Практическое знакомство со свободной альтернативой BIOS+UEFI

Nick Void (mn3m)
mn3m00@gmail.com

June 26, 2015

http://mn3m.info/

План

План выступления:

1. why
2. internals
3. howto
4. demo

coreboot - зачем?

Coreboot – Зачем?

Для чего человеку в 201x году думать о system firmware (System BIOS):

1. баги на уровне firmware;

Coreboot – Зачем?

Для чего человеку в 201x году думать о system firmware (System BIOS):

1. баги на уровне firmware;
2. желание расширить возможности firmware;

Coreboot – Зачем?

Для чего человеку в 201x году думать о system firmware (System BIOS):

1. баги на уровне firmware;
2. желание расширить возможности firmware;
3. желание иметь свободную firmware;

Coreboot – Зачем?

Для чего человеку в 201x году думать о system firmware (System BIOS):

1. баги на уровне firmware;
2. желание расширить возможности firmware;
3. желание иметь свободную firmware;
4. security;

Coreboot – Зачем?

Что должен сделать System BIOS:

- инициализировать аппаратное обеспечение;

Coreboot – Зачем?

Что должен сделать System BIOS:

- инициализировать аппаратное обеспечение;
- инициализировать нормальную работу SMM и ACPI;

Coreboot – Зачем?

Что должен сделать System BIOS:

- инициализировать аппаратное обеспечение;
- инициализировать нормальную работу SMM и ACPI;
- предоставить базовые функции для Legacy OS;

Coreboot – Зачем?

Что должен сделать System BIOS:

- инициализировать аппаратное обеспечение;
- инициализировать нормальную работу SMM и ACPI;
- предоставить базовые функции для Legacy OS;
- загрузить ~~ROMBASIC~~ ОС;

Coreboot – Зачем?

Что должен сделать System BIOS:

- инициализировать аппаратное обеспечение;
- инициализировать нормальную работу SMM и ACPI;
- предоставить базовые функции для Legacy OS;
- загрузить ~~ROMBASIC~~ ОС;
- иногда - предоставить функции собственной ОС UEFI;

Coreboot – Зачем?

Что должен сделать System BIOS:

- инициализировать аппаратное обеспечение;
- инициализировать нормальную работу SMM и ACPI;
- предоставить базовые функции для Legacy OS;
- загрузить ~~ROMBASIC~~ ОС;
- иногда - предоставить функции собственной ОС UEFI;
- иногда - предоставить идентификационные данные для некоторых ОС (OEM);

Coreboot – Зачем?

Что должен сделать System BIOS:

- инициализировать аппаратное обеспечение;
- инициализировать нормальную работу SMM и ACPI;
- предоставить базовые функции для Legacy OS;
- загрузить ~~ROMBASIC~~ ОС;
- иногда - предоставить функции собственной ОС UEFI;
- иногда - предоставить идентификационные данные для некоторых ОС (OEM);
- иногда - предоставить функции восстановления себя и/или предустановленной ОС;

Coreboot – Зачем?

Основные представители System BIOS для x86:

PC BIOS	EFI	Coreboot
Assembly 8086	many	96% ANSI C, 1% asm
real mode	protected mode	protected mode
Closed	Open	GPLv2

Table 1: Сравнение основных BIOS'ов

Coreboot – Зачем?

Сложность темы:

- нет подстраховки
  (no one will catch your exceptions)

Coreboot – Зачем?

Сложность темы:

- нет подстраховки
  (no one will catch your exceptions)
- множество аппаратно-зависимой vendor specific информации
  (которая не всегда есть, не всегда открыта и не всегда соответствует действительности)

Coreboot – Зачем?

Сложность темы:

- нет подстраховки
  (no one will catch your exceptions)
- множество аппаратно-зависимой vendor specific информации
  (которая не всегда есть, не всегда открыта и не всегда соответствует действительности)
- программирование без RAM;

Coreboot – Зачем?

Сложность темы:

- нет подстраховки
  (no one will catch your exceptions)
- множество аппаратно-зависимой vendor specific информации
  (которая не всегда есть, не всегда открыта и не всегда соответствует действительности)
- программирование без RAM;
- программирование без стека;

Coreboot – Зачем?

Сложность темы:

- нет подстраховки
  (no one will catch your exceptions)
- множество аппаратно-зависимой vendor specific информации
  (которая не всегда есть, не всегда открыта и не всегда соответствует действительности)
- программирование без RAM;
- программирование без стека;
- ассемблер, ANSI C;

Coreboot – Зачем?

Сложность темы:

- нет подстраховки
  (no one will catch your exceptions)
- множество аппаратно-зависимой vendor specific информации
  (которая не всегда есть, не всегда открыта и не всегда соответствует действительности)
- программирование без RAM;
- программирование без стека;
- ассемблер, ANSI C;
- real mode programming;

Coreboot – Зачем?

Сложность темы:

- нет подстраховки
  (no one will catch your exceptions)
- множество аппаратно-зависимой vendor specific информации
  (которая не всегда есть, не всегда открыта и не всегда соответствует действительности)
- программирование без RAM;
- программирование без стека;
- ассемблер, ANSI C;
- real mode programming;
- тестирование;

Coreboot – Зачем?

некоторые security векторы:

- System BIOS имеет неограниченный доступ к hardware;

Coreboot – Зачем?

некоторые security векторы:

- System BIOS имеет неограниченный доступ к hardware;
- при cold reboot RAM не очищается;

Coreboot – Зачем?

некоторые security векторы:

- System BIOS имеет неограниченный доступ к hardware;
- при cold reboot RAM не очищается;
- код обработчика SMM выплняется на уровне >= ядра ОС и НЕ является частью ОС;

Coreboot – Зачем?

некоторые security векторы:

- System BIOS имеет неограниченный доступ к hardware;
- при cold reboot RAM не очищается;
- код обработчика SMM выплняется на уровне >= ядра ОС и НЕ является частью ОС;
- в env ОС UEFI можно записать из user space;

Coreboot – Зачем?

некоторые security векторы:

- System BIOS имеет неограниченный доступ к hardware;
- при cold reboot RAM не очищается;
- код обработчика SMM выплняется на уровне >= ядра ОС и НЕ является частью ОС;
- в env ОС UEFI можно записать из user space;
- GPU умеет ходить в RAM в обход CPU через DMA;

Coreboot – Зачем?

Решение некоторых секурити проблем:

- ~~использовать бумагу и печатную машинку, а всю компьютерную технику сжечь~~
  Использовать Свободную firmware;

Coreboot – Зачем?

Решение некоторых секурити проблем:

- ~~использовать бумагу и печатную машинку, а всю компьютерную технику сжечь~~
  Использовать Свободную firmware;
- Разрабатывать свободные альтернативы проприетарного инициализирующего кода и проприетарных драйверов

Coreboot – Зачем?

Решение некоторых секурити проблем:

- ~~использовать бумагу и печатную машинку, а всю компьютерную технику сжечь~~
  Использовать Свободную firmware;
- Разрабатывать свободные альтернативы проприетарного инициализирующего кода и проприетарных драйверов
- Развивать идею Secure Linux Boot

Coreboot internals

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)

1. перейти в protected mode

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)

1. перейти в protected mode
2. скопировать initramfs с GNU/Linux в RAM

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)

1. перейти в protected mode
2. скопировать initramfs с GNU/Linux в RAM
3. jump

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)

```
Linuxbiosmain(){
   void (*v)() = 0x100000;
   unsigned char *rom = 0xfff00000;
   memcpy(v, &rom, SIZE);
   (v + 0x20)();
   /* NOTREACHED */
}
```

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)
Плюсы:

- Busybox with BASH shell in BIOS

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)
Плюсы:

- Busybox with BASH shell in BIOS
- Protected mode ”из коробки”

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)
Плюсы:

- Busybox with BASH shell in BIOS
- Protected mode ”из коробки”
- Native Linux file systems support

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)
Плюсы:

- Busybox with BASH shell in BIOS
- Protected mode ”из коробки”
- Native Linux file systems support
- Меньше задач на Coreboot, больше на Linux

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)
Плюсы:

- Busybox with BASH shell in BIOS
- Protected mode ”из коробки”
- Native Linux file systems support
- Меньше задач на Coreboot, больше на Linux
- Меньше задач - меньше кода

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)
Плюсы:

- Busybox with BASH shell in BIOS
- Protected mode ”из коробки”
- Native Linux file systems support
- Меньше задач на Coreboot, больше на Linux
- Меньше задач - меньше кода
- Меньше кода - меньше багов

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)
Минусы:

- не работает на ”современном” железе из-за Vendor specific условий инициализации RAM, South Bridge

Coreboot – internals

Первые шаги загрузки Coreboot(LINUXBIOS) v0.01 (1999)
Минусы:

- не работает на ”современном” железе из-за Vendor specific условий инициализации RAM, South Bridge
- не работают PCI устройства из-за необходимости их дополнительно инициализировать

Coreboot – internals

”Современный” вариант загрузки Coreboot(LINUXBIOS) v3 и v4 для Intel архитектуры x86

1. переход в Protected Mode

Coreboot – internals

”Современный” вариант загрузки Coreboot(LINUXBIOS) v3 и v4 для Intel архитектуры x86

1. переход в Protected Mode
2. mainboard init 1 (FPU enable, SSE enable, Cache as RAM)

Coreboot – internals

”Современный” вариант загрузки Coreboot(LINUXBIOS) v3 и v4 для Intel архитектуры x86

1. переход в Protected Mode
2. mainboard init 1 (FPU enable, SSE enable, Cache as RAM)
3. mainboard init 2 (Vendor Specific code, Southbridge, init RAM (ROM stage)

Coreboot – internals

”Современный” вариант загрузки Coreboot(LINUXBIOS) v3 и v4 для Intel архитектуры x86

1. переход в Protected Mode
2. mainboard init 1 (FPU enable, SSE enable, Cache as RAM)
3. mainboard init 2 (Vendor Specific code, Southbridge, init RAM (ROM stage)
4. copy decompressed Coreboot in RAM

Coreboot – internals

”Современный” вариант загрузки Coreboot(LINUXBIOS) v3 и v4 для Intel архитектуры x86

1. переход в Protected Mode
2. mainboard init 1 (FPU enable, SSE enable, Cache as RAM)
3. mainboard init 2 (Vendor Specific code, Southbridge, init RAM (ROM stage)
4. copy decompressed Coreboot in RAM
5. jump to RAM entry point (start RAM stage)

Coreboot – internals

”Современный” вариант загрузки Coreboot(LINUXBIOS) v3 и v4 для Intel архитектуры x86

1. переход в Protected Mode
2. mainboard init 1 (FPU enable, SSE enable, Cache as RAM)
3. mainboard init 2 (Vendor Specific code, Southbridge, init RAM (ROM stage)
4. copy decompressed Coreboot in RAM
5. jump to RAM entry point (start RAM stage)
6. Initialize console, enumerate devices, ACPI Table, SMM handler (RAM stage)

Coreboot – internals

”Современный” вариант загрузки Coreboot(LINUXBIOS) v3 и v4 для Intel архитектуры x86

1. переход в Protected Mode
2. mainboard init 1 (FPU enable, SSE enable, Cache as RAM)
3. mainboard init 2 (Vendor Specific code, Southbridge, init RAM (ROM stage)
4. copy decompressed Coreboot in RAM
5. jump to RAM entry point (start RAM stage)
6. Initialize console, enumerate devices, ACPI Table, SMM handler (RAM stage)
7. jump to payload entry

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;
5. TianoCore (UEFI);

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;
5. TianoCore (UEFI);
6. Etherboot / GPXE / iPXE;

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;
5. TianoCore (UEFI);
6. Etherboot / GPXE / iPXE;
7. OS as payload: Linux, NetBSD, PLAN9;

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;
5. TianoCore (UEFI);
6. Etherboot / GPXE / iPXE;
7. OS as payload: Linux, NetBSD, PLAN9;
8. Depthcharge, Uboot, Explorer (Chromebooks payloads);

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;
5. TianoCore (UEFI);
6. Etherboot / GPXE / iPXE;
7. OS as payload: Linux, NetBSD, PLAN9;
8. Depthcharge, Uboot, Explorer (Chromebooks payloads);
9. Games: Tint, Invaders;

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;
5. TianoCore (UEFI);
6. Etherboot / GPXE / iPXE;
7. OS as payload: Linux, NetBSD, PLAN9;
8. Depthcharge, Uboot, Explorer (Chromebooks payloads);
9. Games: Tint, Invaders;
10. Tests and info: Memtest86, Memtest86+, CoreInfo, nvramcui;

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;
5. TianoCore (UEFI);
6. Etherboot / GPXE / iPXE;
7. OS as payload: Linux, NetBSD, PLAN9;
8. Depthcharge, Uboot, Explorer (Chromebooks payloads);
9. Games: Tint, Invaders;
10. Tests and info: Memtest86, Memtest86+, CoreInfo, nvramcui;
11. Bayou (load menu for multiple payloads);

Coreboot – internals

Payloads:

1. SeaBIOS (old name - ADLO);
2. OpenBIOS;
3. OpenFirmware;
4. GRUB2, FILO;
5. TianoCore (UEFI);
6. Etherboot / GPXE / iPXE;
7. OS as payload: Linux, NetBSD, PLAN9;
8. Depthcharge, Uboot, Explorer (Chromebooks payloads);
9. Games: Tint, Invaders;
10. Tests and info: Memtest86, Memtest86+, CoreInfo, nvramcui;
11. Bayou (load menu for multiple payloads);
12. Libpayload;

Own Payload with libpayload:

```
#include <libpayload.h>

int main(void)
{
	printf("Hello, world!\n");
	halt();
	return 0;
}
```

Coreboot – internals

История проекта CoreBoot:

1. существует с 1999 года как свободная GPLv2 firmware для кластерных решений (LINUXBIOS);

Coreboot – internals

История проекта CoreBoot:

1. существует с 1999 года как свободная GPLv2 firmware для кластерных решений (LINUXBIOS);
2. 95,7% - ANSI C, 1,4% - Assembly, rest - C++, Perl, Shell, text files

Coreboot – internals

История проекта CoreBoot:

1. существует с 1999 года как свободная GPLv2 firmware для кластерных решений (LINUXBIOS);
2. 95,7% - ANSI C, 1,4% - Assembly, rest - C++, Perl, Shell, text files
3. May/2015 219 new commits per 30 days (7.3 commits/day, 1 commit every 197 mins)

Coreboot – internals

История проекта CoreBoot:

1. существует с 1999 года как свободная GPLv2 firmware для кластерных решений (LINUXBIOS);
2. 95,7% - ANSI C, 1,4% - Assembly, rest - C++, Perl, Shell, text files
3. May/2015 219 new commits per 30 days (7.3 commits/day, 1 commit every 197 mins)
4. с 2008 называется CoreBoot т.к. может загружать не только Linux, но и BSD и PLAN9

Coreboot – internals

История проекта CoreBoot:

1. существует с 1999 года как свободная GPLv2 firmware для кластерных решений (LINUXBIOS);
2. 95,7% - ANSI C, 1,4% - Assembly, rest - C++, Perl, Shell, text files
3. May/2015 219 new commits per 30 days (7.3 commits/day, 1 commit every 197 mins)
4. с 2008 называется CoreBoot т.к. может загружать не только Linux, но и BSD и PLAN9
5. с 2010 разрабатывается для Google для Chrome буков под x86

Coreboot – internals

История проекта CoreBoot:

1. существует с 1999 года как свободная GPLv2 firmware для кластерных решений (LINUXBIOS);
2. 95,7% - ANSI C, 1,4% - Assembly, rest - C++, Perl, Shell, text files
3. May/2015 219 new commits per 30 days (7.3 commits/day, 1 commit every 197 mins)
4. с 2008 называется CoreBoot т.к. может загружать не только Linux, но и BSD и PLAN9
5. с 2010 разрабатывается для Google для Chrome буков под x86
6. http://coreboot.org

Coreboot – internals

Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS

Coreboot – internals

Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS
2. 61 desktop MB

Coreboot – internals

Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS
2. 61 desktop MB
3. 39 server MB

Coreboot – internals

Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS
2. 61 desktop MB
3. 39 server MB
4. 30 laptop MB (and some Apple)

Coreboot – internals

Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS
2. 61 desktop MB
3. 39 server MB
4. 30 laptop MB (and some Apple)
5. 33 embedded boards

Coreboot – internals

Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS
2. 61 desktop MB
3. 39 server MB
4. 30 laptop MB (and some Apple)
5. 33 embedded boards
6. 17 Mini-ITX / Micro-ITX / Nano-ITX

Coreboot – internals

Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS
2. 61 desktop MB
3. 39 server MB
4. 30 laptop MB (and some Apple)
5. 33 embedded boards
6. 17 Mini-ITX / Micro-ITX / Nano-ITX
7. 7 Set-top-boxes / Thin clients

Coreboot – internals

Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS
2. 61 desktop MB
3. 39 server MB
4. 30 laptop MB (and some Apple)
5. 33 embedded boards
6. 17 Mini-ITX / Micro-ITX / Nano-ITX
7. 7 Set-top-boxes / Thin clients
8. QEMU

Coreboot – internals

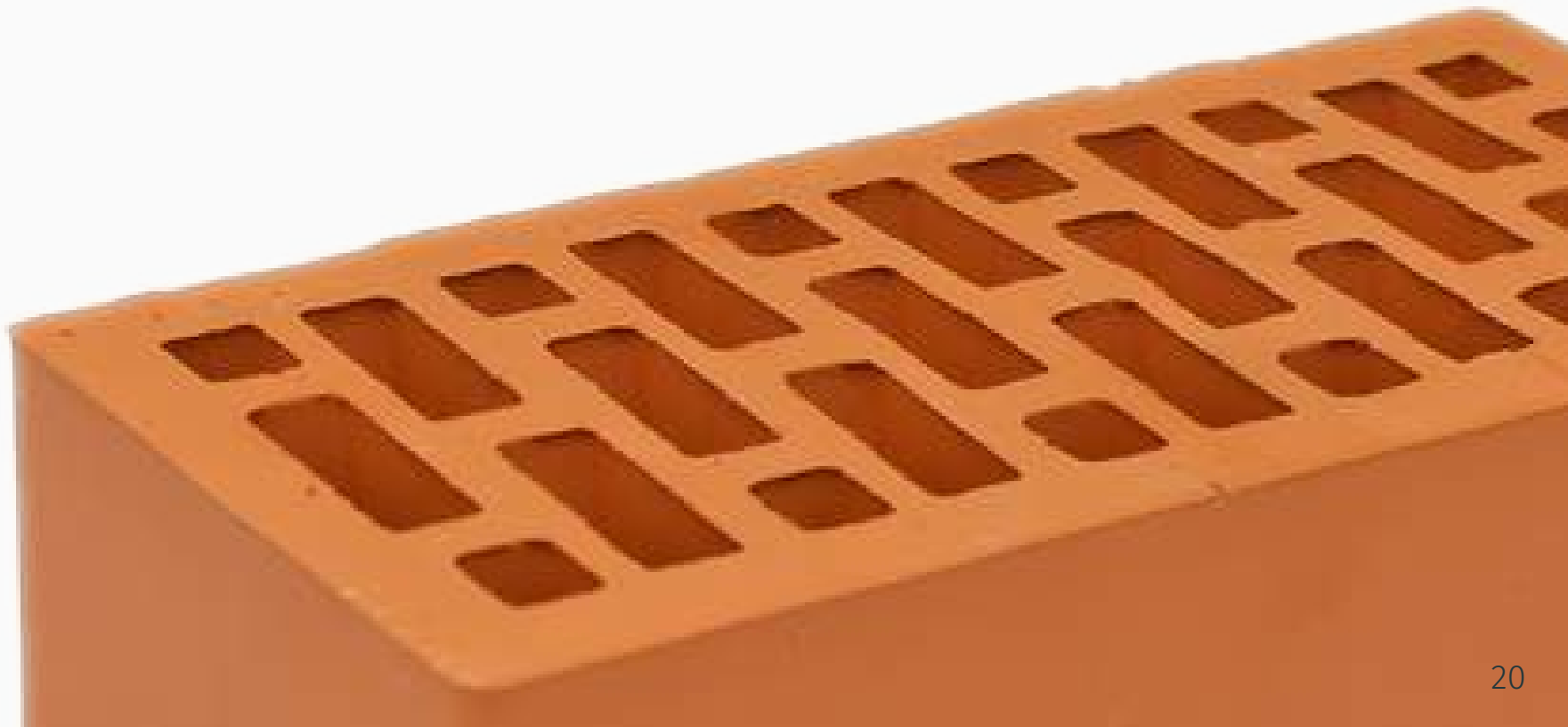
Заявленное поддерживаемое оборудование в CoreBoot v4 на May/2015:

1. x86, x86_64, ARM, ARM64, MIPS
2. 61 desktop MB
3. 39 server MB
4. 30 laptop MB (and some Apple)
5. 33 embedded boards
6. 17 Mini-ITX / Micro-ITX / Nano-ITX
7. 7 Set-top-boxes / Thin clients
8. QEMU
9. + more

Coreboot HOWTO

Будет не лишним напомнить:

Автор не несёт ответственности за возможные повреждения оборудования, упущенную выгоду, ~~разрушенную психику~~ и отсутствие удачи в столь нелёгком деле;



Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;

Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;
2. обновить оригинальный проприетарный BIOS до последней версии;

Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;
2. обновить оригинальный проприетарный BIOS до последней версии;
3. ROM datasheet и руководство по разборке устройства;

Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;
2. обновить оригинальный проприетарный BIOS до последней версии;
3. ROM datasheet и руководство по разборке устройства;
4. программный программатор - flashrom (иногда работает);

Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;
2. обновить оригинальный проприетарный BIOS до последней версии;
3. ROM datasheet и руководство по разборке устройства;
4. программный программатор - flashrom (иногда работает);
5. аппаратный программатор (когда не работает программный);

Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;
2. обновить оригинальный проприетарный BIOS до последней версии;
3. ROM datasheet и руководство по разборке устройства;
4. программный программатор - flashrom (иногда работает);
5. аппаратный программатор (когда не работает программный);
6. паяльная станция (когда аппаратный нельзя подключить через ISP);

Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;
2. обновить оригинальный проприетарный BIOS до последней версии;
3. ROM datasheet и руководство по разборке устройства;
4. программный программатор - flashrom (иногда работает);
5. аппаратный программатор (когда не работает программный);
6. паяльная станция (когда аппаратный нельзя подключить через ISP);
7. документации по offset для модулей в оригинальном проприетарном BIOS;

Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;
2. обновить оригинальный проприетарный BIOS до последней версии;
3. ROM datasheet и руководство по разборке устройства;
4. программный программатор - flashrom (иногда работает);
5. аппаратный программатор (когда не работает программный);
6. паяльная станция (когда аппаратный нельзя подключить через ISP);
7. документации по offset для модулей в оригинальном проприетарном BIOS;
8. git, GCC, binutils + некоторые other dependencies;

Coreboot – HowTO

Что потребуется для установки CoreBoot?

1. ~~удача~~ найти MB в списке поддерживаемого оборудования;
2. обновить оригинальный проприетарный BIOS до последней версии;
3. ROM datasheet и руководство по разборке устройства;
4. программный программатор - flashrom (иногда работает);
5. аппаратный программатор (когда не работает программный);
6. паяльная станция (когда аппаратный нельзя подключить через ISP);
7. документации по offset для модулей в оригинальном проприетарном BIOS;
8. git, GCC, binutils + некоторые other dependencies;
9. CoreBoot source;

Coreboot – HowTO

Общий алгоритм для установки CoreBoot на устройство:

1. Прочитать из ПЗУ оригинальный проприетарный System BIOS;

Coreboot – HowTO

Общий алгоритм для установки CoreBoot на устройство:

1. Прочитать из ПЗУ оригинальный проприетарный System BIOS;
2. Извлечь жизненно необходимые модули для которых нет свободной замены;

Coreboot – HowTO

Общий алгоритм для установки CoreBoot на устройство:

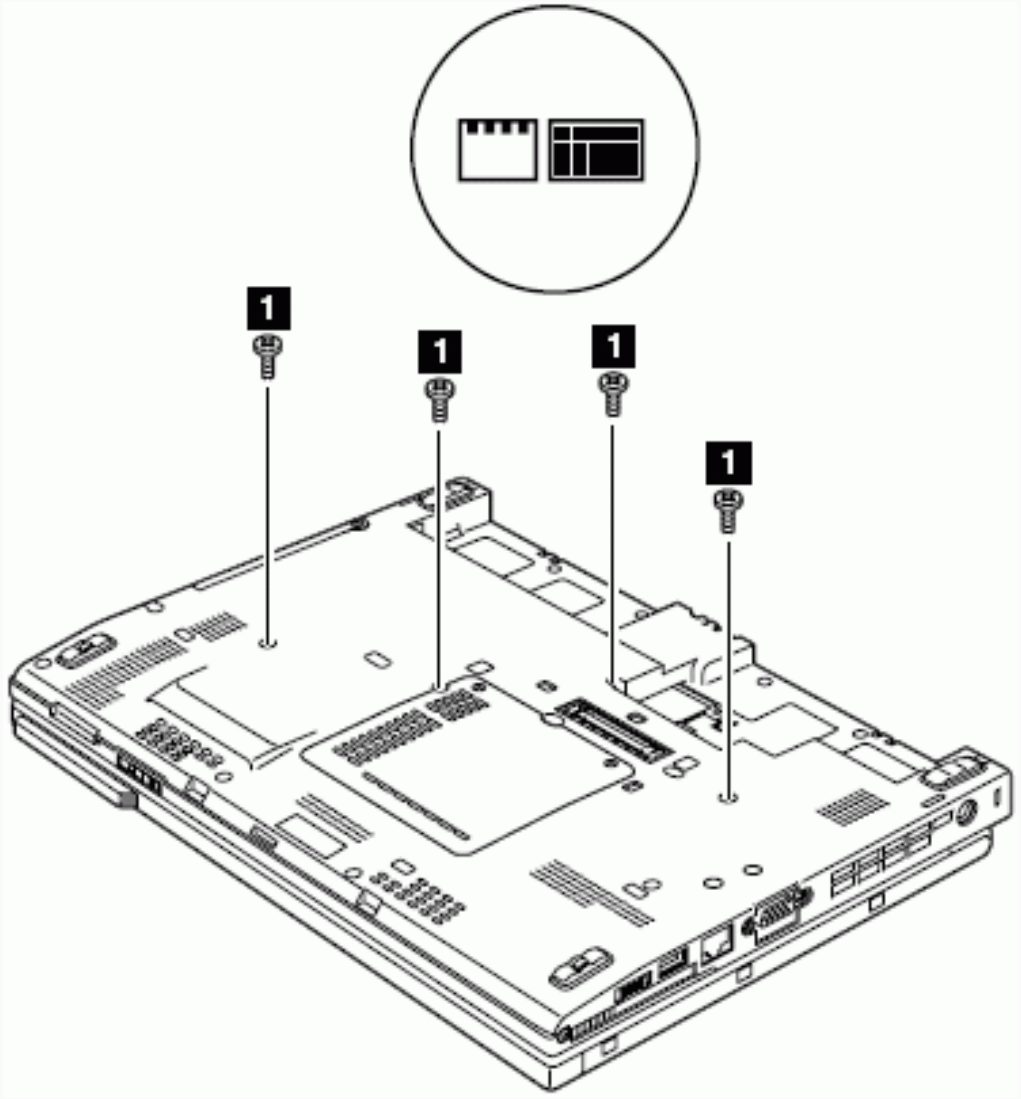
1. Прочитать из ПЗУ оригинальный проприетарный System BIOS;
2. Извлечь жизненно необходимые модули для которых нет свободной замены;
3. Собрать Coreboot с модулями;

Coreboot – HowTO

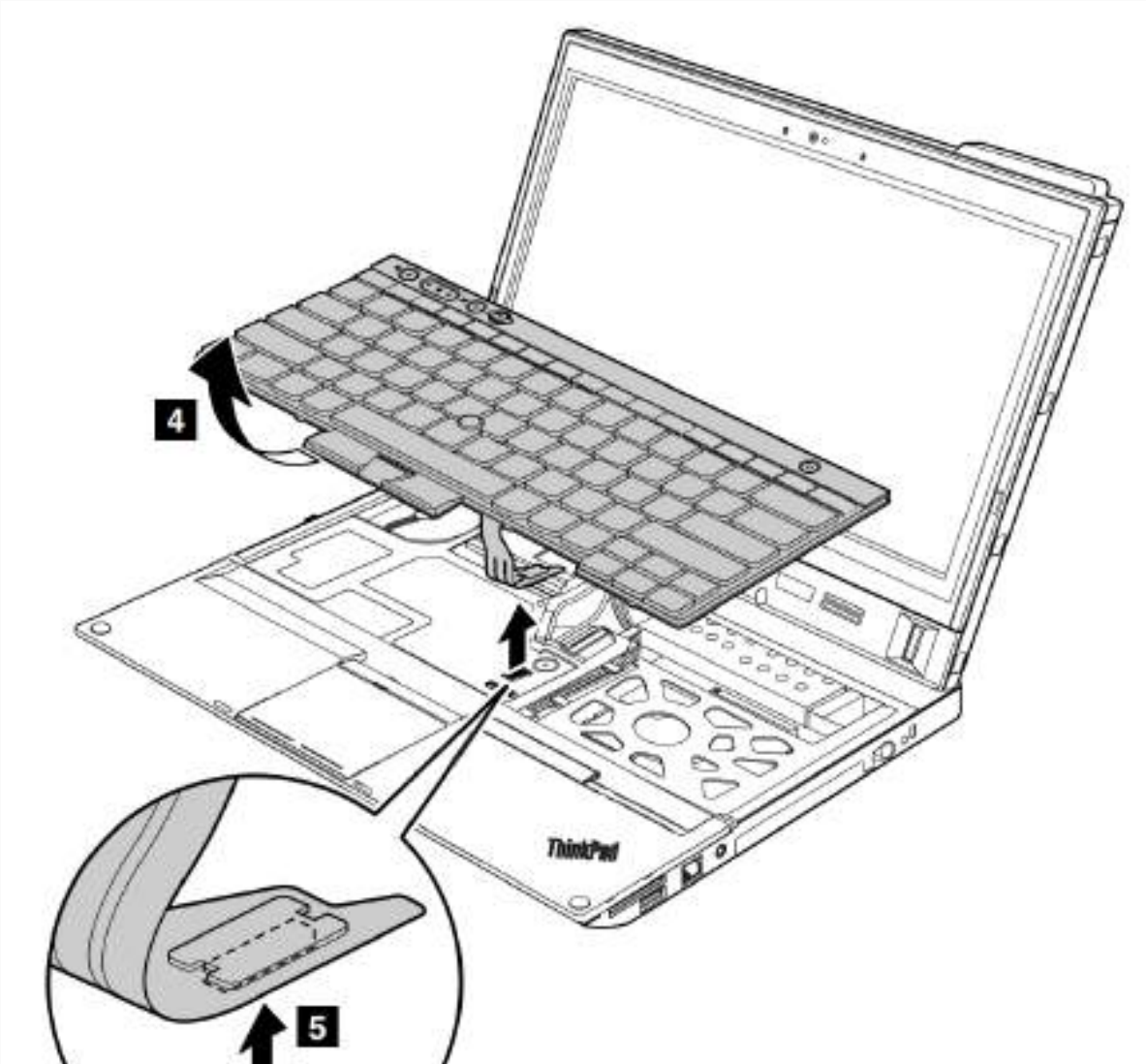
Общий алгоритм для установки CoreBoot на устройство:

1. Прочитать из ПЗУ оригинальный проприетарный System BIOS;
2. Извлечь жизненно необходимые модули для которых нет свободной замены;
3. Собрать Coreboot с модулями;
4. Записать build/coreboot.rom в ПЗУ;

Thinkpad x201 - Разборка и локализация ПЗУ

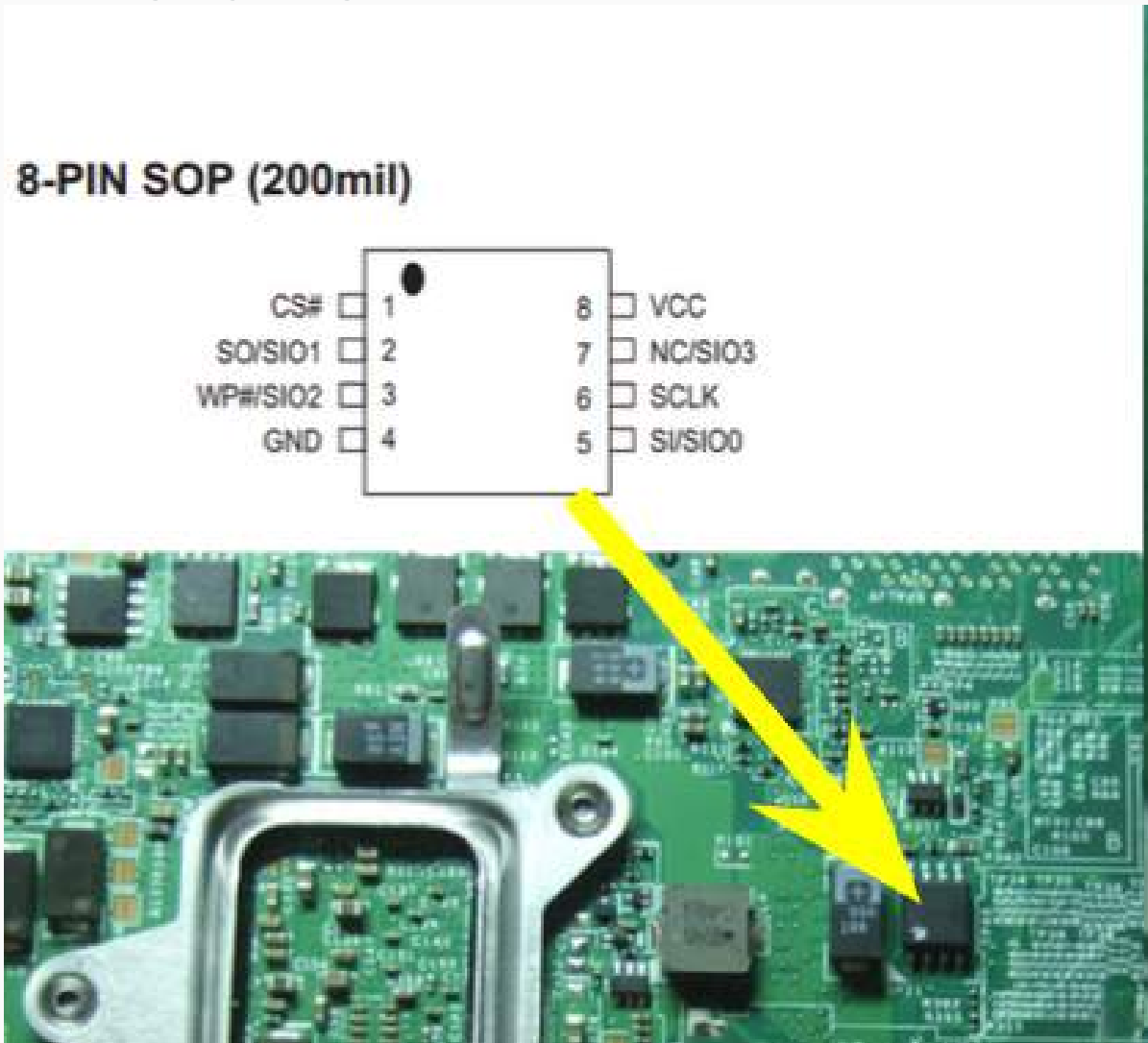


Thinkpad x201 - Разборка и локализация ПЗУ



Thinkpad x201 - Разборка и локализация ПЗУ

MX25L6445E



магия извлечения ME.bin и descriptor.bin для Thinkpad x201:

```
dd if=flash.bin of=descriptor.bin count=12288 bs=1M\
 iflag=count_bytes
dd if=flash.bin of=me.bin skip=12288 count=5230592\
 bs=1M iflag=count_bytes,skip_bytes
cat /proc/iomem | grep 'Video ROM' | (read m; \
m=${m/ :*}; s=${m/-*}; e=${m/*-}; dd if=/dev/mem\
of=vgabios.bin bs=1c skip=$[0x$s] \
count=$[$[0x$e]-$[0x$s]+1]
```

Собираем Coreboot для Thinkpad x201:
важные параметры сборки

make menuconfig

1. MX25L6445E

Собираем Coreboot для Thinkpad x201:
важные параметры сборки

make menuconfig

1. MX25L6445E
2. LENOVO x201

Собираем Coreboot для Thinkpad x201:
важные параметры сборки

make menuconfig

1. MX25L6445E
2. LENOVO x201
3. ME.bin

Собираем Coreboot для Thinkpad x201:
важные параметры сборки

make menuconfig

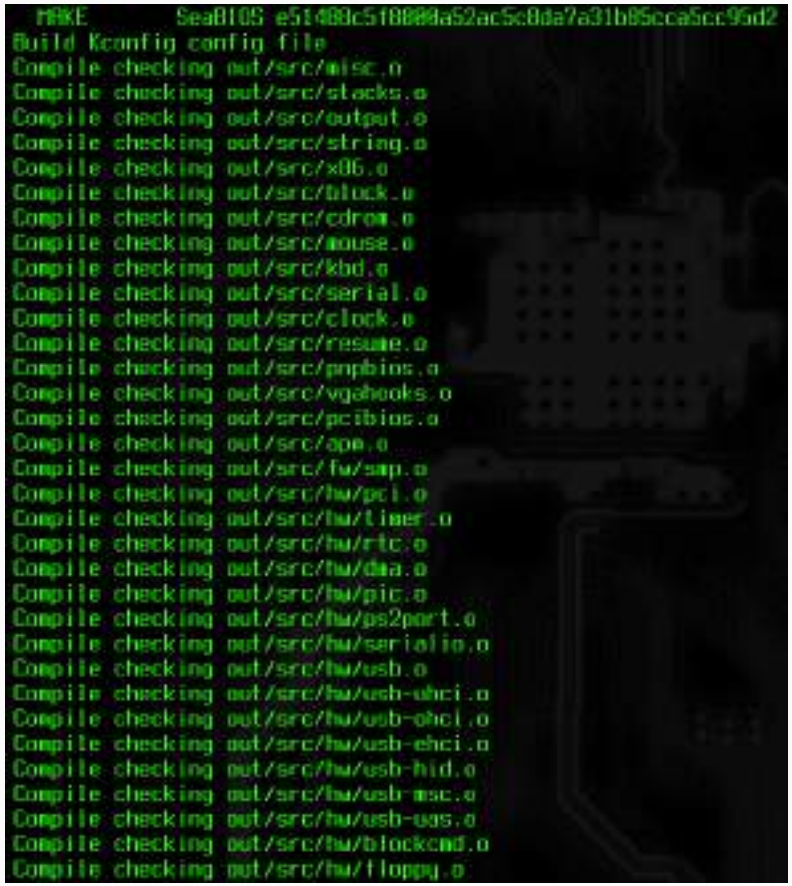
1. MX25L6445E
2. LENOVO x201
3. ME.bin
4. descriptor.bin

Собираем Coreboot для Thinkpad x201:
важные параметры сборки

make menuconfig

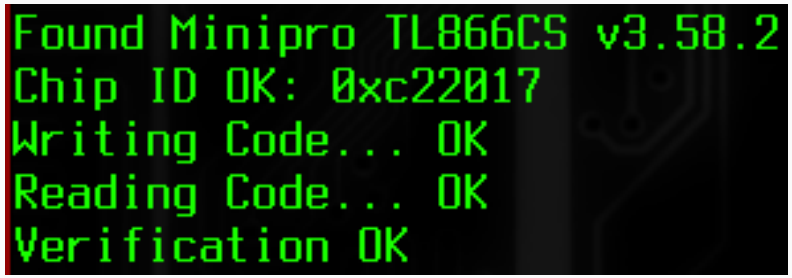
1. MX25L6445E
2. LENOVO x201
3. ME.bin
4. descriptor.bin
5. videorom.bin

make


MAKE SeaBIOS e5148Bc5f8888a52ac5c8da7a31b85cca5cc95d2
Build Kconfig config file
Compile checking out/src/misc.o
Compile checking out/src/stacks.o
Compile checking out/src/output.o
Compile checking out/src/string.o
Compile checking out/src/x86.o
Compile checking out/src/block.o
Compile checking out/src/cdrom.o
Compile checking out/src/mouse.o
Compile checking out/src/kbd.o
Compile checking out/src/serial.o
Compile checking out/src/clock.o
Compile checking out/src/resume.o
Compile checking out/src/pnpbios.o
Compile checking out/src/vgahooks.o
Compile checking out/src/pcibios.o
Compile checking out/src/ape.o
Compile checking out/src/fw/smp.o
Compile checking out/src/hw/pci.o
Compile checking out/src/hw/timer.o
Compile checking out/src/hw/rtc.o
Compile checking out/src/hw/dma.o
Compile checking out/src/hw/pic.o
Compile checking out/src/hw/ps2port.o
Compile checking out/src/hw/serialio.o
Compile checking out/src/hw/usb.o
Compile checking out/src/hw/usb-uhci.o
Compile checking out/src/hw/usb-ohci.o
Compile checking out/src/hw/usb-ehci.o
Compile checking out/src/hw/usb-hid.o
Compile checking out/src/hw/usb-msc.o
Compile checking out/src/hw/usb-uas.o
Compile checking out/src/hw/blockcmd.o
Compile checking out/src/hw/floppy.o

Подключение программатора к ПЗУ ноутбка:

Запись в ПЗУ:


Found Minipro TL866CS v3.58.2
Chip ID OK: 0xc22017
Writing Code... OK
Reading Code... OK
Verification OK

DEMO TIME

Coreboot – demo

Questions?