Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Using Zephyr for Embedded Controllers

Keith Short (keithshort@google.com)
Discord: Keith Short#2976
GitHub: keith-zephyr

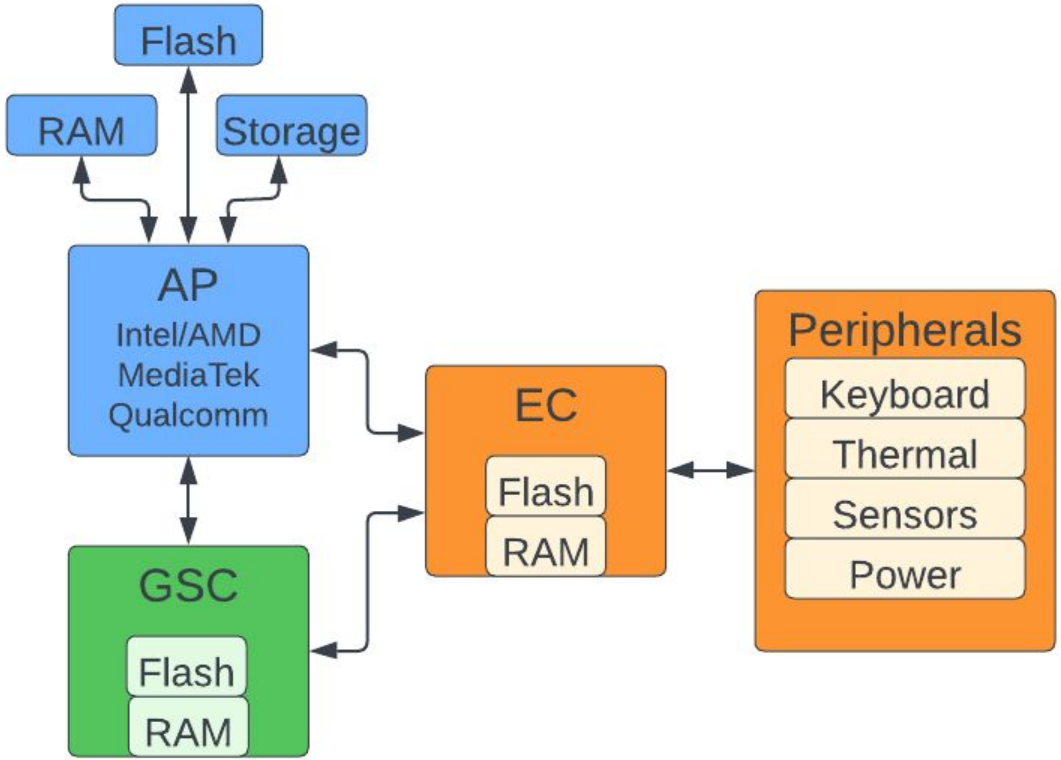
Agenda

- What is an Embedded Controller (EC)?
- Why switch to Zephyr?
- Transition to Zephyr
- How Google qualified its Zephyr based EC firmware

What is an Embedded Controller (EC)?

Chromebook block diagram (simplified)

- Chromebook CPUs
  - AP: application processor
    - >= 4 GiB RAM
    - >= 32 GiB SSD
    - >= 8 MiB flash
  - GSC: Google Security Chip
  - EC: embedded controller
    - >= 512 KiB flash
    - >= 64 KiB RAM
    - 48Mhz core
    - 60 - 80 GPIOs



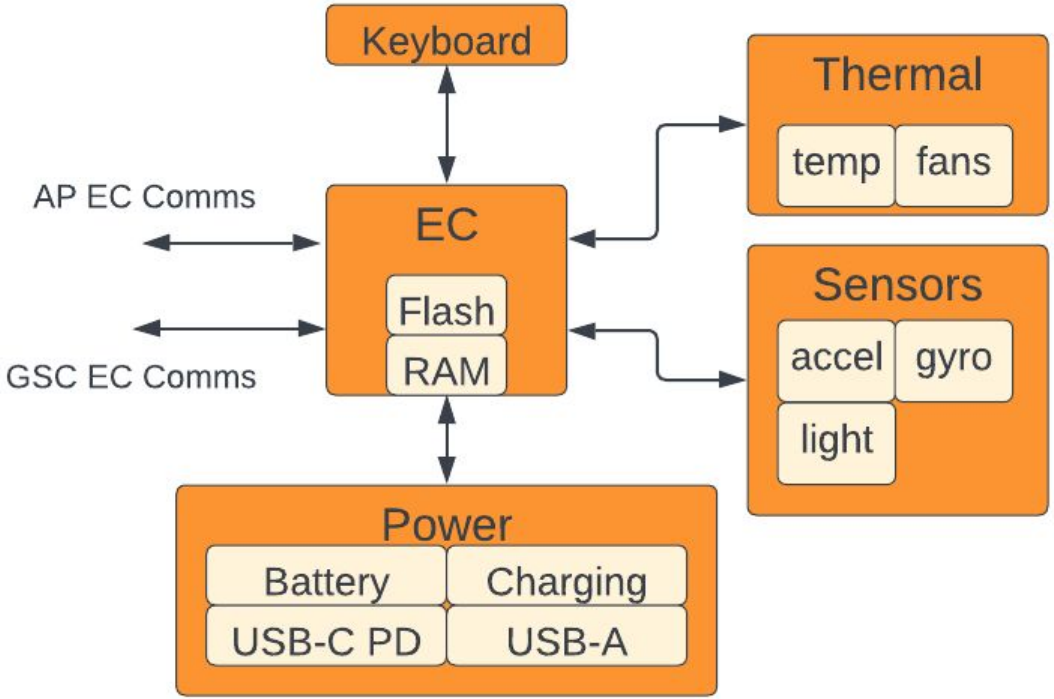
Zephyr™

Embedded Controller Overview

Always on microcontroller found on notebook and desktop PCs

EC role on Chromebooks

- Power on the AP
  - Wake the system from sleep
- Keyboard
- Sensors
- Thermal management
- Power management
  - Battery charging
  - USB-C Power Delivery
  - Power USB-A ports
- Shell interface



Zephyr™

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Why switch to Zephyr?

Google’s original ChromeOS EC

- Google’s original EC code
  - Open source RTOS
  - Created ~2012
  - > 200 Chromebook variants supported
- Non-technical limitations
  - Little community contributions
  - Vendors develop twice

Long term goal is an industry-standard EC supporting ChromeOS and Windows.

What Zephyr brings to ECs

- Community benefits
  - Open source
  - Neutral governance
  - More EC chip and peripheral support

- Technical benefits
  - Structured board configuration
    - Kconfig and devicetree
  - Memory protection

Zephyr™

What Google brings to Zephyr

- Hierarchical State Machine Framework
- USB-C Power-Delivery stack
  - Power sourcing and sinking
  - Alternate mode support
- New sensor framework
  - Android's CHRE available as Zephyr module
- Unified AP power sequencing

Zephyr™

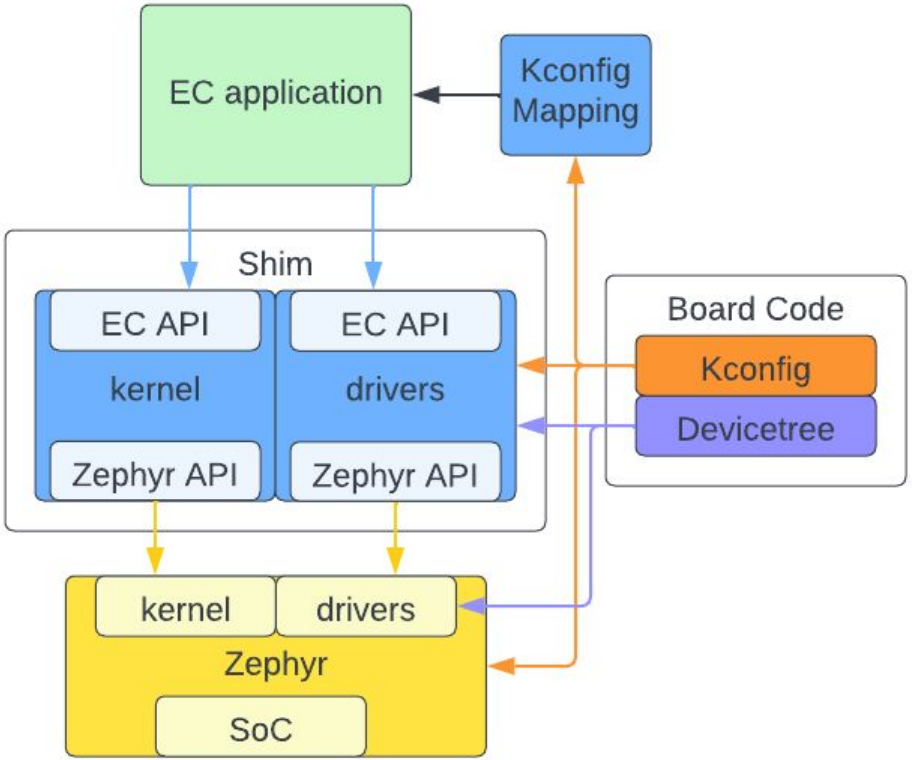
Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Transition to Zephyr

Migrating existing code to Zephyr

- Shim layer
  - Translate existing APIs to Zephyr APIs
  - Configured via Kconfig and devicetree
  - Reduces custom board code
- Kconfig mapping
  - Maps Kconfig options to existing `#defines`

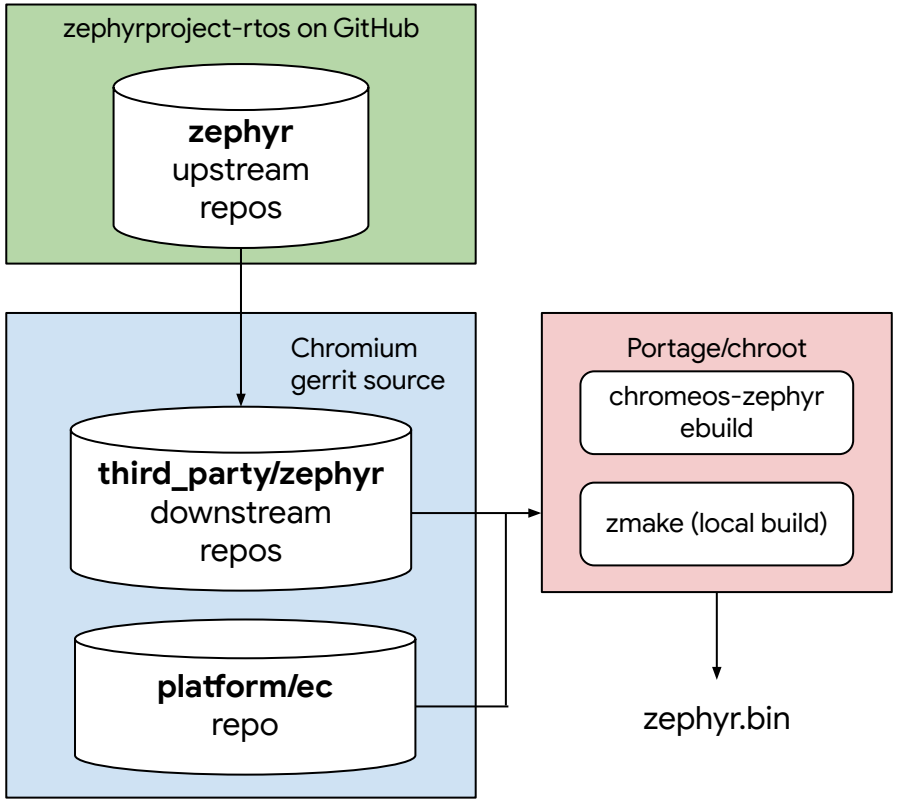


Zephyr™

Zephyr EC structure

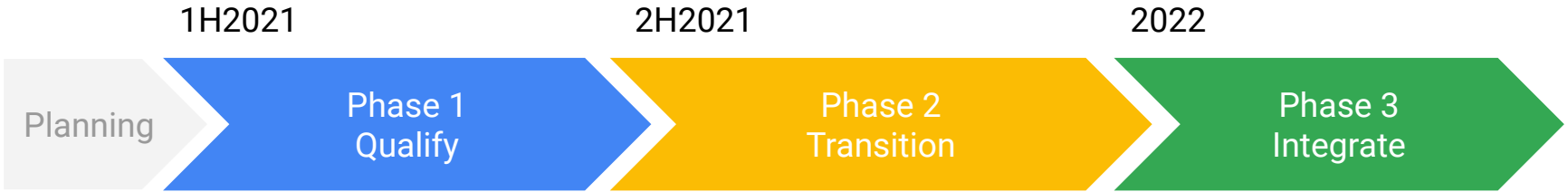
Multiple Chromium repositories

- third_party/zephyr
  - mirror of Zephyr upstream on GitHub
- platform/ec
  - Shared code with legacy EC
  - Zephyr shim and board projects under platform/ec/zephyr directory



Zephyr™

Timeline



Proof of Concept

- SoCs: Intel Tiger Lake, Qualcomm 7c, and MediaTek Kompanio 820
- ECs: ITE and Nuvoton

Commit to use of Zephyr to power EC on new devices

All new reference boards use Zephyr EC:

- AMD
- Intel
- Mediatek
- Qualcomm

Start moving code upstream

Zephyr™

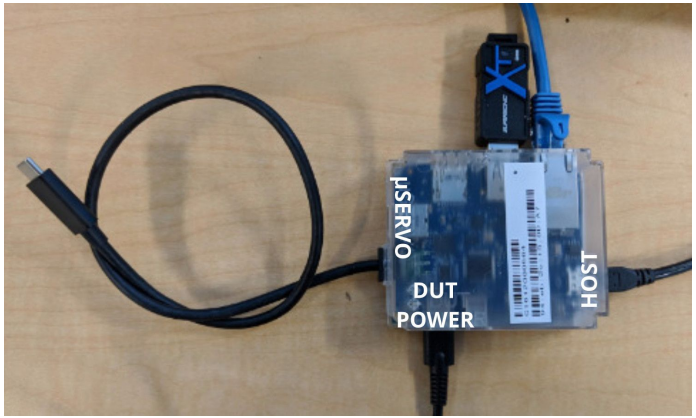
Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Qualification

Hardware Based Qualification

- External APIs kept the same
  - AP to EC communication
  - shell commands
- Closed case debug (CCD)
  - Access to GSC, AP, and EC consoles
- Automated firmware tests
  - CCD connection and SSH connection
  - Also verifies USB-PD
- Power consumption
  - No increase in low-power consumption
- Image size comparison
  - Flash size increase: < 10%
  - RAM size increased: ~ 25%



Zephyr™

Hardware-free testing

- Tests use public APIs for verification
- There are 2 types of tests
  - Driver tests
  - Integration tests
- Emulators
  - I2C controller
  - I2C targets
  - USB-C Power delivery partner
- Emulators are in platform/ec/zephyr/emul and tests are in platform/ec/zephyr/test
- Coverage reports are generated in our GitLab CI, and are publically available

Demo


1
ec:~$ *** Booting Zephyr OS build v2.7.99-7890-g5a7e91b1da52 ***
--- UART initialized after reboot ---
[Image: RO, herobrine_v3.0.89800-ec:445a4a,os:5a7e91,cmsis:45216b,hal_stm32:24c512,nanopb:8eb
[Reset cause: reset-pin]
[0.015800 KB boot key mask 0]
[0.016500 init buttons]
[0.017200 VB Main]
[0.017800 VB Ping Cr50]
[0.019900 hash start 0x00040000 0x00037e90]
ec:~$

Zephyr™

Questions?

References

- Google Security Chip: https://showcase.withgoogle.com/titan-c/
- Google's original EC code introduction: https://chromium.googlesource.com/chromiumos/platform/ec/+/HEAD/README.md
- Zephyr EC Application
  - Introduction Document: https://chromium.googlesource.com/chromiumos/platform/ec/+/HEAD/docs/zephyr/README.md
  - Source code: https://source.chromium.org/chromiumos/chromiumos/codesearch/+/main:src/platform/ec/
- Upstream code
  - State Machine Framework: https://docs.zephyrproject.org/latest/services/smf/index.html
  - USB-C Drivers: https://github.com/zephyrproject-rtos/zephyr/tree/main/drivers/usbc
  - CHRE module: https://github.com/zephyrproject-rtos/chre
  - I2C Controller Emulator: https://github.com/zephyrproject-rtos/zephyr/blob/main/drivers/i2c/i2c_emul.c

References (continued)

- Test Coverage: https://gitlab.com/zephyr-ec/ec/-/jobs
- Chromium repositories
  - third_party/zephyr: https://source.chromium.org/chromiumos/chromiumos/codesearch/+/main:src/third_party/zephyr/
  - platform/ec: https://source.chromium.org/chromiumos/chromiumos/codesearch/+/main:src/platform/ec/
- Closed case debug (CCD)
  - https://chromium.googlesource.com/chromiumos/third_party/hdctools/+/HEAD/docs/ccd.md
-

Zephyr EC threads (highest to lowest priority)

- USB-C Power delivery interrupt (1 per type-C port)
- USB-C Power delivery thread (1 per type-C port)
- Keyboard scanning
- AP command processing
- Sensors
- AP Power Sequencing
- Shell
- Charger
- BC1.2 host/client (1 per type-C port)
- System Work Queue
- Idle