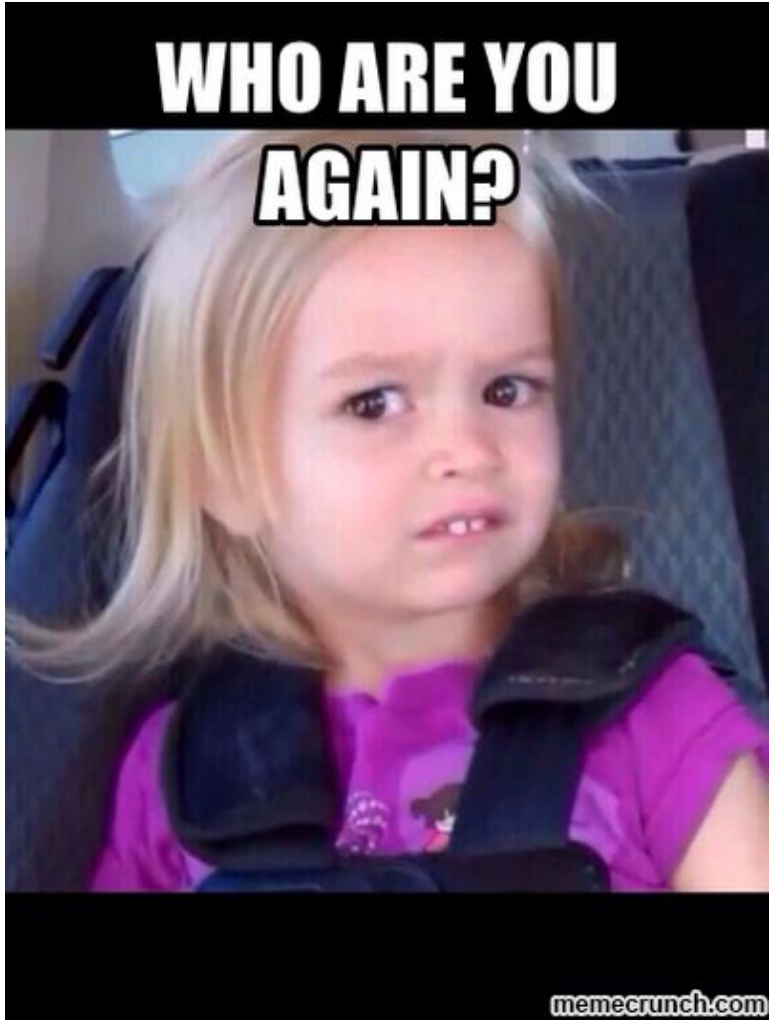
An Open Source Embedded Controller

Bill Richardson

2016 coreboot conference


WHO ARE YOU
AGAIN?
memecrunch.com

Chrome OS firmware engineer since 2009

- Verified Boot
- Developer Mode
- Embedded Controller
- Case-closed debugging
- Some other stuff I can’t talk about (yet)

What’s an Embedded Controller anyway?

- A tiny SoC that manages battery charging, fans, keyboard, LEDs, etc.
- Typically runs even when the main system processor is off
  - We call the main system CPU the “AP” (for Application Processor)
- Most laptops have them
- Most Chromebooks do too
- Ours is open source, which is unusual

What’s this have to do with coreboot?

- Not much, really
- It’s used in most Chromebooks, though
- Stefan Reinauer thought y’all might find it interesting

What’s this have to do with coreboot?

- Not much, really
- It’s used in most Chromebooks, though
- Stefan Reinauer thought y’all might find it interesting

So here I am

A bit of history

- The first three Chromebooks used a UEFI BIOS
- It worked, but had several drawbacks
  - Large
  - Slow
  - Complicated
  - Expensive
  - Closed source (TianoCore is only part of it)
  - Only builds on Windows
- After that, we switched to coreboot

and there was much rejoicing

But the EC was still provided by the ODM

- This had several drawbacks too
  - Slow
  - Buggy
  - Source unavailable (and probably not worth it)
  - Long turnaround time for every change
  - … which usually introduced new bugs

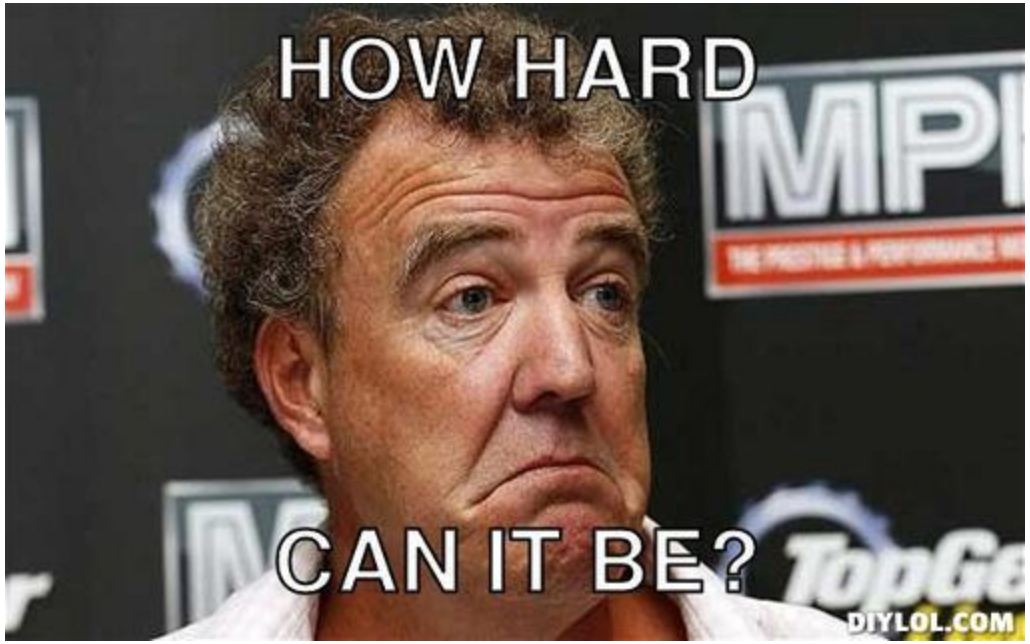
- Maybe we can make our own...

Primary responsibilities of the EC

- AP power sequencing
- Battery charging
- Thermal management
- Keyboard scan matrix
- Buttons and switches
- Backlights, indicator LEDs
- Various other board-specific peripherals

Power Sequencing

- Each AP family & motherboard has its own
  - Power states
  - Voltage regulators
  - Controlling GPIOs (both input and output)
  - Transition rules
  - Timing requirements
  - Trigger events
- The EC must manage and respond to all those requirements as the AP boots, sleeps, idles, or transitions between various subtle states.
- It must also ensure that certain peripherals are brought up and down too (USB, WiFi, etc.)


HOW HARD
CAN IT BE?
DIYLOL.COM

It’s actually not that bad

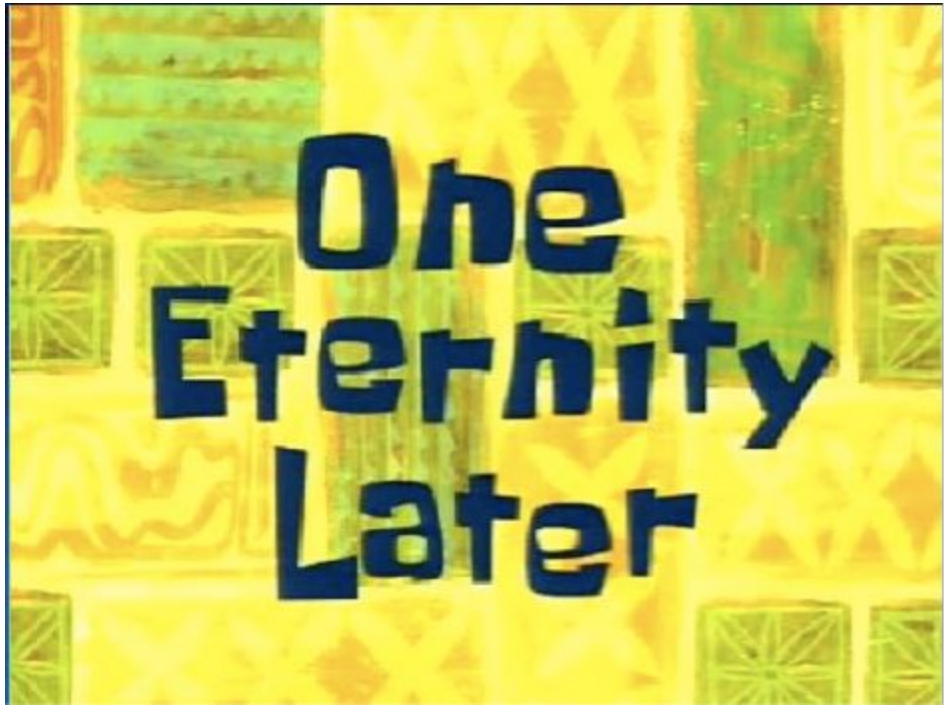
- All we need is a midrange SoC
- With various GPIOs and peripherals
  - Hm, there aren’t a ton of choices...
- Oh, and we’ll need an SDK or something to write the software

Texas Instruments had a nice SoC

- The Stellaris LM4 (now called TM4) should do fine
  - ARM M4F core
  - Integrated flash and RAM
  - Lots of GPIOs for keyboard scanning
  - ADCs for power & thermal monitoring
  - PWM controllers for fans and backlighting
  - Timers, counters, blah blah blah
- Their engineers were very helpful
- And we can license their SDK

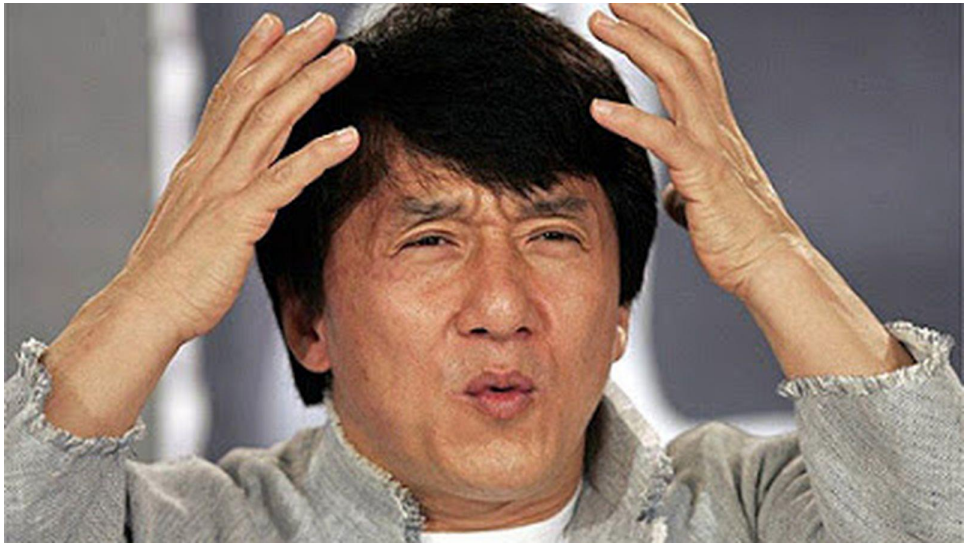
Texas Instruments had a nice SoC

- The Stellaris LM4 (now called TM4) should do fine
  - ARM M4F core
  - Integrated flash and RAM
  - Lots of GPIOs for keyboard scanning
  - ADCs for power & thermal monitoring
  - PWM controllers for fans and backlighting
  - Timers, counters, blah blah blah
- Their engineers were very helpful
- And we can **license** their SDK
  - **#include <lawyers.h>**


One
Eternity
Later

We now have permission to look at their SDK

- It’s … not ideal



- Let’s think of something else...

The Chrome OS EC Firmware

- Randall Spangler and Vincent Palatin wrote a basic OS in about three days
- It does all we need
  - A small number of independent tasks
  - Each task has its own stack
  - Task switching is interrupt-driven
  - Strictly ordered list of task priorities
  - Events, mutexes, timers, callbacks
  - No heap (no malloc/free), so no memory leaks
  - Modular, configurable
  - Written in C
  - Open source
- It doesn’t have a clever name. It’s just “the EC firmware”. Sorry.

I already gave a talk on it a couple years ago

- www.chromium.org/chromium-os/2014-firmware-summit

We’ve made some improvements since then

- Code cleanup & refactoring
- More chip vendors
  - It83xx, lm4, mec1322, npcx, nrf51, stm32
- More CPU cores
  - cortex-m, cortex-m0, nds32
- Many more Chromebooks
- More use cases
  - Original EC functions
  - USB-PD controllers
  - Case-closed debug controller
  - USB Type-C power brick
  - More sensors and peripherals


SECURE
ALL THE THINGS
memegenerator.net

EC Software Sync

It is important that the AP firmware (BIOS) and the EC firmware remain compatible through upgrades. At every* cold boot/reset of the EC

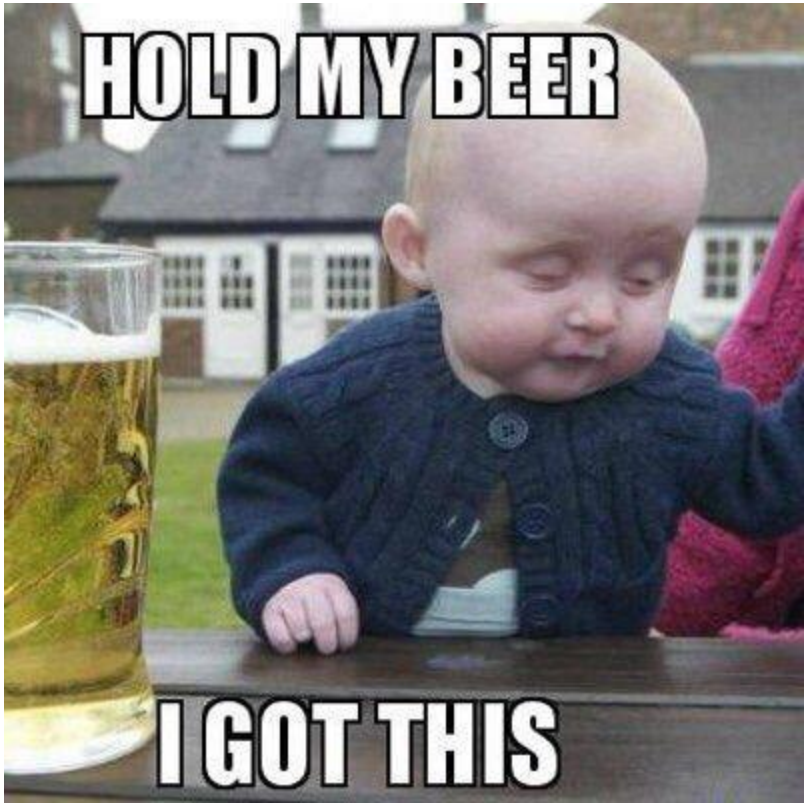
1. The EC boots its RO firmware, and powers on the AP.
2. The AP boots its RO firmware.
3. The AP verifies its RW firmware and jumps to it.
4. The EC computes a hash of its RW firmware.
5. The AP RW firmware contains a copy of the EC's RW firmware. The AP compares its hash with the EC's hash.
6. If they differ, the AP gives the EC the correct RW firmware, which the EC writes to its flash.
7. The EC jumps to its RW firmware.

There also are a few other tricks to ensure the EC isn't lying about its hash

*Normal mode, anyway. In recovery mode both AP and EC stay in their RO firmware

Standalone configuration

- Verified Boot requires some time-consuming cryptographic calculations
- Software Sync is an optimization to avoid doing this on the EC
- Removable devices like USB Type-C power bricks have no choice
- They use the same sort of vboot implementation as the AP
  - Can be configured for RO / RW, or RO / RW_A / RW_B
  - No TPM, so preventing rollback is a little different
  - No dev-mode in USB power bricks, duh
- It can take a second or two after power on before they’re ready for use


HOLD MY BEER
I GOT THIS

If you want to play around with it

- We support two boards from STMicroelectronics (www.st.com)
  - 32F072BDISCOVERY
  - STM32L476G-EVAL
  - Others will probably work without much trouble

- You'll need
  - GNU make version 4.1 (or a minor edit to the Makefiles)
  - openocd version 0.9.0 or newer (building from source is easy)
  - ARM cross-compiler (gcc-arm-none-eabi)

If you want to fiddle with the EC in your Chromebook

- Be very careful
- Use the Chrome OS build environment
  - www.chromium.org/chromium-os/developer-guide
- Use the correct release branch
  - chrome://system -> ec_info -> Expand
  - Or take apart the recovery image for your device
  - www.chromium.org/chromium-os/how-tos-and-troubleshooting/working-on-a-branch
- flashrom can be used to program the EC from the AP
  - flashrom -p ec --fast-verify -w ec.bin


I only changed one line
and it was a comment

Really, be very careful

- This is a warranty-voiding process
- Disable EC software sync
- Only update the RW half
  - flashrom has options to do this
- Accessing the serial port may require soldering
- Other than the Pixel lightbar, there's very little that's interesting
  - and you can drive that from the AP
- If you mess up the power sequencing, the AP may no longer boot
  - Ever

Ongoing work

- More use cases
- Reduce size, simplify code
- Get partners more involved
  - This is actually going very well
- Support more SoCs, cores, etc.

Maybe someday

- Major refactoring
- Improved security
- ???


THANK YOU!
ANY QUESTIONS?
memegenerator.net

Example build

```
sudo apt-get install gcc-arm-none-eabi

git clone https://chromium.googlesource.com/chromiumos/platform/ec
cd ec
make BOARD=discovery-stm32f072
```

Makefile.rules:243: *** multiple target patterns. Stop.

You need to edit Makefile.rules as follows:

```
@@ -240,12 +240,12 @@ $(out)/$(PROJECT).hex: $(out)/$(PROJECT).bin
     $(call quiet,bin_to_hex,OBJCOPY)

 $(out)/RW/%.elf: override BLD:=RW
-$(out)/RW/%.elf: private objs := $(rw-objs)
+$(out)/RW/%.elf: objs := $(rw-objs)
 $(out)/RW/%.elf: $(out)/RW/%.lds $(rw-objs) $(libsharedobjs_elf-y)
     $(call quiet,elf,LD     )

 $(out)/RO/%.elf: override BLD:=RO
-$(out)/RO/%.elf: private objs := $(ro-objs)
+$(out)/RO/%.elf: objs := $(ro-objs)
 $(out)/RO/%.elf: $(out)/RO/%.lds $(ro-objs) $(libsharedobjs_elf-y)
     $(call quiet,elf,LD     )
```

Build and install openocd

```
sudo apt-get install libtool autoconf libusb-1.0-0-dev
git clone git://git.code.sf.net/p/openocd/code openocd-code
cd openocd-code/
./bootstrap
./configure --enable-stlink
make
sudo make install
```

```
sudo cp /usr/local/share/openocd/contrib/99-openocd.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
```

Program the board

```
cd ec
make BOARD=discovery-stm32f072
```

The EC firmware will export a console over USB

```
> version
Chip:    stm stm32f07x
Board:   0
RO:      discovery-stm32f072_v1.1.4751-1
RW:      discovery-stm32f072_v1.1.4751-1
Build: discovery-stm32f072_v1.1.4751-1ab69e7 2016-06-12 17:50:26>
> help
Known commands:
  chan           gpioget        md             sysinfo        usart_info
  crash          gpioset        panicinfo      sysjump        version
  flashinfo      help           reboot         syslock        waitms
  flashwp        hibernate      rw             taskinfo
  gettime        history        shmem          timerinfo
HELP LIST = more info; HELP CMD = help on CMD.
>
```


THANK YOU FOR LISTENING
ANY QUESTIONS?
imgflip.com


ONE DOES NOT SIMPLY
SAY THANK YOU WITHOUT A MEME
makeameme.org