционной системы с помощью Кловера происходит по следующему пути.

Вариант А. Компьютер основанный на БИОС (старая схема)

BIOS->MBR->PBR->boot->CLOVERX64.efi->OS (boot.efi->mach_kernel в случае MacOSX).

Вариант Б. Компьютер, основанный на УЕФИ БИОС (новая схема)

UEFI BIOS->CLOVERX64.efi->OS

Чтобы это осуществлялось, следующие файлы должны быть прописаны в следующих местах:

MBR сектор

нулевой блок внешнего носителя, с которого происходит загрузка (HDD, SSD, USB Stick, USB HDD, DVD). В этот блок следует записать в первые 440 байт один из вариантов:

boot0 – его роль в поиске активного раздела в MBR разметке диска, и передача управления на его сектор PBR. Возможен вариант гибридной схемы разделов MBR/GPT.

boot0hfs – поиск первого раздела с сигнатурой 0xAF, т.е. HFS+ раздела с установленной OSX, и передача управления в его PBR. Таким образом можно работать с GPT размеченным диском, хотя БИОС этого и не предусматривает.

boot0md – комбинированный вариант, который ищет HFS+ раздел по нескольким дискам, а не только по главному.

PBR сектор

первые блоки каждого раздела на выбранном носителе. Сюда записывается загрузчик фаза два. Этот загрузчик знает файловую систему своего раздела, и способен отыскать там файл с именем boot, чтобы загрузить его, и передать в него управление. Соответственно, существуют варианты под разные файловые системы. *boot1h2* – для файловой системы HFS+ и поддержкой длины файла boot до 472кб. Старый вариант, распространяемый с загрузчиком «Хамелеон» поддерживает только 440кб файл.

boot1f32alt - для файловой системы FAT32. Эта файловая система имеет поддержку записи, поэтому очень удобна для установки на нее загрузчика. Можно использовать EFI раздел, можно использовать флешку, как она есть, поскольку в продажу поступают флешки уже отформатированные в FAT32².

Эти два сектора (точнее сказать загрузчика фазы 2) имеют еще одну полезную функцию. Они имеют начальную задержку на старте длиной в две секунды в ожидании ввода с клавиатуры. Введенная цифра будет присоединена к имени файла, т.е. нажав клавишу 1 мы загрузим файл boot1, нажав клавишу 3 мы загрузим файл boot3, нажав клавишу 6 мы загрузим файл boot6. Смысл в том, чтобы держать в

²По историческим причинам рекомендуют переформатировать каждую флешку, да еще и обязательно в системе Windows. Нынче эта рекомендация устарела, и не имеет под собой основания. Фабричный формат пригоден для установки Кловера.

одном месте разные варианты загрузчиков, или даже разные загрузчики, просто дав им разные цифры. К примеру

boot - Clover, текущая версия, или тестовая версия

boot1 – Хамелеон

boot3 – Clover-32bit, проверенная, рабочая версия

boot6 – Clover-64bit, проверенная, рабочая версия

Кроме этих вариантов в секторе PBR могут находиться boot manager Windows, знающий файловую систему NTFS, GRUB, знающий файловую систему EXT3, и другие, не имеющие отношения к Кловеру. По-крайней мере на данном этапе.

boot

В случае с Хамелеоном это и есть весь загрузчик. В случае с Кловером вариант А в этом файле лежит вся система EFI, а также бут-сервис для передачи управления в следующий этап. Все это, в варианте Б следует считать уже присутствующим в ROM компьютера. Реально получается, что там есть не все, и кое-какие детали следует загружать дополнительно. Детали, уже собранные в файл boot варианта А.

В отличие от предыдущих стадий файл boot уже отличается по битности, т.е. отдельные варианты для 32 и 64 битной загрузки. В большинстве случаев выбирать следует 64битный вариант, если процессор поддерживает этот набор инструкций. Впрочем, если по каким-то причинам предполагается работать только в 32-битной ОС, то имеет смысл грузить именно EFI32. Она меньше по размеру на 20%, и соответственно быстрее, но несовместима с Windows 7 EFI, которая всегда 64бита. По своей сути файл boot представляет собой модифицированный DUET. Причём исправлений по существу вряд ли наберётся на 1%, но этот процент обеспечивает кардинальное отличие от Дуэта – Кловер работает для той цели, для которой предназначен.

В дальнейшем называем эту программу обобщенно CloverEFI.

CLOVERIA32.efi и CLOVERX64.efi

Этот файл, в двух вариантах разной битности, представляет собой графическую оболочку загрузчика для выбора операционной системы, для установки дополнительных опций, для подгрузки дополнительных драйверов и собственно для запуска ОС. Графика и меню изначально основаны на проекте rEFIt, что отражено в названии директории и в меню About. На настоящий момент оригинальная часть составляет едва ли 10% от всей программы, да и то в исправленном виде.

В дальнейшем обобщенно называем эту программу CloverGUI.

Вспомогательные файлы и папки

Кроме того, загрузчику нужны дополнительные файлы, структура папок будет следующая.

EFI:

```
ACPI:
    mini:
        DSDT.aml
WINDOWS:
    SLIC.aml
origin:
patched:
    DSDT.aml
```



```

BOOT:
  CLOVERIA32.efi
  CLOVERX64.efi
  font:
    BoG_LucidaConsole_10W.png
    dark-wide.png
  refit.conf
  themes:
    black_green:
      icons:
        func_about.png
        os_clover.icns
        os_unknown.icns
        logo.png
        Selection_big.png
        Selection_small.png
      hellfire:
      metal:
  config.plist
  drivers32:
    FSInject-32.efi
  drivers64:
    FSInject-64.efi
    NTFS.efi
  drivers64UEFI:
    DataHubDxe.efi
    FSInject-64.efi
    HFSPPlus.efi
    NTFS.efi
    OsxAptioFixDrv-64.efi
    OsxFatBinaryDrv-64.efi
    PartitionDxe.efi-64.efi
  kexts:
    10.6:
    10.7:
    10.8:
    Other:
  misc:
  OEM:
    Inspiron 1525:
      ACPI:
        origin:
        patched:
          DSDT.aml
      config.plist
      kexts:
        10.5:
        10.6:
          Injector.kext
          VoodooSDHC.kext
        10.7:
        Other:
  ROM:
  tools:
    Shell132.efi
    Shell164.efi

```

То есть, файл CLOVERX64.efi должен находиться по адресу /EFI/BOOT/, а шрифт dark-wide.png в папке /EFI/BOOT/fonts/. Реально там лежит больше шрифтов, также и

другие папки наполнены содержимым. По ходу повествования будем описывать подробнее, что для чего служит.

Драйвера EFI

Отдельно упомяну папки `drivers32`, `drivers64`, `drivers64UEFI`, соответственно для 32, для 64битной загрузки по варианту А – BIOS boot, и для UEFI загрузки по варианту Б. Состав этих папок будет отличаться для разных ревизий БИОСа, а также для разных конфигураций разделов.

Стоит заметить, что эти драйвера имеют силу только на период работы загрузчика. На загруженную операционную систему они не влияют, разве что косвенно.

Что следует положить в эти папки? На выбор пользователя.

- `NTFS.efi` – драйвер файловой системы NTFS, для возможности грузить Windows EFI.
- `HFSPlus.efi` – драйвер файловой системы HFS+, необходим для запуска MacOSX. Необходим для варианта Б, а вот в А он уже присутствует.
- `VBoxExt2.efi` – драйвер файловой системы EXT2/3, необходим для запуска Linux EFI.
- `FSInject.efi` – драйвер перехватывающий файловую систему, для возможности инжектировать внешние кексты в систему. Сложно для понимания? Позже к этому вопросу еще вернемся, когда будем рассматривать ключ `WithKexts`
- `PartitionDxe.efi` – вообще-то, такой драйвер есть в CloverEFI, да и в UEFI он есть, но только он не рассчитан ни на Apple partition, ни на гибриды MBR/GPT. Вывод: в варианте Б драйвер нужен.
- `OsxFatBinaryDrv.efi` – необходимый драйвер для варианта Б, обеспечивает запуск толстых (Fat) модулей, каким является `boot.efi`.
- `OsxAptioFixDrv.efi` – особый драйвер, предназначенный для коррекции карты памяти, которую создает AMI AptioEFI, иначе запуск OS невозможен.
- `Usb*.efi`, `UHCl.efi`, `EHCl.efi`, `XHCl.efi` – набор драйверов USB, для тех случаев варианта Б, когда встроенные драйвера почему-то работают плохо. С чего бы вдруг? Возможно, есть какая-то завязка на другие функции, которые пришлось отключить.
- `PS2Mouse...`, `PS2MouseAbsolute...`, `UsbMouse...` - набор драйверов для поддержки указателя мыши/трекпада/тачпада в интерфейсе CloverGUI. На операционную систему эти драйвера не влияют.
- `DataHubDxe.efi` – этот драйвер уже присутствует в варианте А, и вполне возможно, есть и в UEFI. Но на случай если его там нет, стоит загрузить внешний. Конфликта не возникнет, зато будет уверенность, что он есть.

Разработка

Проект не имеет коммерческой значимости по лицензионным причинам, и еще и очень велик по объему, чтобы делать его в одиночку, поэтому самое разумное решение сделать его с открытыми исходниками³, и пусть все желающие внесут свой вклад. В настоящий момент проект базируется на сервере sf.net в репозитории <http://cloverefiboot.sourceforge.net/>

Для того, чтобы скомпилировать проект, необходимы еще компилятор, и библиотеки – азбучная истина. Что в данном случае? В роли библиотек выступают исходные модули EDK2. Как это правильно назвать? Framework? Environment? По-русски наверно надо сказать **СРЕДА**. Скачивается с того же сервера.

<http://sourceforge.net/projects/edk2/>

Поскольку весь проект создавался для Хакинтоша и ради Хакинтоша, то в первую очередь рассматривается компиляция проекта в среде MacOSX. Это, однако, не единственная возможность. Сам по себе EDK2 предусматривает компиляцию также в Windows, Linux, и в каких-то других системах и средах. Для Windows нужно иметь Visual Studio 200x, для MacOSX должен быть установлен Xcode Command Line Tools, а в Linux компилятор gcc входит по-умолчанию. Встроенные средства MacOSX тем не менее недостаточны для компиляции проекта, поэтому предлагается, как и для Linux, скачать и собрать новую версию gcc, используя скрипт buildgcc.sh в папке Кловера.

Теперь к делу. Читатель, заглянувший в эту главу, не может по определению быть простым юзером, и уж ему не требуется рассказывать, как пользоваться терминалом.

1. Скачиваем исходники и готовим среду:

```
cd ~
mkdir src
cd src
svn co https://edk2.svn.sourceforge.net/svnroot/edk2/trunk/edk2 edk2
cd edk2
make -C BaseTools/Source/C
svn co svn://svn.code.sf.net/p/cloverefiboot/code/ Clover
cd Clover
```

2. Собираем компилятор. Даже два, один для 32-битной компиляции, другой для 64.

```
./buildgcc.sh -ia32 -all
./buildgcc.sh -x64 -all
```

3. Исправляем среду EDK2 под наши нужды

```
cp ~/src/edk2/Clover/build_rule.txt ~/src/edk2/Conf/
cp ~/src/edk2/Clover/tools_def.txt ~/src/edk2/Conf/
cp ~/src/edk2/Clover/MdeModulePkg.dec ~/src/edk2/MdeModulePkg/MdeModulePkg.dec
```

3Проекты с открытыми исходниками тем не менее имеют некоторые ограничения в использовании, так сказать “лицензирование”. И, в частности, лицензия GPL примененная для этого проекта, развита в применении к Линуксу. Одно “но”. Эта лицензия плоха и опасна, поэтому лицензия Кловера сменена на BSD .

начиная с версии 599 MdeModulePkg.dec можно не копировать.

4. Теперь можно и сам CloverEFI собирать. Например так:

```
./ebuild.sh -64  
./ebuild.sh -32
```

Созданы и другие скрипты компиляции, как правило самодокументированные. Смотрите, выбирайте, пользуйтесь.

5. Одна тонкость. В репозитории отсутствуют файлы HFSPlus.efi для 32 и 64 бит, ввиду их приватности. Два варианта: раздобыть эти файлы из других источников, либо изменить определения проекта файлы .fdf таким образом, чтобы вместо приватных драйверов использовались свободные VboxHfs. Этот драйвер работоспособен, но медленный и с недостатками, которые в будущем, возможно будут исправлены.

Было

```
# foreign file system support  
#INF Clover/VBoxFsDxe/VBoxHfs.inf  
INF RuleOverride=BINARY Clover/HFSPlus/HFSPlus.inf
```

Сделать

```
# foreign file system support  
INF Clover/VBoxFsDxe/VBoxHfs.inf  
#INF RuleOverride=BINARY Clover/HFSPlus/HFSPlus.inf
```

6. Проект не стоит на месте, поэтому эта инструкция со временем может оказаться неработоспособной из-за какого-то мелкого изменения. Этот проект для тех кто думает, кто сможет разобраться, в чем дело, и как быть.

7. Теперь компилируем вторую часть – графическую оболочку CloverGUI.

```
cd rEFIt_UEFI  
./build32.sh  
./build64.sh
```

8. Ну а теперь собираем Инсталлятор

```
cd ~/src/edk2/Clover/CloverPackage/  
./makepkg
```

Все сделано! Какие-то шаги можно было опустить, если вы делаете для себя и не в первый раз. А теперь можно приступать к инсталляции.

Инсталляция

Использование инсталлятора

Для чего сделан инсталлятор? Чтобы установить программу! Зачем же это делать вручную, инсталлятор все сделает точнее, чем вы сами! Единственное условие, что у вас на этом компьютере уже есть MacOSX. Один из вариантов, что вы запустили установочный DVD с другим загрузчиком, и из интерфейса установки MacOSX запустили инсталлятор. В зависимости от языка ОС инсталлятор будет работать по-русски, по-английски, или даже по-китайски. Здесь приведены инструкции для английского варианта, поскольку по-русски и так разберетесь, а по-китайски и я не знаю. Начиная с версии 649 имеется 9 языков, в том числе индонезийский, может кому надо.

Итак,



Следуем по клавишам Continue и ОК, читаем и соглашаемся с лицензионными соглашениями (хм, а они там есть?), и приходим к выбору, что мы устанавливаем, куда и зачем.



Change Install Location – выбор куда именно ставить загрузчик



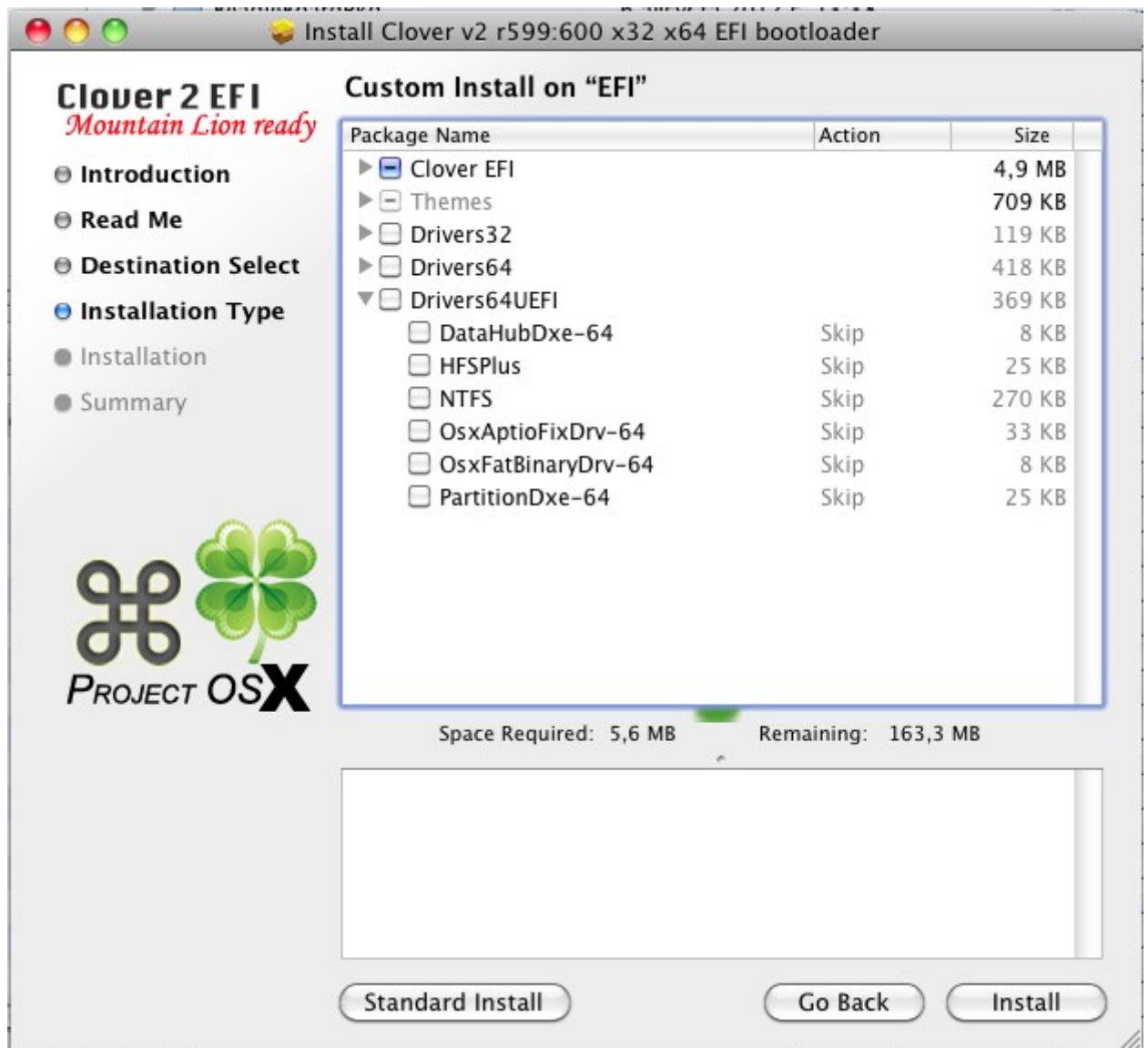
Customize на предыдущем экране – выбор вариантов загрузки



Здесь мы видим пять пунктов с цифрой 1, это означает либо-либо. Если поставить курсор на одну из строк, то в нижнем поле будет краткое описание этого варианта. Первые четыре варианта, это вариант с БИОС (вариант А), при котором используется CloverEFI.

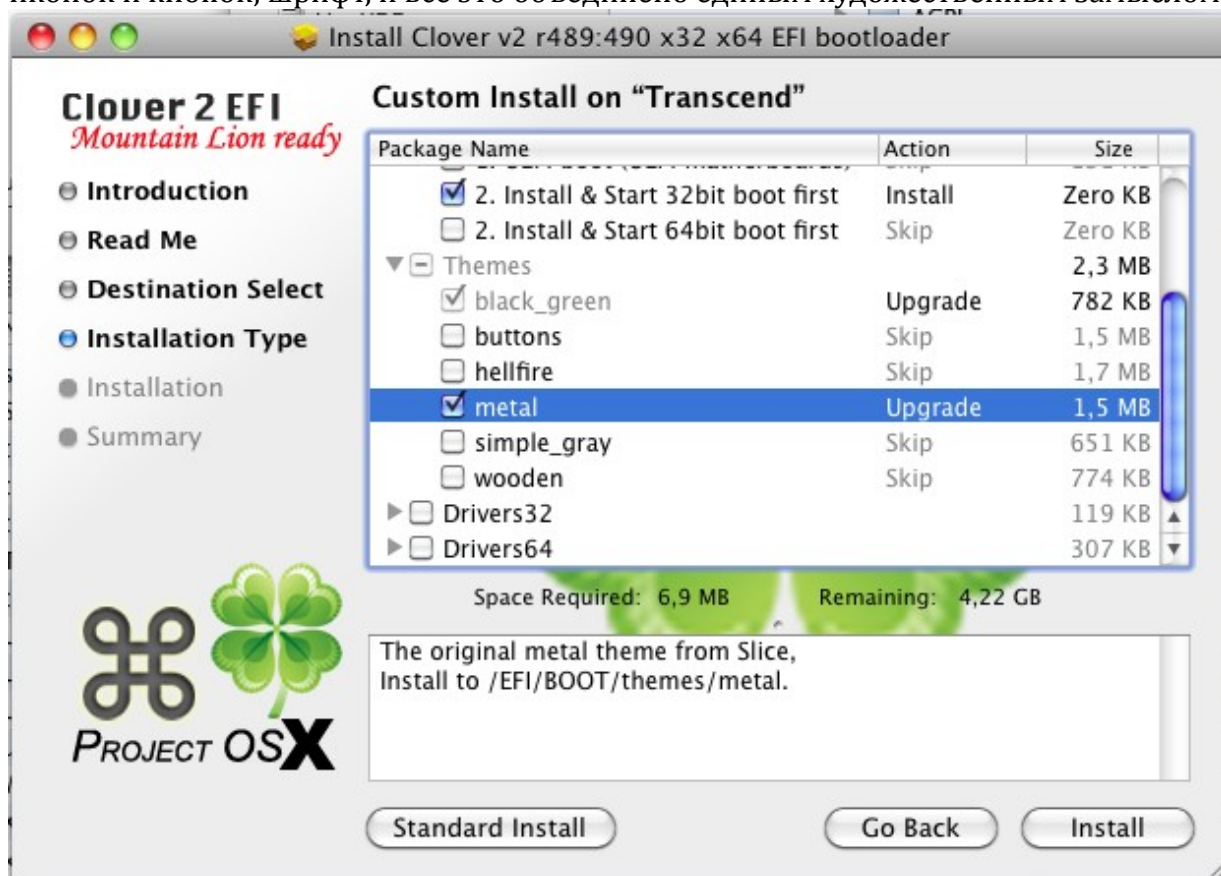
- BIOS MBR boot0 – загрузка с использованием boot0, т.е. поиск активного раздела. Инсталлятор сделает выбранный раздел активным.
- BIOS MBR boot0hfs – загрузка с использованием boot0hfs, т.е. поиск раздела HFS+, даже если он неактивный. Инсталлятор не меняет текущий активный раздел. Это сделано для конфигурации с активным Виндоус разделом – ему это надо.
- BIOS GPT EFI – установка в EFI раздел диска, отформатированного в GPT
- Unpack only – в этом варианте сектора MBR и PBR не устанавливаются, фактически, просто происходит распаковка папки /EFI. Вариант очень удобен, если вы осуществляете просто апдейт существующей инсталляции.
- UEFI boot – вся необходимая инсталляция для варианта Б.

Следующие два пункта с цифрой два , это, как видно из названия, выбор битности загрузчика. Либо 32 бита, либо 64 бита. Выбор драйверов объяснен выше , в главе «Что есть что».



Выбор темы

Теперь выбираем тему. Что есть тема? Это оформление: баннер, цвет фона, рисунки иконок и кнопок, шрифт, и все это объединено единым художественным замыслом.



В инсталляторе можно указать и все темы, чтобы позже менять их по настроению. Сделаем краткий обзор тем.

Темы загрузчика Кловер
Metal. Автор Slice. Тема №1



dawn. Автор Slice. Не включена в стандартный инсталлятор



Black-green. Автор blackosx. Тема, идущая в этом инсталляторе по умолчанию.



steampunk. Автор Xmedik. Тема включена во второй инсталлятор по умолчанию.



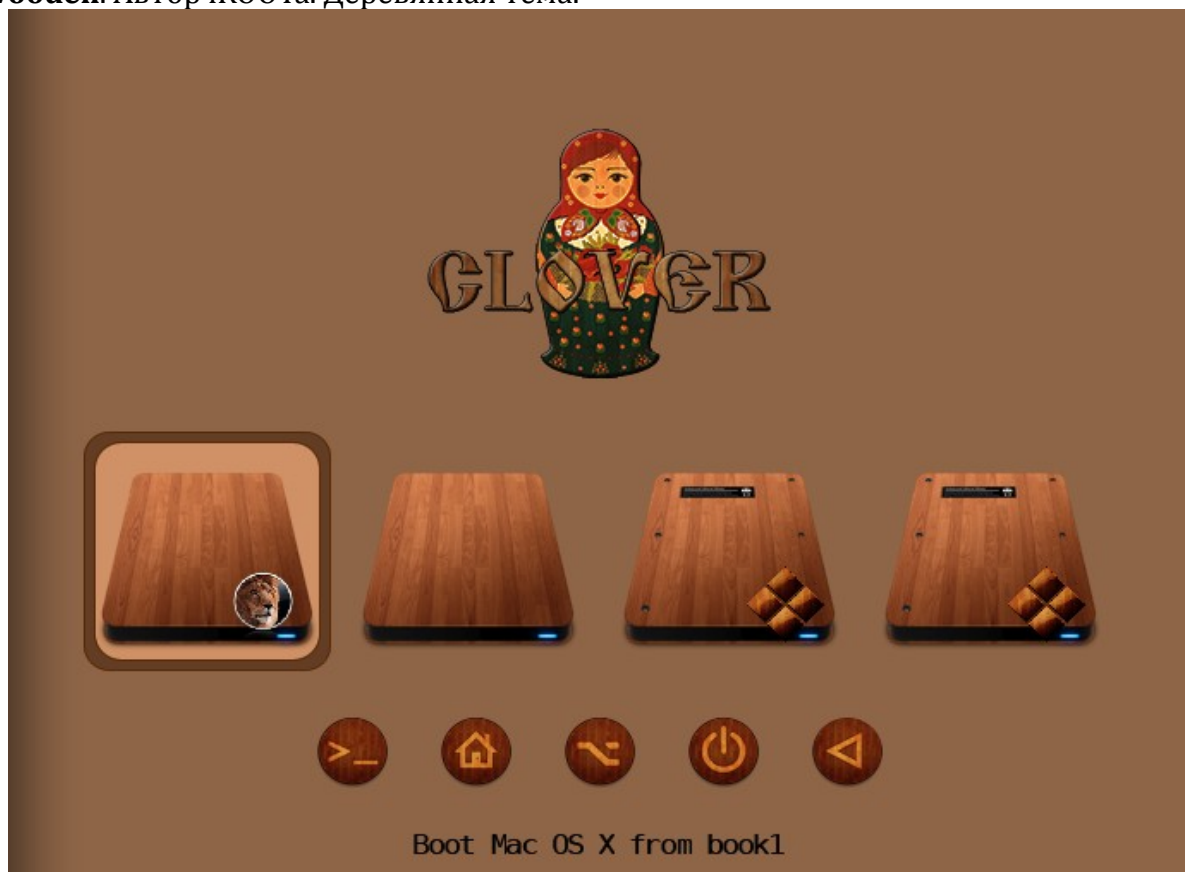
hellfire АВТОР Illevelll.



simple-gray АВТОР hijeane.



Wooden. Автор iROOTa. Деревянная тема.

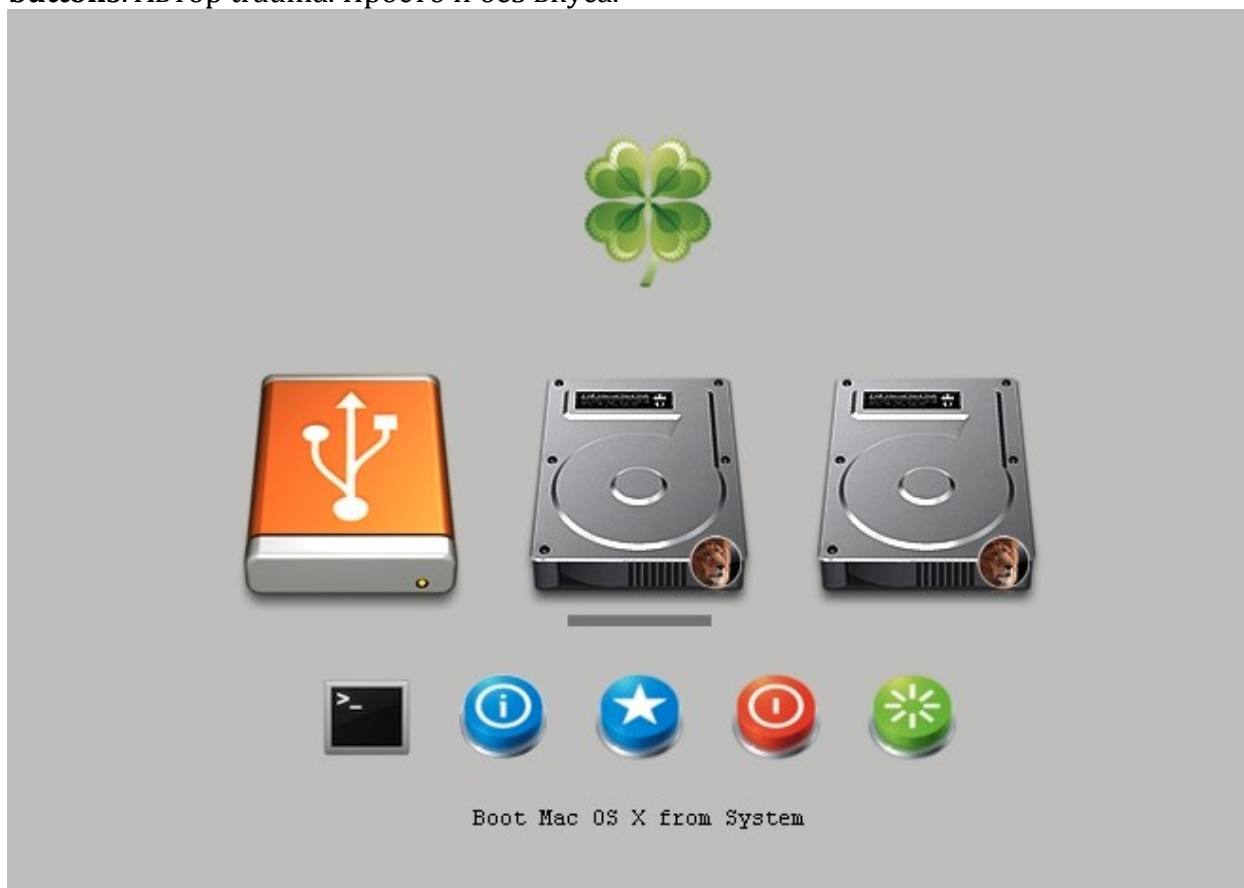


aluminium. Автор iROOTa.



Клевер цвета хаки.
Москва, 2012

buttons. Автор trauma. Просто и без вкуса.



Настройка интерфейса: refit.conf

К темам относится и ряд параметров, прописанных в файле `refit.conf`, находящимся по адресу `EFI/BOOT/`. Инсталлятор создает файл в соответствии с выбранной темой, а также предлагает и варианты для других тем.

`refit.conf-black_green`

`refit.conf-buttons`

`refit.conf-hellfire`

`refit.conf-metal`

`refit.conf-simple_gray`

`refit.conf-wooden`

Рассмотрим этот файл подробнее. Это текстовый файл, такой, как был задуман основателем Рефита, и сохранен практически без изменений, только дополнен некоторыми параметрами для поддержки тем. Редактируется этот файл любым текстовым редактором, каждый параметр внутри сопровождается комментариями на английском языке. Распишем по-русски, что есть что.

Общее правило для этого файла: все, что написано справа от `#` игнорируется.

timeout 5 – загрузчик вошел в графический интерфейс и сделал паузу в 5 секунд перед стартом системы по умолчанию. Если в течение этого времени юзер нажмет какую-либо клавишу, отчет времени прекратится. Варианты: если 0 – ГУИ не вызывается, система сразу стартует. -1 (минус один) – загрузчик входит в меню, попыток старта не делает. Из положительных чисел 5 – единственно разумная цифра. Пауза на 20 секунд?! Для каких тормозов?

#disable hwtest – отдавая дань уважения автору рефита, этот параметр присутствует. Однако нет смысла что-либо запрещать.

theme metal – выбор темы. Название определяется именем папки, где лежат все файлы темы.

font load – выбор шрифта. Есть два встроенных шрифта alfa и gray, и десяток загружаемых – load. В этом случае имя файла указывается в следующем параметре

font_file_name BoG_LucidaConsole_10W.png

Для каждой темы её автор подобрал шрифт, наиболее соответствующий его замыслу, следует смотреть в прилагаемом файле.

Размер одного символа в файле 16 пикселей, однако, сами символы занимают меньше места, поэтому следующим параметром указывается оптимальная ширина, и это, опять-таки, зависит от замысла автора.

char_width 10

Рисунок в верхней половине экрана – баннер, является основным элементом темы

banner logo_metal4.png – файл располагается в папке темы, и потому может иметь одно и то же название для всех тем, например **logo.png**. Поддерживаются форматы PNG и BMP, последний – исторически, и в нем, разумеется, нет смысла.

Логотип следует либо сделать непрозрачным, если мы не собираемся использовать фоновый рисунок. Тогда первый пиксель логотипа определяет цвет фона.

Либо на логотипе есть непрозрачный элемент на прозрачном фоне, а весь экран покрыт фоновым изображением.

background MetalBack.png crop — использовать для фона изображение из файла PNG. В случае, если размер изображения не совпадает с размером экрана, то выполняется операция:

scale — изображение масштабируется до нужного с искажением пропорций;

crop — изображение располагается по центру экрана, выступающие детали обрезаются;

tile — экран заполняется мозаикой из повторяющегося рисунка.

Поскольку этот файл является нижним слоем изображения на экране, его не стоит делать прозрачным.

Пользователь меню должен видеть, какой элемент меню выбран, а какие – нет. Следующие параметры дают гибкость для замысла темы:

selection_big Selection_big.png

selection_small Selection_small.png

Это графика, которая накладывается на большие и маленькие иконки, соответственно, когда они выбраны. Варианты выбора могут быть разные, и серый квадрат, и стрелочка в духе BootCamp, и красно-зеленый светофор. На что хватит фантазии. Рисунок имеет смысл делать полупрозрачным, чтобы более плавно вписывался в фоновое изображение.

selection_color 0xFFCC5580 – а это для текстового меню. Содержит четыре пары шестнадцатеричных цифр, означающие яркость цветов R, G, B соответственно, и канал прозрачности (альфа). В данном случае R=0xFF=255, G=0xCC=204, B=0x55=85, A=0x80=128. 0 – нет свечения, 255 – максимальная яркость. К примеру, маска 0x00FF0000 означает, что светится только зеленый цвет, и на максимальной яркости, маска 0x77990000 соответствует наполовину светящимся красному и зеленому, что даст темно-желтый цвет. Зеленого больше, поэтому оттенок будет лимонный. Ну и так далее. С этой маской цвета хорошо знакомы те, кто работал в фотошопе.

Начиная с ревизии 793 введена прозрачность. A=0 — точка вообще не отображается (абсолютно прозрачна), A=0xFF — точка совсем непрозрачна. Это играет роль, чтобы

сквозь бар выделения был виден фоновый рисунок. Если поставим A=0, что выделения вообще не видно, если 255 — то за выделением не видно фона.

anime 38 ML_Anim 15 100 once — использование анимированных изображений.

Первая цифра — Animation ID — определяет использование данного клипа.

#Logo	(1)
#About	(2)
#Help	(3)
#Options	(4)
#Graphics	(5)
#CPU	(6)
#Binaries	(7)
#DSDT	(8)
#BOOT Sequence	(9)

#Apple	(21)
#WinXP	(22)
#Clover	(23)
#Linux	(24)
#BootX64.efi	(25)
#Vista	(26)

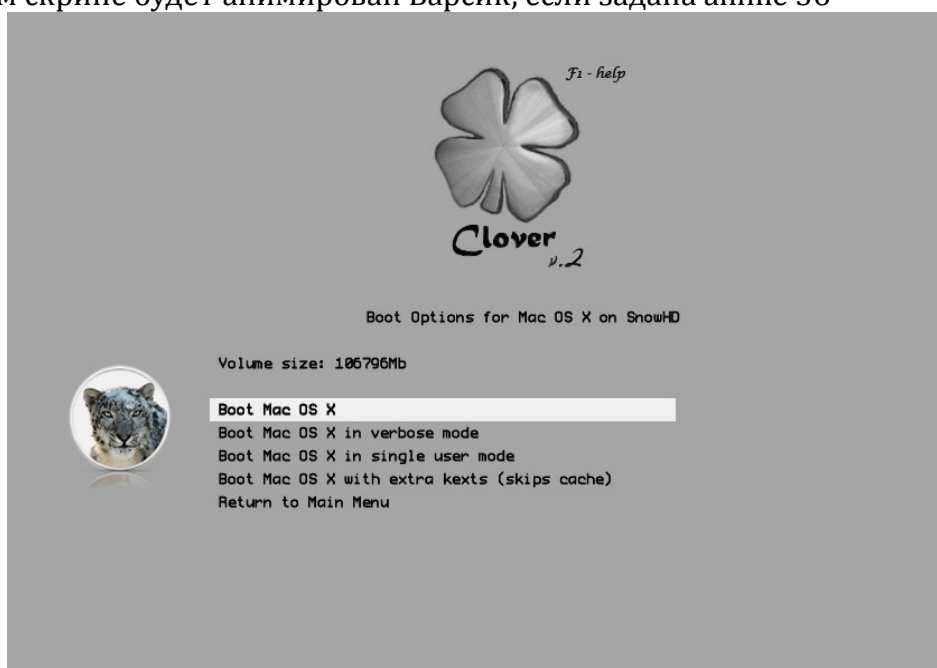
#Recovery	(30)
#Tiger	(34)
#Leopard	(35)
#Snow Leopard	(36)
#Lion	(37)
#Mountain Lion	(38)

Анимируются заголовочные изображения в каждом субменю, а также эта анимация воспроизводится на выбранном пункте главного меню.

1-9 — список существующих субменю настроек.

21-26, 30-38 — это меню подробностей «Опции загрузки», вызываемом пробелом на иконке в главном меню, либо правым щелчком мыши.

Т.е. на этом экране будет анимирован Барсик, если задана anime 36



ML_Anim — Название анимации, определяет имя папки, в которой лежат отдельные кадры с именами

ML_Anim_000.png

ML_Anim_001.png

ML_Anim_008.png

ML_Anim_014.png

В случае пропущенных кадров будет использован последний действующий, т.е. в качестве кадров 002-007 будет использован кадр 001, а в качестве 009-013 — кадр 008. Это удобно, если по сюжету в этот период времени картинка не меняется.

Следующее число — **15** — общее число кадров в анимации. Недостающие будут заполнены по вышеуказанному алгоритму.

Следующее число - **100** — временной интервал между кадрами в мс. Переменный интервал реализуется с помощью пропущенных кадров.

once — если указано, то анимация будет сыграна всего один раз, до выхода из главного меню (щелчок правой клавишей мыши в молоко на главном экране, либо клавиша Escape). Если не указано, то анимация проигрывается по бесконечному циклу, за последним кадром идет нулевой после такого же интервала, без дополнительной паузы.

hidebadges none – в главном меню есть большие иконки устройств, и к ним прикреплены маленькие иконки операционных систем – баджики. Смотрите снимки тем. В этом параметре мы управляем, что и как показывать. Варианты: **swap** – большие иконки ОС и маленькие устройств (как в теме black-green), **none** – как в теме metal, **drive** – как в теме simple-gray, **all** — не показывать баджики — в этом специальном варианте баджик появится в строке названия тома, **internal** — не показывать баджики на внутренних дисках, но показать на внешних.

#hideui tools func hdbadges – убрать какие-то элементы с экрана, например баннер. Не думаю, что автор темы согласен с таким использованием.

Может быть имеет резон убрать с экрана циферку номера ревизии, потому что не вписывается в художественный замысел. **hideui revision**

#textonly – использовать текстовый вариант загрузчика. В настоящий момент этот вариант не работает. Есть только графическое меню.

quiet – молчаливая загрузка, когда на экран не выводится лишних сообщений. Сообщения нужны для отладки, если что-то не работает, посмотреть, на каком этапе.

#nolegacy – как правило в меню появляется масса ненужных дисков, на которых загрузчик нашел какие-то следы операционных систем, и предполагает, что оттуда можно загрузиться. Если же вы уверены, что вам легаси-загрузка не нужна, можете включить этот параметр.

nolog — запрещает создание boot-log. Это может помочь с ускорением загрузки, если вы уже насмотрелись на свой лог. Но для отчетов он таки нужен. Впрочем, при удачной загрузке вы получите свой boot.log либо как следствие установленного скрипта rc.local, либо с помощью программы **DarwinDumper**.

Установка загрузчика вручную

Нужна в двух случаях. Во-первых, когда человек хорошо знает, что он делает, и хочет контролировать каждый шаг, не веря инсталлятору (а зря!), и во-вторых, при установке из-под другой ОС, где запуск инсталлятора невозможен.

OSX

Очень не рекомендуется заниматься этим тому, кто не знает, что такое терминал.

Установка на раздел HFS+ в MBR или гибридной разбивке. Почему MBR? Это очень стандартная ситуация, когда компьютер уже существует, и уже с информацией, ничего терять нельзя, можно только поставить новый загрузчик.

Установка сектора MBR

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

Что в этой команде?

fdisk440 – специальная версия утилиты fdisk, поправленная так, чтобы использовала только 440 байт нулевого сектора, есть сведения, что это необходимо для совместимости с Windows (проблема просыпания), о чем Apple не позаботилась.

boot0 – файл, описанный выше в главе "Что есть что"

rdisk0 – физическое устройство, на которое вы собираетесь ставить загрузчик.

Убедитесь, что оно действительно имеет номер 0.

Эти файлы поставляются вместе с Кловером.

Установка сектора PBR

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

boot1h2 – файл сектора PBR для файловой системы HFS+, отличается от аналогичных поддержкой больших файлов boot, и возможностью выбора boot1,3,6 по горячей клавише. Подробности в главе "Что есть что".

rdisk0s9 – девятый раздел на выбранном устройстве... Почему девятый? А чтобы дураки ничего не испортили, тупо повторяя написанные команды, такого раздела наверняка нету. А ставить нужно реальную цифру, например первый раздел.

Ну и после того, как сектора MBR и PBR успешно записаны на выбранное устройство/выбранный раздел, следует этот раздел сделать активным

```
fdisk440 -e /dev/rdisk0
```

```
>f 9
```

```
>w
```

```
>q
```

Девятка во второй строке – это опять номер раздела (их всего четыре!) – делайте вывод.

Теперь можно на этот раздел скопировать файл boot и папку EFI в корень раздела.

Установка на раздел FAT32.

В отличие от предыдущего метода здесь есть одна тонкость. Сектор PBR должен содержать геометрию раздела. Эти сведения туда заносятся в процессе разбивки на разделы, поэтому потеря такой информации чревата последствиями. Сам же метод установки сектора усложняется

```
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 - уже упоминался в главе "Что есть что" – сектор для установки на раздел FAT32. Но не FAT16! Будьте внимательны!

rdisk1s9 – опять девятый раздел на первом устройстве. Подставьте свои цифры.

Остальные буквы и цифры в этом рецепте обсуждению и пересмотру не подлежат.

Остальные действия аналогичны установке на HFS+.

Linux

Под линуксом также имеется терминал, и почти такие же команды, но установка возможна только на FAT32. Отличия следующие.

- вместо rdisk1 будет sdb – смотрите в вашей версии Линукса точнее.
- вместо fdisk440 для записи MBR нужно использовать тот же 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

Из-под Виндоуз также есть смысл ставить загрузчик только на флешку FAT32, для этого достаточно запустить скрипт

makeusb.bat E:

где E: - это буква вашей флешки. Наличие нескольких разделов на ней не предполагается. Это же Виндоус!

Все файлы, необходимые для скрипта прилагаются в комплекте с Кловером. Однако, инсталлятор надо предварительно распаковать, либо получить эти файлы непосредственно с svn. Они лежат в папке

edk2/Clover/CloverPackage/CloverV2/BootSectors

После выполнения скрипта флешку нужно извлечь и вставить заново, затем скопировать на нее файл boot и папку EFI.

Еще лучше воспользоваться BootDiskUtility.exe by Cvad которая поможет сформировать флешку из-под Windows.

<http://www.applelife.ru/threads/bootdiskutility-exe-создаем-clover-boot-usb-flash-disk-под-windows.37189/>

Конфигурирование аппаратной части

Файл config.plist

Вообще, Кловер проделывает это автоматически. Но автомат никогда не бывает совершенным, поэтому пользователь имеет возможность менять различные параметры через файл config.plist, либо просто в меню графического интерфейса. Это файл формата xml, однако, в данный момент удобно его рассматривать как текстовый файл. Редактировать этот файл можно текстовым редактором или специализированной программой типа PListEditor. Вместе в Кловером распространяется два варианта этого файла – максимальный и почти минимальный. Совсем минимальный файл – пустой.

Общее правило – если вы не знаете, какое значение следует дать какому-то параметру, исключите этот параметр вообще из файла. Не оставляйте параметр без значения! И уж тем более, не ставьте значения, которого не понимаете!

Все параметры объединены по группам: SystemParameters, SMBIOS, CPU, Graphics, PCI, ACPI, KernelAndKextPatches, Pointer, Volumes. Рассмотрим поподробнее.

SystemParameters

```
<key>boot-args</key>
<string>-v arch=i386</string>
```

Это аргументы, которые передаются в boot.efi, а он, в свою очередь, часть их передает ядру системы. Конкретный список аргументов ядра следует искать в документации Apple. Список аргументов, необходимых самому boot.efi вообще незадокументирован, известны следующие

Kernel=mach_kernel

Mkext=extensions.mkext

Кроме того, самому Кловеру нужно, чтобы именно туда был записан ключ WithKexts, если нужно загрузить дополнительные кексты.

Этот ключ анализируется драйвером FSInject.efi, и если он присутствует происходит следующее:

- игнорируется наличие kernelcache и mkext, все кексты загружаются заново;
- загружаются дополнительные кексты из папки EFI/kexts/10.7 – конкретная папка выбирается в соответствии с версией загружаемой ОС.

<key>prev-lang:kbd</key>

<string>ru:0</string>

На данный момент установка языка имеет смысл только для меню "Help" вызываемого по клавише F1.

<key>DefaultBootVolume</key>

<string>MacHDD</string>

Здесь указывается имя раздела, с которого должна производиться загрузка системы по-умолчанию. Вообще этот раздел не обязан совпадать с разделом, куда установлен загрузчик. Наличие этого ключа обязательно, если вы хотите иметь полные сервис с загрузкой по таймауту, и с перезагрузкой из контрольной панели. *А вот имя должно быть свое, а не то, которое приведено здесь! И уж тем более не то, что по-умолчанию* " <string>Name of the volume of your HD Mac Boot</string> "

<key>CustomUUID</key>

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

Уникальный идентификационный номер вашего компьютера. Если вы не поставите этот ключ, будет сгенерен какой-то из аппаратных сведений, если же вы хотите полный контроль над происходящим, напишите свои 16-чные цифры.

Но, ради бога, не копируйте мои образцовые цифры! Они давно уже не уникальны, дураков полно, которые их скопировали!

<key>InjectSystemID</key>

<string>No</string>

Этот же номер будет инжектирован другим способом, и в свойствах системы будет преобразован во что-то другое. Смысл в этой операции в том, чтобы точно повторить UUID, сгенеренный Хамелеоном. Для этого ставим Yes, а в качестве CustomUUID используем то значение, которое присутствует с Хамелеоном в реестре IODeviceTree:/efi/platform=>system-id. Тогда в профайлере мы увидим другое значение, но такое же, как и раньше.

<key>LegacyBoot</key>

<string>PBR</string>

LegacyBoot, необходимый для запуска старых версий Windows и Linux. очень сильно зависит от аппаратной части, от построения БИОСа, поэтому разработаны несколько алгоритмов, и выбор алгоритма производится в этом ключе. Варианты:

LegacyBiosDefault – для тех UEFI BIOS, где есть протокол LegacyBios.

PBRtest , **PBR** – варианты алгоритма PBR boot, кому с каким повезет.

```
<key>BacklightLevel</key>
```

```
<string>0x0101</string>
```

Это свойство инжектируется в систему, и система знает о его существовании. Однако влияние заметно только на очень редких конфигурациях. Что это? Яркость монитора... как следует из названия. Это свойство также считывается из NVRAM, и, по-умолчанию, используется то значение (которое выставила система). Значение же, прописанное в конфиге, или выставленное в меню, будет перебивать значение по-умолчанию.

```
<key>iCloudFix</key>
```

```
<string>Yes</string>
```

Этот параметр появился для того, чтобы в системе была возможность зарегистрироваться в iCloud и iMessage. В настоящее время Yes стоит по-умолчанию, и нет необходимости прописывать его в конфиге. Да и не видно смысла, чтобы отключать его.

SMBIOS

Эта группа параметров нужна для мимикрии вашего PC под Mac. Кловер это сделает автоматически, основываясь на обнаруженной модели CPU, видеокарте, и признаке мобильности. Однако, вы можете захотеть и другой выбор. Берите программу MacTracker и подбирайте модель Мака, которая вам больше нравится, а затем ищите по интернету, или по знакомым все номера и серийники от этой модели. Комментировать тут особо нечего. Эти параметры не для чайников. Знаете их – меняйте, наугад не получится. Вычислить их тоже нельзя.

Параметр **SmUUID** введен для проверки, влияет ли это на что-нибудь. Это UUID, который пишется в SMBIOS таблицу 1. Тесты не выявили никакого влияния.

Отдельное слово про параметр

```
<key>FirmwareFeatures</key>
```

```
<string>0xC0001403</string>
```

Эти цифры выходят за рамки стандарта SMBIOS, это нечто, специфичное для Эппл. В разных настоящих Маках можно встретить разные цифры, описания нигде никакого нет, разве что в исходниках bless можно найти

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

Следовательно, и нам надо иметь здесь нечетное число.

```
<key>BoardType</key>
```

```
<string>10</string>
```

Этот параметр введен для МакПро, у которого здесь стоит не 10 — Motherboard, а 11 — ProcessorBoard, видимо по историческим причинам. Смысл неочевиден, но на Систем Профайлере это заметно.

```
<key>Mobile</key>
```

```
<string>Yes</string>
```

Вообще-то, Кловер всегда правильно вычисляет, является ли данная платформа мобильной (т.е. с питанием от аккумуляторов, требующее экономии энергии), или же нет. А параметр нужен, если по какой-то причине мы хотим обмануть систему, указать, что никакой батарейки у нас нет, либо наоборот.

CPU

Эта группа параметров помогает с определением ЦПУ, когда внутренние алгоритмы не справляются.

<key>Turbo</key>

<string>Yes</string>

Добавляет одну ступень по частоте для процессора. К примеру был $200 \times 12 = 2400$, стал $200 \times 13 = 2600$. Работает не для всех аппаратных конфигураций. Есть процессора, которые официально не поддерживают эту технологию.

<key>CpuFrequencyMHz</key>

<string>3200</string>

Базовая частота процессора в МГц. Обычно Кловвер получает это значение из DMI, но если там неверно, можно подставить через этот ключ. **Неправильное значение может вызвать неправильную работу системы – рассинхронизацию, тормоза и т.п. Лучше вообще ничего не пишите, даже самого ключа.**

<key>BusSpeedkHz</key>

<string>133330</string>

Этот параметр – базовая частота шины, является критически важной для работы системы, и передается из загрузчика в ядро. **Если частота неправильная, ядро вообще не запускается, если частота немного не соответствует, могут возникнуть проблемы с часами, и очень странное поведение системы.**

Значение в DMI хранится в МГц, и это неточно, более правильно вычисленное из частоты ЦПУ, ну а вы можете подобрать значение более точное, и прописать его в этом ключе в килогерцах. К примеру, у меня в ДМИ написано 100МГц, а для часов лучше стало, когда я прописал 99790кГц.

Один момент. Некоторые производители имеют другое понятие, что есть BusSpeed, а что есть FSBSpeed, и вписывают в БИОС значение в четыре раза больше. Разобраться в правильности можно по диапазону: оно должно быть от 100 до 400МГц, либо по формуле $\text{ЧастотаЦПУ} = \text{ЧастотаШины} * \text{МножительЦПУ}$.

Понятно, что если АСУС пишет частоту шины 1600МГц, да множитель процессора 8, то формула не сойдется, процессоров на 12,8ГГц не существует. Реально надо делить на 4.

<key>QPI</key>

<string>4800</string>

В системном профайлере эта величина называется Processor Bus Speed или просто Bus Speed, на некоторых реальных Маках она оказалась совпадающей с QPI. На сайте Интел нет никакого внятного объяснения, как она вычисляется для той или иной модели процессора, а к некоторым процессорам вместо нее дается DMI speed. В Хамелеоне есть алгоритм ее вычисления для процессоров семейства Nehalem. Реально почти всегда получается $\text{BusSpeed} * 18$, и это где-то сходится с табличным значением DMI speed! В исходниках кекста AppleSmbios рассматриваются два варианта: либо значение уже прописано в SMBIOS, как там изготовитель прописал, либо просто вычисляется $\text{BusSpeed} * 4$. После долгих споров эта величина вынесена в конфиг – пишите что вам нравится (МГц). На работу это никак не влияет – чистой воды косметика.

<key>ProcessorType</key>

<string>0x0201</string>

Этот параметр придуман Apple и используется в окошке «Об этом Маке», внутренними средствами переводящим такую константу в обозначение процессора. Иначе будет показан «Неизвестный процессор». Почему нельзя было вызвать CPUID? Ну или в SMBIOS посмотреть в таблице 4? Нет, у Эппл свое мировоззрение, а нам приходится приспосабливаться, какой процессор как зашифрован. В основном Кловер знает все шифры, но, поскольку прогресс не стоит на месте, то оставлена возможность вручную изменить этот параметр. Правильность установки этого параметра контролируется в окошке «About this Mac». Опять-таки, косметика чистой воды.

Graphics

Эта группа параметров служит для инъектирования свойств видеокарточки, как это делает, к примеру, Natit.kext. Параметров, которые реально инъектируются много, но это в основном константы, некоторые вычисляемые, некоторые заданы в таблице, и только совсем отдельные параметры вводятся через конфиг.

<key>GraphicsInjector</key>

```
<string>Yes</string>
```

Собственно включение этой функции инъекции. Кстати, по умолчанию она включена, ибо инъекция должна работать при чистом конфиге – условие запуска системы. Выключать инъекцию стоит в том случае, если вы знаете лучший способ.

<key>VRAM</key>

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

Объем видеопамати в Мб. Вообще-то он и автоматически определяется, но если пропишете правильное значение – никто не пострадает.

<key>LoadVBios</key>

```
<string>No</string>
```

Загрузка видеобиоса из файла, который должен лежать в папке EFI/OEM/xxx/ROM или EFI/ROM и иметь имя файла vendor_device.rom, например 1002_68d8.rom. Это иногда имеет смысл, если использовать патченный видеобиос. Также бывают проблемы, что видеокарточка не показывает системе свой видеобиос, хотя система и требует, например в случае мобильных радеонов. В этом случае можно поставить этот параметр в Yes, но никакого файла не подсовывать. Кlover возьмет ВидеоБИОС из легаси-памяти по адресу 0xc0000, как ни странно, он там практически всегда есть, и теперь Clover его инжектирует в систему, и мобильный радеон включается!

<key>PatchVBios</key>

<string>No</string>

Когда-нибудь Кlover научится и сам править видеобиос на лету. В данный момент не работает (ревизия 659).

<key>InjectEDID</key>

```
<string>Yes</string>
```

1. Существуют мониторы без DDC, например панели ноутбуков.
2. Существует варианты, когда ЕДИД есть, но Эппловские драйвера его не видят. На второй вариант мы ставим просто InjectEDID=yes, и Кловер сам извлечет ЕДИД, и подsunет его драйверам. Необходимость такого действия подмечена в теме про мобильные радеоны.

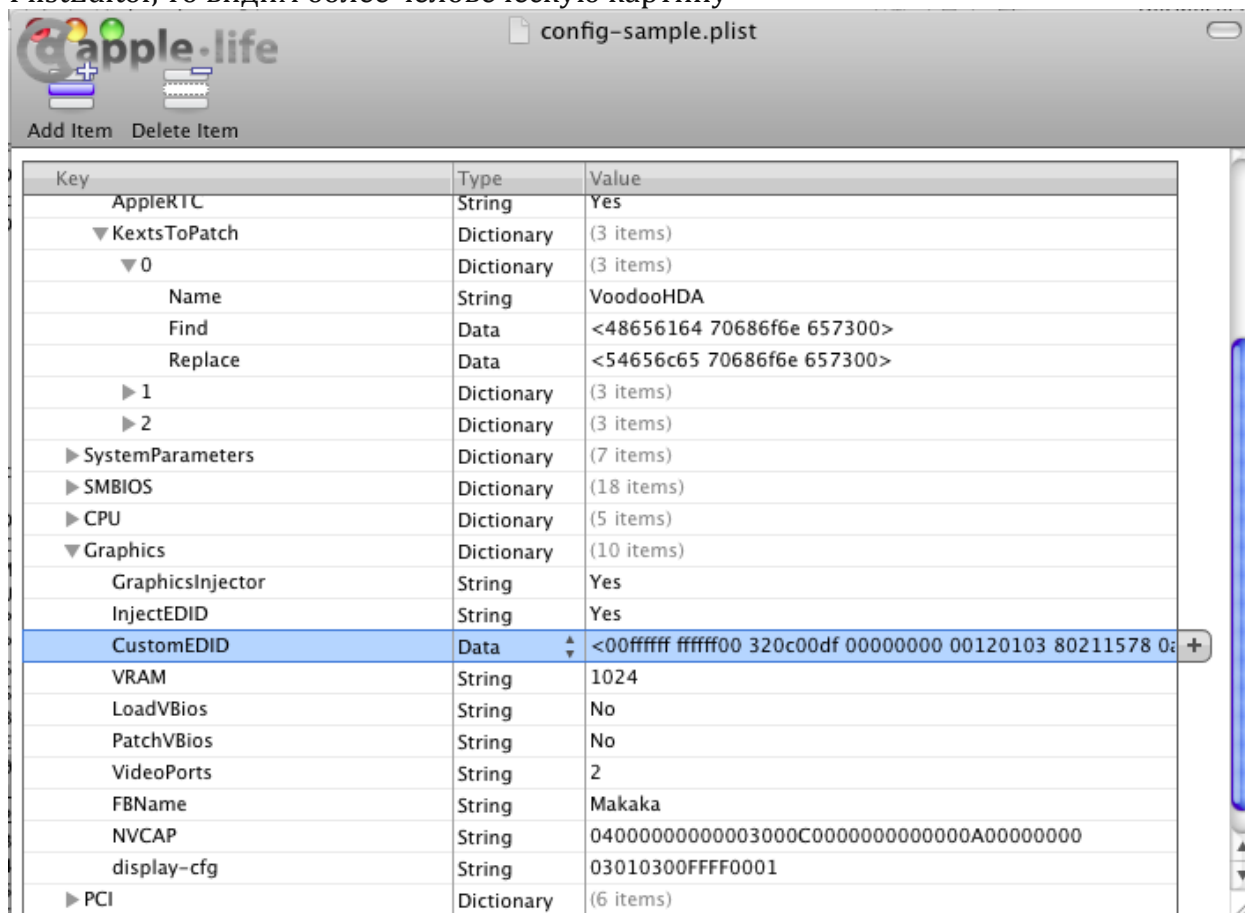
На первый вариант мы еще должны прописать новый едид ручками. Пишем так:

<key>CustomEDID</key>

```
<data>AP////////wAyDADfAAAAAASAQOAIRV4CunVmVlTjigmUFQAAAABAQEBAQEBAQE  
EBAQEBAQEB3iGgcFCEHzAgIFYAS88QAAAY3iGgcFCEHzAgIFYAS88QAAAAAAAA/gBXNjU3  
RwAxNTRXUDEKAAAA/gAiMz1IZYSq/wIBCiAgAJ0=</data>
```

Откуда взять? Ну раз из этого компа он никак не извлекается, то берем чужой. Главное условие - чтобы было правильное максимальное разрешение. В свой образец конфиг-листа я ставлю ЕДИД от Делла Инспирон. Матрица 1440x900.

Буковки в этом примере — это стандартная шифровка XML, если смотреть через PlistEditor, то видим более человеческую картину



Еще вариант изготовления ЕДИДа — воспользоваться программой ViewSonic EDID Editor (версия 3.1.5), которая, при желании, легко портируется в OSX. Но это уже не касается собственно Кловера. Изучайте теорию. Кловер дает вам возможность инжектировать свой EDID, хороший, качественный.

```
<key>VideoPorts</key>
<string>2</string>
```

Количество видеовыходов на карте, включая TVO и/или HDMI.

```
<key>FBName</key>
<string>Makaka</string>
```

Этот параметр специфичен для ATI Radeon, к которым имеется три десятка разных фреймбуферов без какой-либо закономерности кому что. Кловер автоматически выбирает из таблицы на большинство известных карточек наиболее подходящее имя. Однако, другие пользователи точно такой же карточки оспаривают, им нужно другое имя. Вот и напишите в этот параметр то, что вам кажется наиболее правильным. Общее правило: не знаете, что писать, вообще сотрите параметр. Но не пишите уж эту макаку! Специально прописал для абсурда – нет, все равно копируют в свой конфиг!

```
<key>NVCAP</key>
<string>040000000000003000C0000000000000A000000000</string>
```

Это параметр для видеокарт NVidia, конфигурирует типы и назначения видеопортов. В этой строке 40 шестнадцатеричных цифр заглавными буквами. Теория здесь отсутствует, есть эмпирика, да еще и с противоречивыми результатами. Есть вот такая табличка, но ее правильность оспаривается.

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
Desktop 1	0	0	0	1	1	1	111
Desktop 2	1	1	1	0	0	0	111000
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		

На форумах можно найти другие способы вычисления правильного значения этой строки.

<key>display-cfg</key>

<string>03010300FFFF0001</string>

Это тоже параметр только для карт NVidia. Подробности смотрите в обсуждениях <http://www.projectosx.com/forum/index.php?showtopic=1105>

Однако, сведения, приведенные там являются спорными. Реальные конфиги можно посмотреть в теме <http://www.projectosx.com/forum/index.php?showtopic=370>. А вообще-то, конфиг по-умолчанию, который создает Кловер, похож и является лучшим вариантом. Просто не указывайте вообще этот параметр, дайте возможность Кловеру его вычислить.

KernelAndKextPatches

Эта группа параметров для осуществления бинарных патчей на лету. Надо заметить, что это осуществимо, только если загрузка происходит через kernelcache либо через параметр WithKexts. Если кеш не загрузился по другим причинам, то эти фиксы не работают.

`<key>KernelCpu</key>`

`<string>Yes</string>`

Предотвращает панику ядра на неподдерживаемом ЦПУ, в частности Yonah, Atom, IvyBridge.

Нужно понимать, что в ядре есть и другие алгоритмы, которые будут неправильно работать с неподдерживаемым ЦПУ, поэтому не ждите, что этот патч решит все ваши проблемы. Очень сомнительно, что это будет работать с Pentium M, Pentium 4 или Amd. для таких случаев лучше все же найти специально сделанное ядро.

`<key>AsusAICPUPM</key>`

`<string>No</string>`

Оказывается, БИОС на материнских платах ACYC (который раз нам ACYC портит настройку?) что-то делает, что MSR регистр 0xE2 становится ReadOnly, но он используется в кексте **AppleIntelCPUPowerManagement**, причем используется по записи. Авторы этого фикса не придумали ничего лучшего, как исправить сам кекст, ибо как вернуть регистру E2 его былую функциональность, увы, никто не знает. Ставьте Yes, если при старте системы вы имеете панику на этот кекст.

`<key>AppleRTC</key>`

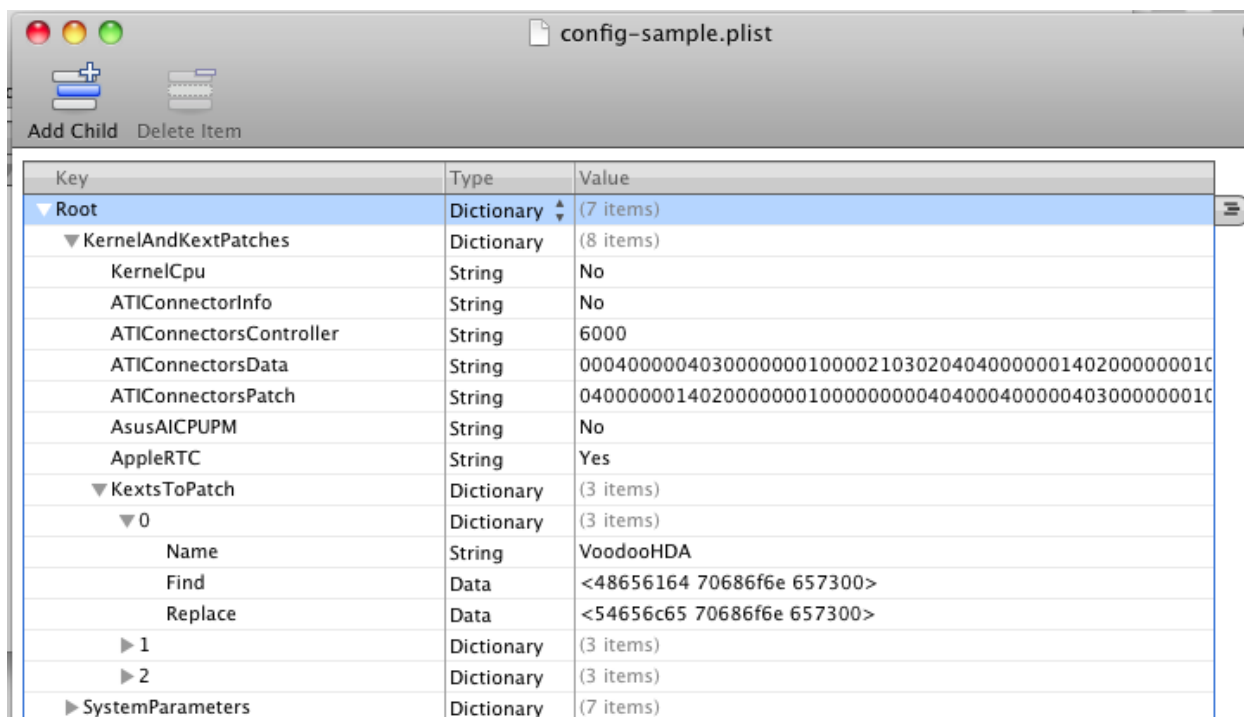
`<string>Yes</string>`

Операционная система OSX как-то не так работает с CMOS, как это предусмотрено BIOSом, в результате при пробуждении из сна или при перезагрузке происходит сброс CMOS. Не у всех, больше в этом грехе замечены платы от Gigabyte. Более того, часто эта проблема решается просто патчем DSDT: Deivce(RTC) что делает и Кловер. Однако, в некоторых случаях и этот патч не помогает. Тогда можно поправить сам кекст AppleRTC, что здесь и делается.

`<key>KextsToPatch</key>`

`<dict>`

Помимо специфических патчей, можно сделать патч любого другого кекста, принцип простой: 16чная строка, что искать, и строка, на что заменить. Образец патчим VoodooHDA на предмет замены названия Headphones на Telephones. Условие — количество букв должно быть таким же. Либо меньше и дополнить нулями.



Этот метод успешно применяется для включения поддержки Trim для SSD
<http://www.applelife.ru/threads/clover.32052/page-539#post-310105>

<key>ATIConnectorsController</key>

```
<string>6000</string>
```

Для полноценного запуска карточек ATI (AMD) Radeon 5000 и 6000 серий недостаточно инжектировать свойства в реестр, необходимо еще подкорректировать коннекторы в соответствующем контроллере. В данном случае указываем на 6000 контроллер. Следующие два свойства указывают, что найти, и на что изменить.

<key>ATISConnectorsData</key>

```
<string>000400000403000000010000210302040400000014020000000100000000  
04031000000001000000000001000000000001</string>
```

<key>ATIConnectorsPatch</key>

```
<string>0400000014020000001000000004040004000004030000001000011020105000000000000000000000000000000</string>
```

Этот метод работает только для систем 10.7 и выше.

Расскажу подробнее, как получить эти цифры.

Оригинальная статья от bcc9

<http://www.insanelymac.com/forum/index.php?showtopic=249642>

Полный рецепт от Xmedik на русском языке с обсуждениями

<http://www.applelife.ru/threads/Завод-ати-hd-6xxx-5xxx-4xxx.28890/>

Здесь изложу вкратце, с учетом специфики Кловера.

1. Прежде всего, надо получить свой видеобиос. Загрузиться в CloverGUI и нажать F6. Ваш Биос будет сохранен в файле `/EFI/misc/c0000.bin`, если, конечно, Кловер установлен в раздел с файловой системой FAT32.
2. Загрузите по одной из этих ссылок программу `radeon_bios_decode`. В ту же папку с этой утилитой положите файл биоса `c0000.bin`. Допустим, это папка `~/RadeonPatch`

Выполняем в терминале следующие команды

```
cd ~/RadeonPatch
./radeon_bios_decode < c0000.bin
```

3. На экране вы получите информацию по вашим коннекторам, которую стоит скопировать/сфотографировать для дальнейшего использования.

Вот что у меня

```
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. Загрузите по одной из ссылок скрипт **ati-personality.pl**

5. Положите в эту же папку, и выполните в терминале
`perl ati-personality.pl -386 >frames.txt`

если вы делаете это для 32-битной системы, или
`perl ati-personality.pl >frames.txt`
для 64-битной.

6. Теперь нужно определиться с выбором подходящего фреймбуфера. Эппл предлагает нам широкий выбор: и птички, и рыбки, и даже обезьяны. Но реальные отличия там в основном в коннекторах, которые мы и собираемся изменить. Если не слишком задумываться, то минимальные рекомендации следующие:

ATI5000 – мобильный Galago, десктопный – Baboon

ATI6000 – мобильный Osmunda, десктопный – Promoea

7. Для выбранного фреймбуфера берем распечатку коннекторов из нашего файла frames.txt, полученного на шаге 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
```

Красным цветом выделены цифры, которые необходимо править. Синие цифры – просто адреса, нужно отбросить. Третья цифра с конца – encoderid, последняя цифра — senseid. Первые 4 цифры в каждой строке – тип монитора (точнее сказать тип коннектора).

```
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-V
00 04 00 00  DP
00 08 00 00  HDMI
```

- senseid мы получили на шаге 3 для каждого из наших коннекторов. encoder можно просто всюду занулить. На остальные цифры не обращаем внимания. Получаем следующую таблицу:

```
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
```

Т.е. Первая строка DVI-D, вторая – VGA, третья – HDMI, и все с моими значениями senseid.

- Отбросив синие цифры остальные вписываем в config.plist без пробелов и переносов строк. Исходная таблица в ATICConnectorsData, после наших правок в ATICConnectorsPatch. Смотрите образец выше по тексту.

PCI

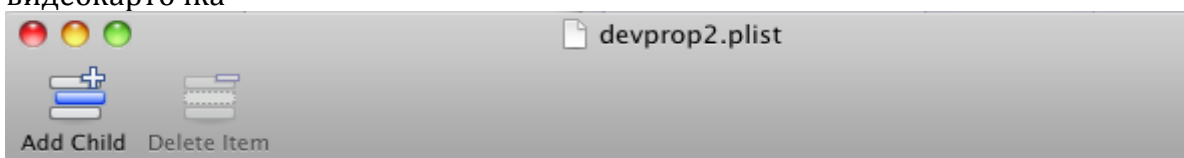
Группа параметров для остальных PCI устройств и шины вообще.

```
<key>StringInjector</key>
<string>No</string>
```

Поставить это значение в Yes –вся внутренняя инжекция заменяется на ввод единой строки DeviceProperties

```
<key>DeviceProperties</key>
<string>0207364862FA54HG345</string>
```

Как сделать свою строку? Для этого вам нужен gxfutil, который не входит в стандартную поставку Кловера, зато входит в комплект DarwinDumper. Сначала надо создать нужный xml файл с группами параметров, имеющими в качестве заголовка DevicePath по стандартной нотификации, соответствующий устройству, чьи свойства вы собрались инжектировать. Вот, к примеру, встроенная видеокарточка



Key	Type	Value
▼ Root	Dictionary (1 item)	
▼ PciRoot(0x0)/Pci(0x2,0x0)	Dictionary (5 items)	
AAPL,HasPanel	Data	<01000000>
built-in	Data	<00>
class-code	Data	<00000300>
device_type	Data	<64697370 6c617900>
model	Data	<474d4120 39353000>

Можно, к примеру, с помощью DarwinDumper посмотреть, какой plist сгенерировал Кловвер по-умолчанию, затем поправить его, преобразовать в вид 16-чной строки командой

```
./gfxutil -i xml -o hex devprop.plist devprop.hex
```

и получится текстовый файл вида

```
d30000000100000001000000c70000000500000002010c00d041030a0000000001010600000027fff04001
00000006d006f00640065006c0000000c000000474d4120393530001c0000006400650076006900630065
005f00740079007000650000000c000000646973706c617900200000004100410050004c002c004800610
07300500061006e0065006c0000000800000001000000160000006200750069006c0074002d0069006e00
000005000000001a00000063006c006100730073002d0063006f0064006500000008000000000000300
```

Этот текст нужно скопировать в значение параметра DeviceProperties.

В принципе, такой же результат достигается и вставкой методов _DSM в DSDT, если он уже есть, и если заниматься его совершенствованиями. Кому как. Если же ДСДТ еще нет, а готовые строки для ваших карточек уже есть, то почему бы и не воспользоваться таким методом?

<key>PCIRootUID</key>

<string>0</string>

Оказывается, инъекция свойств видеокарты зависит от того, какое число стоит в DevicePath=PciRoot(0x0) или PciRoot(0x1). Ранее считалось, что это аппаратная характеристика. Однако, еще на заре хакинтошестроения выяснилось, что это число – просто идентификатор, прописанный в DSDT. Вот здесь:

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

    Name (_UID, Zero)
```

_UID=Zero – значит 0, если же равно One, значит 1.

Причем, если это число поменять насильно, оно поменяется, и будет успешно работать. Так вот, настоящие Маки имеют всегда 0. И соответственно boot.efi всегда предполагает 0, поэтому лучше, если поправить свой ДСДТ, чтобы там тоже получился 0, и ничего не писать в конфиге, значение 0 стоит по-умолчанию. И в таком раскладе наличие этого параметра следует считать устаревшим методом.

<key>HDAInjection</key>

<string>Detect</string>

Инъекция свойств звуковой карточки. Правда, это работает, только если устройство в ДСДТ называется HDEF, если же заниматься его переименованием, то и остальное можно другим способом инжектировать. Также эти усилия не нужны при использовании драйвера VoodooHDA.

Варианты следующие:

NO – ничего не инжектируется.

Detect – автоматическое определение установленной звуковой микросхемы, чтобы ее ID употребить в качестве лейаута. Вообще-то бред, но очень популярный. Во многих случаях не мешает, и оказывает влияние на отображение звуковой карточки в Систем-Профайлере.

883 – в десятичном виде номер лейаута. Имеется ввиду Realtek ALC883.

0x373 – тоже самое в 16-чном виде становится неузнаваемым.

На самом деле эти числа неправильные, правильный лэйаут 12 = 0x0C, но, как ни странно, являются допустимыми.

<key>USBInjection</key>

<string>Yes</string>

Можно поставить No, если вы почему-то хотите отказаться от инъекции свойств USB. В некоторых конфигурациях они не нужны. Инжекция свойств ЮСБ также отключится, если прописана маска патча ДСДТ 0x1000. Смысл в том, чтобы не дублировать такую инъекцию и патч в ДСДТ.

<key>LpcTune</key>

<string>Yes</string>

Разрешать или нет инициализацию чипа контроллера LPC. Согласно документу от Интел в чипсете ICH8M есть устройство LPC controller, в котором регистр A0 имеет следующие значения битов:

12 -> 0 = Enable C4

11 -> 1 = Enable C6

7 -> 1 = Enter C4 when C3 invoked – выполнить C4, когда запрошен C3.

3 -> 1 = Intel SpeedStep Enable

Эти биты определены только для мобильного чипсета, поэтому на десктопах параметр не имеет смысла. А если честно, то и на ноутбуках влияния не замечено.

Pointer

Начиная с версии 724 в интерфейсе Кловера завелась мышь, и, соответственно, ее параметры нужно уметь регулировать.

<key>Speed</key>

<string>8</string>

Наиболее часто встречающееся значение. Больше цифра — больше скорость.

<key>DoubleClickTime</key>

<string>500</string>

Время в миллисекундах между двумя кликами, меньше которого считается за двойной клик. При этом смещение мыши не учитывается — возможны баги.

Volumes

Эта группа параметров служит для скрытия некоторых элементов главного меню. Вообще за внешний вид отвечает файл refit.conf, но в данном случае речь идет не столько о внешнем виде, сколько о свойстве всей системы.

Поскольку в главном меню значатся не только разделы (как в Хамелеоне), но и отдельные загрузчики, которых может быть несколько в одном разделе, то в конфиге мы прописываем какого типа загрузчиков мы не желаем видеть HideAllOSX, HideAllOSXInstall, HideAllRecovery, HideAllUEFI, HideAllWindowsEFI, HideAllGrub, HideAllGentoo, HideAllRedHat, HideAllUbuntu, HideAllSuSe, HideAllLegacy, к примеру

<key>HideAllUbuntu</key>

<string>Yes</string>

по умолчанию все No.

Кроме того, можно погасить отдельные разделы, если вам оттуда нечего грузить.

<key>HideVolumes</key>

<dict>

<key>0</key>

<dict>

<key>VolumeString</key>


```
<string>HD(1,GPT,E223FF7F-F2DA-4DBB-B765-756F2D95B0FE</string>
</dict>
</dict>
```

Это строка, однозначно идентифицирующая раздел (даже если он не имеет названия, или имеет типичное название `untitled`, берется из бут-лога, где перечислены все разделы, которые видит загрузчик, и как он их видит. Первый ключ 0 — просто нумерация в файле конфига. 0, 1, 2, 3....

ACPI

Группа параметров, регулирующих коррекцию различных ACPI таблиц. И дело не только в том, что у Мака свои требования, но и просто разные версии ACPI спецификации, и элементарная лень производителей, и просто в БИОСе системной платы нет сведений об установленных картах и ЦПУ (а динамически определить слабо? Кловер же это делает!).

```
<key>DropOemSSDT</key>
<string>Yes</string>
```

Поскольку мы собираемся создавать или подгружать динамически свои таблицы SSDT, то нужно избежать ненужного пересечения интересов. Этот параметр позволяет отбросить все родные таблицы, в пользу новых. У вас есть и такой вариант: подложить родные таблицы с небольшими правками в папку `EFI/OEM/xxx/ACPI/patched/`, а непатченные дропнуть.

```
<key>GenerateCStates</key>
<string>Yes</string>
```

Автоматическая генерация таблицы SSDT, дополняющей процессорную секцию методами `_CST` для каждого процессора. На формирование самого метода `_CST` влияют параметры **EnableC2**, **EnableC4**, **EnableC6**, **EnableISS**, **C3Latency**. Собственно эти параметры нечего комментировать, включено или нет, все равно все работает, так что экспериментируйте. Кроме того, Кловер уже вычислил, какой имеется процессор, и сколько у него ядер. То, что этот параметр вступил в силу, контролируется по кернел-логу. Без этого метода присутствует ошибка *ACPI_SMC_PlatformPlugin::pushCPU_CSTData - _CST evaluation failed*.
Отдельное слово про

```
<key>C3Latency</key>
<string>0x03E9</string>
```

Это задержка на включение C3 state. Критическое значение `0x3E8=1000`. Меньше — включается спидстеп, больше — не включается. На нативниках всегда `0x03E9`, то есть спидстеп не работает. На Хаках приходится выбирать, что мы хотим, быть похожим на нативника, или включить управление питанием. Разумное значение во втором случае — `0x00FA`, как встречается на некоторых ноутбуках.

```
<key>GeneratePStates</key>
<string>Yes</string>
```

Автоматическая генерация таблицы SSDT, дополняющей процессорную секцию методами `_PPC`, `_PCT` и `_PSS`.

_PCT – Performance Control – контроль за управлением спидстепом.

_PPC – Performance Present Capabilities – возможности спидстепа, Эта функция возвращает одно число, которое означает ограничение частоты. Подробности ниже, в параметре **PLimitDict**.

_PSS – Performance Supported States – набор возможных состояний процессора – P-states. Этот массив формируется на основе данных о процессоре, которые Кловер уже вычислил, а также с учетом параметров пользователя **PLimitDict**, **UnderVoltStep** и **Turbo**. А именно, Turbo=Yes и PLimitDict=1 компенсируют друг друга.

```
<key>PLimitDict</key>
```

```
<string>1</string>
```

Суть параметра очень проста – ограничить максимальную частоту процессора. Значение 0 – работа до максимума, 1 – на одну ступень меньше максимума, 2 – на две ступени. Пример: Core2Duo T8300 2400MHz работает на максимальной частоте 2000, если ограничить на две ступени. Зачем? Да чтобы ноутбук не перегревался, там возможности ЦПУ намного превышают возможности по охлаждению. Точно такой же параметр присутствует в платформ-пLISTах, например:

```
/
System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns/ACPI_SMC_PlatformPlugin.kext/Contents/Resources/MacBook5_1.plist
```

Далее мы еще обсудим эти пLISTы.

Для некоторых процессоров, например Core2Quad, замечено, что PlimitDict работает наоборот, и лучший вариант =1.

```
<key>UnderVoltStep</key>
```

```
<string>1</string>
```

Дополнительный параметр для снижения температуры процессора путем снижения его рабочего напряжения. Возможные значения 0, 1, 2, 3 ... чем больше, тем сильнее охлаждаем, пока компьютер не повиснет. В этом месте работает защита против дурака, Кловер не позволит поставить значение вне допустимого диапазона, вернее пишите, что хотите, а работать будет только то, что дозволено. Впрочем и дозволенные значения могут давать неустойчивую работу. Эффект от этого параметра реально наблюдается.

```
<key>ResetAddress</key>
```

```
<string>0x64</string>
```

```
<key>ResetValue</key>
```

```
<string>0xFE</string>
```

Эти два параметра служат для одного очень ценного фикса – исправление рестарта. Эти значения должны быть в таблице FADT, но почему-то они там не всегда есть, более того, бывает и сама таблица короче необходимого, короче настолько, что эти значения оказались отброшены. По умолчанию идет эта пара **0x64/0xFE** что означает рестарт через PS2 контроллер. Практика показала, что это не у всех работает, другая возможная пара значений **0x0CF9/0x06**, что означает рестарт через PCI шину. Эта пара используется и на нативнике, но не всегда работает на хакинтошах. Разница понятна, на хакинтошах есть еще и PS2 контроллер, который может помешать рестарту, если его не сбросить. Единого рецепта нет, автодетекта нет, поэтому эти параметры прописываются через конфиг.

```
<key>smartUPS</key>
```

```
<string>No</string>
```

Вообще-то, этот параметр предназначен для того, чтобы прописать в таблице FADT профиль питания=3. Логика следующая:

PM=1 – desktop , питание от сети

PM=2 – notebook, питание от сети или от батарейки

PM=3 – server, питание от SmartUPS, про который MacOSX тоже что-то знает.

Выбор между 1 и 2 Кlover сделает на основе анализа бита мобильности, но также есть и параметр **Mobile** в секции SMBIOS. Можно, к примеру, сказать, что у нас МакМини, и что он мобильный. Значение же 3 будет подставлено, если **smartUPS=Yes**.

`<key>PatchAPIC</key>`

`<string>No</string>`

На некоторых компьютерах можно загрузить систему только с `crus=1`, либо со специальным патченным ядром (Lapic NMI patch). Простейший анализ показал, что у них неправильная таблица MADT, а именно, в ней отсутствуют разделы NMI. Этот параметр служит для корректировки таких таблиц на лету. Для здорового компьютера ничего плохого не произойдет.

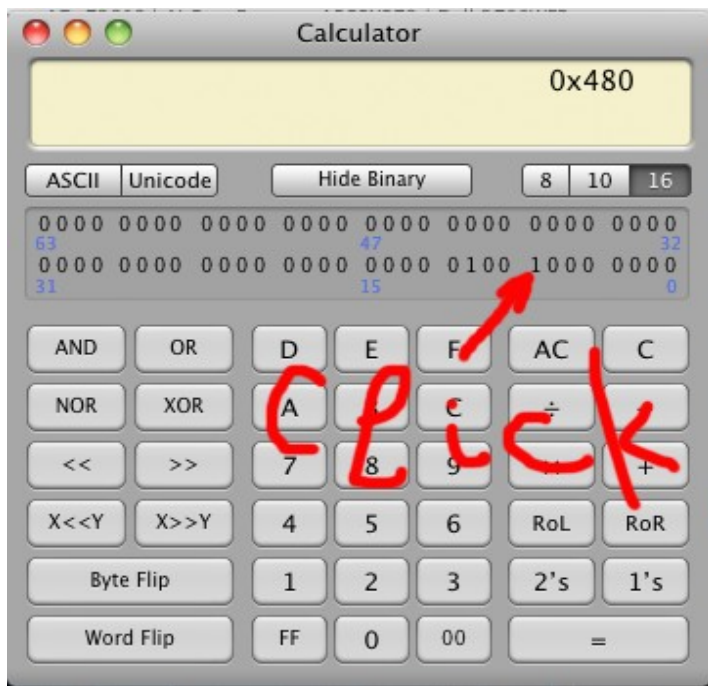
`<key>FixDsdtMask</key>`

`<string>0xFFFF</string>`

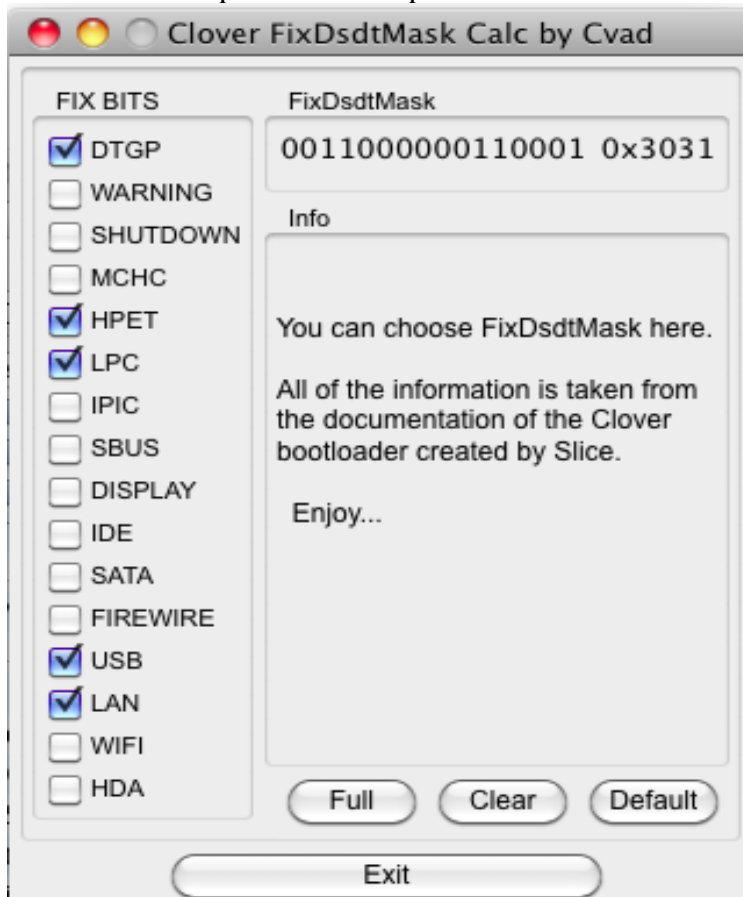
Под этим параметром скрывается сразу 16 патчей для таблицы DSDT, по числу битов в маске FFFF. Перечень патчей следующий

```
//0x00FF
#define FIX_DTGP      bit(0)
#define FIX_WARNING   bit(1)
#define FIX_SHUTDOWN  bit(2)
#define FIX_MCHC      bit(3)
#define FIX_HPET      bit(4)
#define FIX_LPC       bit(5)
#define FIX_IPIC      bit(6)
#define FIX_SBUS      bit(7)
//0xFF00
#define FIX_DISPLAY   bit(8)
#define FIX_IDE       bit(9)
#define FIX_SATA      bit(10)
#define FIX_FIREWIRE  bit(11)
#define FIX_USB       bit(12)
#define FIX_LAN       bit(13)
#define FIX_WIFI      bit(14)
#define FIX_HDA       bit(15)
```

Чтобы подсчитать, как сумма битов складывается в ту или иную маску, можно вызвать системный калькулятор, перевести его в вид для программиста, и переключить на 16-чные числа. А теперь щелкая мышью по битам с 0 по 15 мы наберем нужную маску.



Есть более современный вариант: CloverFixDsdMaskCalculator by Cvad



<http://www.applelife.ru/attachments/cloverfixdsdtmaskcalculator-app-zip.43973/>
А вот для того, чтобы рассказать, что означают эти фиксы, придется открыть новую главу.

Корректировка DSDT

DSDT - Differentiated System Description Table – самая большая и самая сложная ACPI таблица. Минимальная длина 36 байт, реальная – 20кб, бывают варианты и больше. Эта таблица описывает устройства и методы доступа к ним. Методы доступа могут содержать арифметические и логические выражения, и, таким образом, представляют собой программу на некоем языке программирования, похожим на C своими фигурными скобками. Исправлять эту таблицу значит что-то понимать в программировании. Кловер предлагает некий вариант автоматической правки, но надо понимать, что искусственный интеллект еще не создан, и автоматическая корректировка программ пока еще далека до совершенства. Человек сделает лучше.

А зачем ее нужно исправлять? Весь патч ДСДТ, со времен его основания был нацелен в первую очередь на исправление устройства HPET – High Precision Events Timer. Дело в том, что в системе OSX присутствует кекст AppleIntelCPUPowerManagement, который служит, чтобы управлять питанием процессора (спидстеп), и которому строго необходимо, чтобы в системе был HPET, имеющий прерывания IRQ0 и 8. Без этого условия кекст уходит в панику. Работать можно только запретив, или удалив этот кекст. Но есть и другой вариант – скорректировать DSDT, и устройство HPET включится как положено!

Это – патч №1, жизненная необходимость. Только ли МакОСу нужен этот HPET? Нет, конечно, но производители БИОСов только еще начинают это осознать и прописывать правильные параметры, до сих пор редко встретишь, чтобы ДСДТ работал без патча.

Момент №2. В ДСДТ можно разглядеть некоторые зависимости от операционной системы, "Windows 98", "Windows 2001", "Windows 2006", "Linux", МакОС имеет идентификатор "Darwin", и, как правило, на него ДСДТ не рассчитан. А даже если и рассчитан, то на версию типа FreeBSD. MacOSX является серьезной ACPI системой, т.е. использует ДСДТ по максимуму, так, как использует Windows 2001, но не Linux, не Windows 98, и не Windows 2006. Правильнее всего сделать мимикрию под Windows 2001. И даже если у вас уже есть "Darwin", добейтесь, чтобы он работал как "Windows 2001". **На многих БИОСах этому соответствует значение OSYS = 0x07D2. Но не 7D6, не 7D9, и уж никак не 0x2410, как это прописано в нативнике.**

Момент №3. Производитель системной платы, а с ней и его БИОСа, и его ДСДТ, не может предусмотреть, какой процессор будет установлен, какая видеокарточка и другие PCI устройства. А ведь их стоит прописать в ДСДТ! И наоборот, исключить из ДСДТ такие устройства как спикер, флоппи дисковод, параллельный порт. Драйверов на них нет и не нужны. Также часто необходимо добавлять или убавлять коннекторы/порты у каких-то устройств, например, у видеокарты, или у SATA контроллера.

DSDT лежит в БИОСе и используется в системе в бинарном коде AML, существует компилятор/декомпилятор IASL, который переводит коды в понятный для человека язык DSL. Человеческий путь правки такой AML->DSL->edit->DSL->AML. И тут возникает момент №4. Последняя компиляция становится невозможной из-за ошибок, синтаксических и логических, изначально присутствующих в OEM DSDT. В процессе правки требуется и их исправлять. Ну а заодно исправить и смысловые ошибки, из-за которых, например, компьютер не может заснуть, или не может проснуться. А может еще и новые устройства прописать. (А вообще странно, но компиляция/декомпиляция не являются строго обратными операциями, туда-

обратно меняет таблицу, а то и вообще, туда идет, обратно нет — требуется вмешательство).

Когда мы проделали весь этот путь, мы можем подсунуть загрузчику наш исправленный ДСДТ, положив его в папку /EFI/OEM/xxx/ACPI/patched, или, если OEM имя компьютера еще не известно, в папку /EFI/ACPI/patched, или загружаемая система сама имеет свой вариант ДСДТ, лежащий в корне системного диска.

Где взять исходный ДСДТ, который необходимо патчить? Есть варианты добыть его используя Windows, Linux или даже OSX. если его как-то удалось запустить, но теперь и Кловер предоставляет такую возможность. Надо войти в графическое меню и нажать клавишу F4. Если Кловер установлен на раздел FAT32, то ему удастся сохранить все OEM ACPI таблицы, включая нетронутые DSDT и FADT. Будьте терпеливы, если сохранение происходит на флешку, и таблиц много, процесс может занять заметное время. Теперь можно брать этот файл и редактировать... Ну а для тех, кто не силен в языке DSL, Кловер предлагает проделать некоторые фиксы автоматически. Рассмотрим подробнее.

FIX_DTGP bit(0)

Для описания свойств устройства, кроме варианта DeviceProperties, рассмотренном выше, есть вариант с методом _DSM, прописанным в DSDT.

_DSM - Device Specific Method – хорошо известна заготовка этого метода, которая работает в MacOSX, этот метод содержит массив с описанием устройства и вызов универсального метода **DTGP**, который един для всех устройств.

Данный фикс просто добавляет этот метод, чтобы потом его применять для других фиксов. Самостоятельного значения не имеет.

FIX_WARNING bit(1)

Мировой накопленный опыт по корректировке DSDT содержит ряд типичных ошибок типичных производителей (в основном ACUC, чьи ДСДТ на редкость кривые). Перечислять, пожалуй, не будем. Фикс рекомендуемый для всех.

FIX_SHUTDOWN bit(2)

В функцию _PTS добавляется условие: если аргумент = 5 (выключение), то никаких других действий делать не надо. Странно, а почему? Тем не менее, есть неоднократные подтверждения эффективности это патча для плат ACUC, может и для других. В некоторых ДСДТ такая проверка уже есть, в этом случае такой фикс следует отключить.

FIX_MCHC bit(3)

Автор всей методики патча ДСДТ — rcj - поставил этот фикс для себя. Он создаёт устройство с DeviceID=0x0044, что соответствует IntelHD2000. Такое устройство класса 0x060000, как правило, отсутствует в ДСДТ, но для некоторых чипсетов это устройство является обслуживаемым, а поэтому его нужно прописать, чтобы правильно развести управление питанием PCI шины. Вопрос о необходимости патча решается экспериментально.

FIX_HPET bit(4)

Как уже сказано, это главный фикс. Кроме собственно устройства HPET, этот фикс затрагивает также устройства RTC и TMR, и попутно решает проблему сброса БИОСа после пробуждения.

Таким образом, минимально необходимая маска патча ДСДТ выглядит как 0x0010

FIX_LPC bit(5)

Подменяет DeviceID LPC контроллера, чтобы кекст AppleLPC прицепился к нему. Нужен для тех случаев, когда чипсет не предусмотрен для OSX. Впрочем, родной список чипсетов Intel и NForce настолько большой, что необходимость такого патча очень редка. Проверяется в системе, загружен ли кекст AppleLPC, если нет – патч нужен. Хотя, это тоже еще не факт. Бывает, что кекст сам выгружается из памяти за ненадобностью, хотя чипсет поддерживаемый.

FIX_IPIC bit(6)

Удаляет прерывание из устройства IPIC. Назначение такого фикса спорно. Для ноутбуков скорее надо не удалять, а добавлять такое прерывание.

FIX_SBUS bit(7)

Добавляет SMBusController в дерево устройств, тем самым удаляя предупреждение об его отсутствии из системного лога. И также создает правильную разводку управления питанием шины.

FIX_DISPLAY bit(8)

Производит ряд патчей и инъекцию свойств для видеокарточки. В принципе, этот метод включения видео имеет больше возможностей, чем через DeviceProperties, поскольку имеет возможность не только прописать свойства через метод _DSM, но также и изменить другие ACPI свойства дисплея. Например, для ноутбуков Делл строго требуется удалять устройство CRT из описания видеокарточки.

FIX_IDE bit(9)

В системе 10.6.1 появилась паника на кекст AppleIntelPIIXATA. Два варианта решения проблемы – использование исправленного кекста, либо исправить устройство в ДСДТ. А для более современных систем? Не знаю, пусть будет.

FIX_SATA bit(10)

Фиксирует какие-то проблемы с SATA, и убирает желтизну иконок дисков в системе путем микриии под ICH6. Вообще-то спорный метод, однако без этого фикса у меня DVD-диски не проигрываются, а ля ДВД привод не должен быть съемным.

FIX_FIREWIRE bit(11)

Добавляет свойство "fwhub" к контроллеру Firewire.

FIX_USB bit(12)

Попытки решения многочисленных проблем с USB. Инжектирование DeviceProperties в настоящее время дает более правильное решение.

FIX_LAN bit(13)

Инжектирование свойства "built-in" для сетевой карточки – необходимо для правильной работы. Также инжектируется модель карточки – для косметики.

FIX_WIFI bit(14)

Аналогично LAN, кроме того, создается само устройство, если еще не прописано в DSDT.

FIX_HDA bit(15)

Корректировка описания звуковой карточки в ДСДТ, чтобы работал родной драйвер AppleHDA. Производится переименование AZAL -> HDEF, инжектируется layout-id и PinConfiguration.

Послесловие

Как выбрать, какие патчи необходимы, какие безвредны, а какие опасны? Ну компьютер вы не погубите ни в каком случае. Все это происходит только в оперативной памяти, и будет забыто после перезагрузки. Можно испытывать набор фиксов, исправляя маску в графическом меню, и сохраняя результат по клавише F5 – "Сохранить DSDT-xxxx.aml, скорректированный по текущей маске".

Можно попытаться и загрузиться с текущей маской. Чтобы не мешался настоящий, патченный ДСДТ, уже присутствующий в системе, можно указать в меню DSDT name: NO.aml

Не найдя такого файла система возьмет OEM DSDT из БИОС и проделает над ним фиксы, согласно установленной маске. В случае неудачи после перезагрузки компьютера текущие установки будут утеряны, и в силу вступят установки по умолчанию, которые у вас работоспособные.

Маска 0xFFFF соответствует включению всех фиксов, и если ОС после этого загрузится, труд программистов потрачен не зря. По описанию выше вы уже сообразили, что некоторые фиксы вам просто ни к чему (например WIFI), а вот попортить картину могут. Для большинства не очень новых, но и не очень старых конфигураций маска 0xA7D7 является достаточной, и даже избыточной.

Нативный спидстеп

Правильнее говорить Управление Питанием и Частотой Процессора (УПиЧП), по-английски это будет EIST – Enhanced Intel Speedstep Technology, откуда и русское слово "Спидстеп".

Собственно эта тема не столько для загрузчика, как вообще для настройки ХакОС, но поскольку Кловер делает некоторые шаги, то опишем отдельной главой. Кловер делает не все, что нужно, требуется и немного поработать руками.

Для чего это вообще нужно? Смысл такой: процессор в бездействии работает на минимальной частоте с минимальным напряжением, под нагрузкой скорость и напряжение растут. (А напряжение-то зачем? А потому что фронт импульса становится круче, и потому быстрее набирает уровень, быстрее переходит из состояния 0 в состояние 1).

УПиЧП можно осуществить двумя способами: специализированной утилитой, типа КулБукКонтроллер, или ДжениерикЦПУПМ, или же понять нативный спидстеп, благо МакОС это умеет делать.

Следующие шаги необходимы:

1. В ДСДТ обязательно должен быть поправлен HPET, что успешно делается Кловером при маске 0x0010.
2. Должна быть правильная процессорная секция, что делается Кловером при ключе GeneratePStates=Yes (ну и вдобавок DropOemSsdtd=Yes)
3. Должна быть выбрана MacModel как образец вашего SMBIOS, для которого предусмотрена технология EIST. Оказывается, не для всех

моделей. К примеру для модели MacBook1,1 спидстеп работать не будет, а для MacBook5,1 – будет.

Пункт 3 можно переосмыслить следующим образом: пусть, все-таки модель будет более похожа по конфигурации к настоящей, но исправим ее платформ-плист так, чтобы спидстеп появился.

Для каждой модели существует свой plist, смотрите здесь

/System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns/ACPI_SMC_PlatformPlugin.kext/Contents/Resources/MacBook5_1.plist

Смотрим сходства и различия разных plistов, и исправляем свой в правильную сторону.

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>
```

Этот ключ restart-action означает на какой P-State должен свалиться CPU при рестарте. Только при наличии этого ключа заработали сон и выключение компьютера!

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>
```

Этот ключ PLimitDict уже упоминался в генерации P-states. Повторим: это ограничение максимальной скорости процессора. 0 – скорость максимальна, 1- на одну ступень ниже максимальной. Если же этот ключ здесь отсутствует, то процессор застрянет на минимальной частоте.

CStateDict

```
<key>CStateDict</key>
<dict>
  <key>MacBook4,1</key>
```

```
<string>CSD3</string>
<key>CSD3</key>
<dict>
  <key>C6</key>
  <dict>
    <key>enable</key>
    <true/>
```

Практика показывает, что эту секцию лучше всю удалить, чтобы работало управление питанием именно по PState, а не по CState. Хотя, кому как, может и этот вариант стоит проработать.

Симптом – процессор стоит на максимальной частоте, не падает. После удаления секции начинает варьировать частоту.

Проблема сна

А что проблема сна? Когда все вышесказанное будет сделано, компьютер будет ложиться спать и просыпаться, как послушный ребенок. Самое главное, необходимое для этого, Кловер уже проделал: скорректировал FADT и FACS. Осталось только поправить ДСДТ, завести спидстеп, пользоваться только хорошими кекстами, и будем вам счастье.

Хорошему сну может мешать любое устройство, в том числе незаведенное PCI устройство, или заведенное частично. К примеру, AppleHDA. Сну категорически мешает NullCPUPM.kext. Вам, может, спидстеп и не нужен, но вы должны так сделать патч HPET, чтобы запустился родной AppleCPUPM, и нуль был не нужен.

В ДСДТ есть группа методов _GPE с нотификациями на каждое устройство, которое нужно пробудить после сна. Сам-то компьютер проснулся, а вот может оказаться, что видео/сеть/звук/мышь забыли проснуться. Смотрите ДСДТ, учите теорию, как это делать.

ЧаВо

Часто задаваемые **В**опросы.

В. Хочу попробовать Кловер, с чего начать?

О. С чтения этой книги.

ЗЫ. Странно писать это внутри книги, но может эти ЧаВо окажутся вне ее страниц.

В. Не работает.

О. Сам дурак.

ЗЫ. Ну а что тут еще ответишь?

В. Установил Кловер, но получаю черный экран.

О. Загрузка ОС происходит в восемь этапов (см. Стр.6). Будьте добры, уточните, на каком именно этапе происходит остановка. И в своем отчете обязательно укажите «Устанавливал инсталлятором с выбором таких опций». Тогда и будет разговор.

В. Вижу на экране 7_ и больше ничего не происходит.

О. Это самый тяжелый случай несовместимости по железу. Встречается сейчас уже очень редко. Продиагностировать сможет только программист, который сможет вставлять в коды Кловера отладочные сообщения, и делать ребут за ребутом до полного выяснения проблемы. Увы, простым пользователям посоветовать нечего.

В. Происходит загрузка только до текстового аналога БИОСа с пятью пунктами, верхний – Continue>

О. Это означает, что файл boot успешно загрузился, и работает, но не находит файла CloverX64.efi. То ли того раздела не видит, то ли вообще устройства – надо разбираться далее, прогулявшись по опциям этого меню.

В. Установил Кловер на флешку, загрузился с нее, и не вижу своего HDD.

О. Во-первых, HDD надо вставить в порт Sata0. В будущем может быть это будет исправлено. Во-вторых я понимаю, если у вас есть хорошо работающий Хам, Химера, ХРС, короче, ББХ (Бутер на Букву Х), вы не хотите его убивать, но хотите попробовать Кловер, то такой поступок кажется естественным. Но, тем не менее, есть варианты установки Кловера на жесткий диск, не убивающие старого загрузчика, и в таком раскладе озвученная ошибка пропадет.

В. При УЕФИ-загрузке не вижу раздела с МакОСью, только легаси.

О. Это означает, что в папке /EFI/drivers64UEFI отсутствует HFSPlus.efi

В. При УЕФИ-загрузке Виндоус выглядит как легаси, хотя он EFI.

О. То же самое, отсутствует драйвер NTFS.efi

ЗЫ. Эти два драйвера отсутствуют в репозитории по лицензионным причинам, поэтому эти две ошибки возникают у тех, кто не желает использовать инсталлятор, все пытается «я сам».

В. При попытке запуска ОСи зависает после синей строки с надписью rEFIt

О. В этот момент происходит патч ДСДТ с вашей маской. Да, в идеале тут не должно виснуть. Но проблема в том, что очень много производителей БИОСов не соблюдает стандарты, не умеют программировать, и не желают отшлифовать свой ДСДТ под нужды OSX. Очень легко убедиться, что операция декомпилировать-снова скомпилировать не проходит – ДСДТ кривой. Кловер желал бы все это исправить, но увы, количество плохих вариантов пока не поддается даже обзору. Поэтому, от вас требуется подобрать такую маску фикса ДСДТ, чтобы загрузчик не повис, а затем и чтобы ОСь не повисла, а в идеале, чтобы она еще и работала. Это – реально. Либо отказаться от автопатча (маска = 0), а ДСДТ сделать вручную.

В. Ядро начинает грузиться, но паникует после десятой строки.

О. Это отсутствующий, или неправильный ДСДТ. Если автопатчем не получается, добавьте ДСДТ, сделанный вручную.

В. Система начинает грузиться, но стопорится на still waiting for root device....

О. Кроме обычного для таких случаев совета включить AHCI в БИОСе, или, если такого нет, найти правильный драйвер (в смысле кекст) для вашего IDE контроллера, тут есть еще совет загрузиться с ключом WithKexts, тогда загрузка пойдет медленнее, и контроллер успеет включиться. Кстати, такая ошибка может возникнуть только если Кловер и система находятся на разных устройствах.

В. Система грузится до сообщения: Waiting for DSMOS....

О. Отсутствует FakeSMC. Может быть с Хамелеоном у вас этот кекст лежал в Экстре, а Кловер этой папки не видит. Для него предназначена папка /EFI/kexts/10.7 или другие.

В. Система проходит это сообщение, но дальше ничего не меняется, хотя винчестер жужжит, как будто система грузится.

О. Типичная ситуация, когда не включилась видеокарта. Пробуйте GraphicInjector=Yes в конфиге, либо наоборот =No. Во втором варианте Радеоны запускаются на «нативной заводке», которая позволяет даже работать в системе, за небольшими исключениями, например DVDplayer не будет работать. Для полной же заводки Радеона требуется еще и коннекторы поправить. Для других случаев можно попытаться загрузить систему с ключом -x, и войти на десктоп в режиме VESA. Не очень здорово, но зато позволит что-то исправить.

В. Система загрузилась, все хорошо, но в Систем Профайлере ошибки...

О. Вообще это косметика, на функциональность не влияет.

О платах PCI. Если желаете информацию, добавьте в ДСДТ в нужное устройство свойство Name (_SUN, 0x1234) – число на ваше усмотрение. Оно отобразится в профайлере.

О памяти. Есть две величины скорости, номинальная и фактическая, и они часто не совпадают. Какую показать в профайлере? Поставил первую – заорали, что это неверно. Поставил вторую, эти замолчали, другие пользователи заорали, что это неправильно. Идите все в жопу. Сейчас Кловер показывает номинальную скорость.

Заключение

Кловер, конечно, еще далек до идеала, но процесс совершенствования программ никогда не бывает завершенным. Будут новые ревизии, будут новые функции, а пока так.