Google

Chrome Firmware 101

(AP/EC/PD Firmware)

POCs:

subratabanik@, digehlot@, jayvik@, pranavayn@

OOSC'24
August 2024

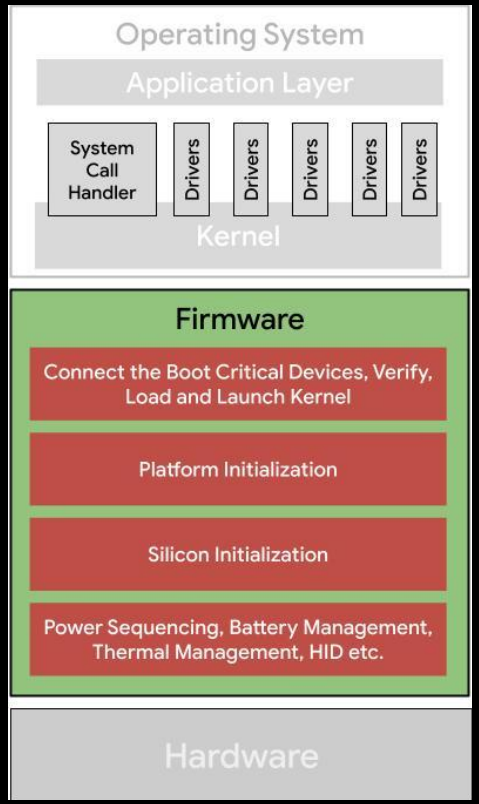
Objectives

Provide participants with a foundational understanding of Chrome firmware, specifically focusing on the roles and interactions of AP, EC, and PD firmware components. Upon completion, attendees will be able to:

- Describe the primary functions and responsibilities of AP, EC, and PD firmware within the Chrome OS ecosystem.
- Explain the key communication and data exchange mechanisms between these firmware components.
- Gain the confidence to delve deeper into Chrome firmware exploration and development.

Google

What is Firmware?



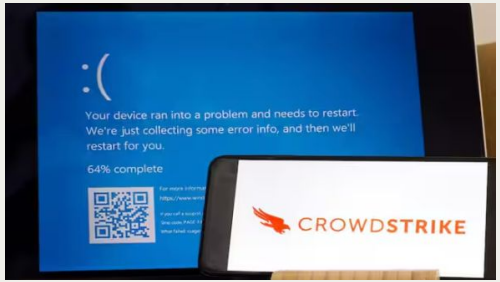
As per IEEE 610.12-1990, the definition of Firmware is:

The *tiny* block that combines of hardware device, computer instructions and data, reside as *read-only* software on the device.

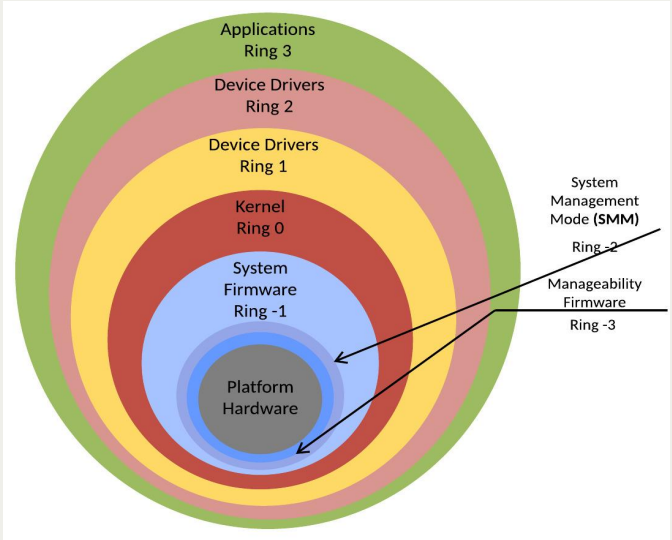
"***Tiny***" and "***read-only***" terms are misleading while defining the scope of the modern firmware.

An ***essential*** piece of code that is **responsible for performing** the underlying **hardware initialization** prior to **handing over to the operating system**.

Why Investing in Firmware ?

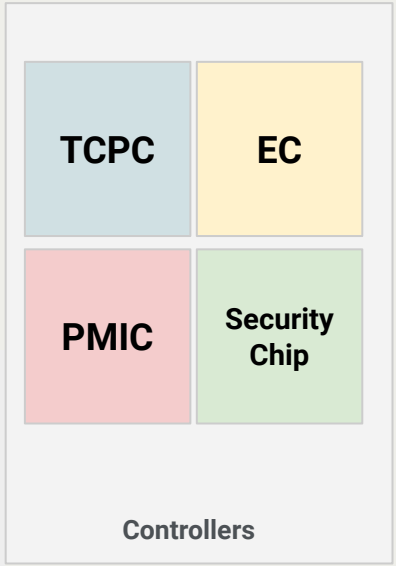


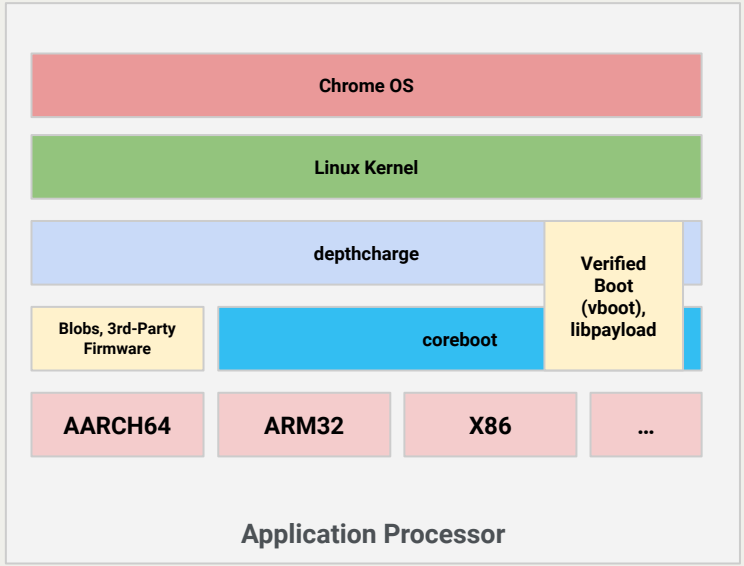
- **Consistent behaviour across architectures** (i.e. ARM, PowerPC, X86).
- **Flexible** Firmware to OS interfaces.
- Firmware been close to HW its provides **better control** of the Platform.
- Time is money - fixing bugs in firmware is **comparatively easy.**

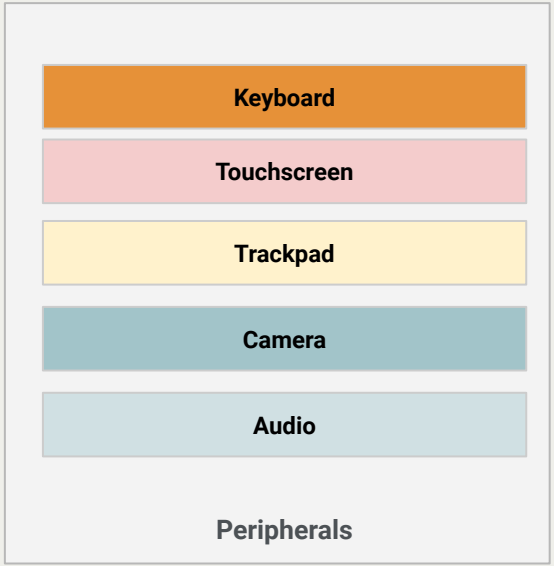


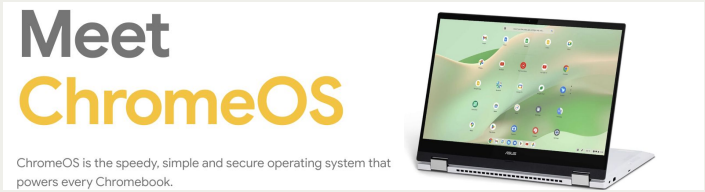
Firmware = Security

Chromebook Platform Overview









Google

Chromebook Firmware Architecture

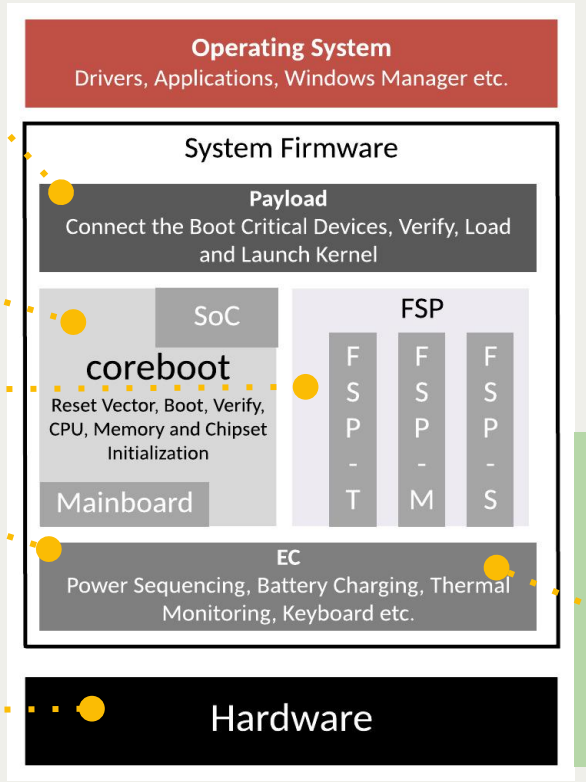
depthcharge, specific payload designed to meet chromeOS boot requirements. **(O)**

coreboot, open source boot firmware is responsible to perform platform initialization (along with silicon init). Three copies of the firmware in flash
- 1 in RO flash for recovery mode
- 2 in RW flash for verified A / B boot **(O)**

Silicon Reference Code (owned by SoC vendors)

The **PD** firmware is running on Type-C Port Controller (TCPC) chip, is responsible for end-point device negotiation/detection over USB-C ports. **(O)**

chromeOS running on the different types of SoC architecture (Intel/AMD/MTK/QC)



Google Security Chip (GSC) running specific firmware is responsible for defining platform Root-of-Trust (RoT). Additionally, provides all TPM related services as per TCG specifications.

The **EC** is a low-power microcontroller (running Zephyr) that keeps your Chromebook working when it's off or sleeping. **(O)**

(O) : Open Source Firmware Development

Introduction to Chrome Firmware

Jayvik Desai
Overview of Embedded Controller (EC) Firmware

Dinesh Gehlot
Overview of Application Processor (AP) Firmware / BIOS

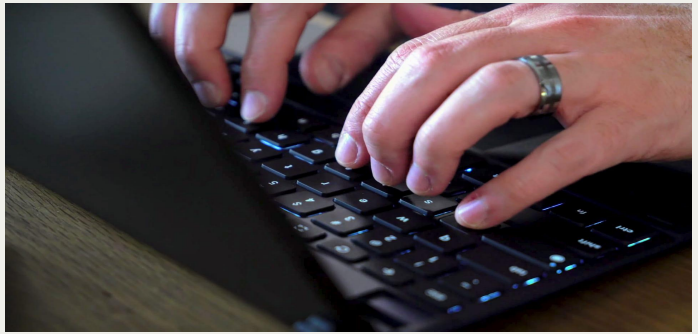
Pranava Y N
Overview of Power Delivery (PD) Firmware for TCPC

Introduction to Embedded Controller (EC) Firmware

Embedded Controller Introduction

What happens when you press the power button ?

How is the keyboard input detected ?



Embedded Controller Introduction

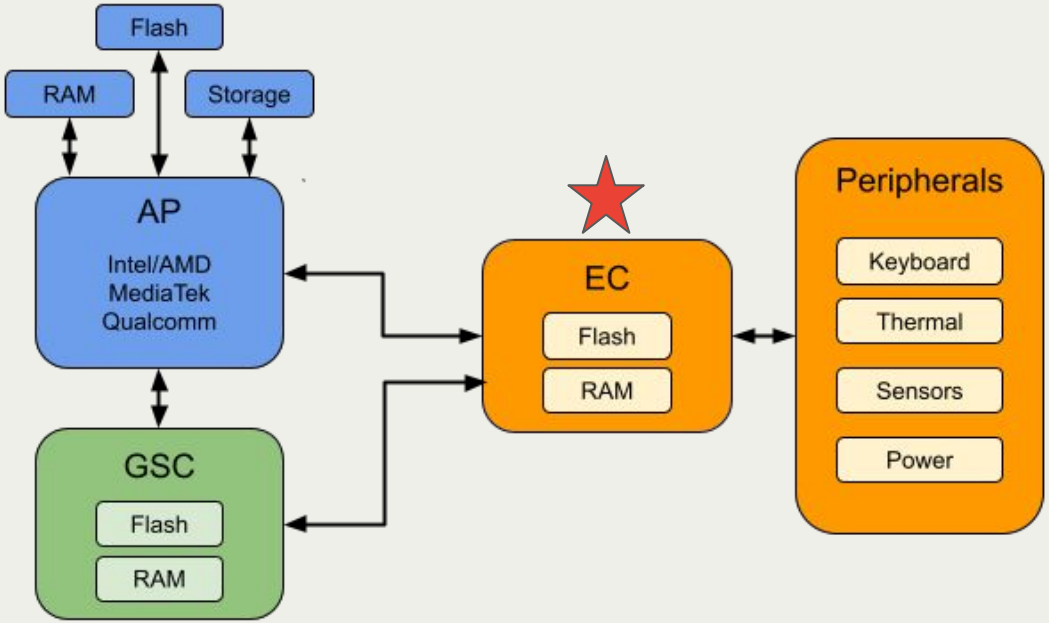
What manages your USB connections ?

How does the laptop battery get charged ?

Embedded Controller Hardware

Typical hardware configuration

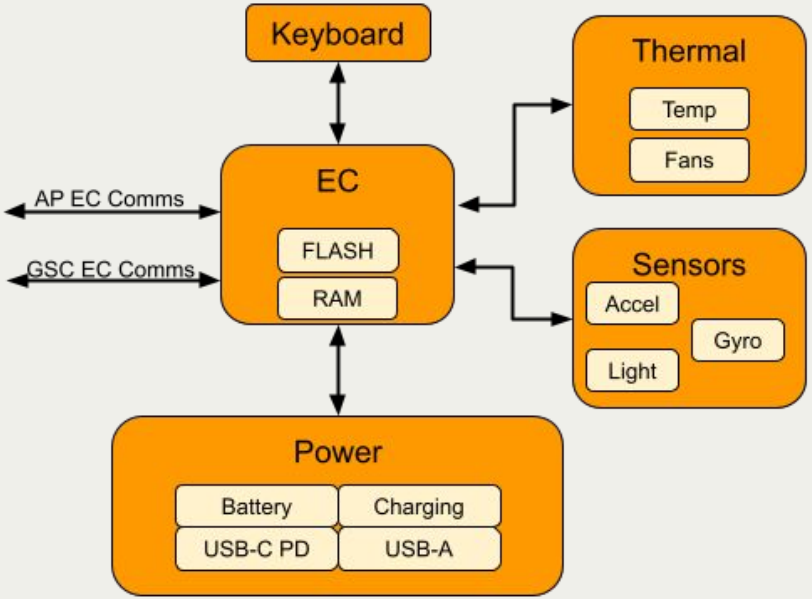
- ARM cortex-M4 processor
- >= 512 KB flash
- >= 64 KB RAM
- 48Mhz core
- 60 - 80 GPIOs
- Integrated flash and RAM
- XIP support
- Peripheral hardware



Embedded Controller Overview

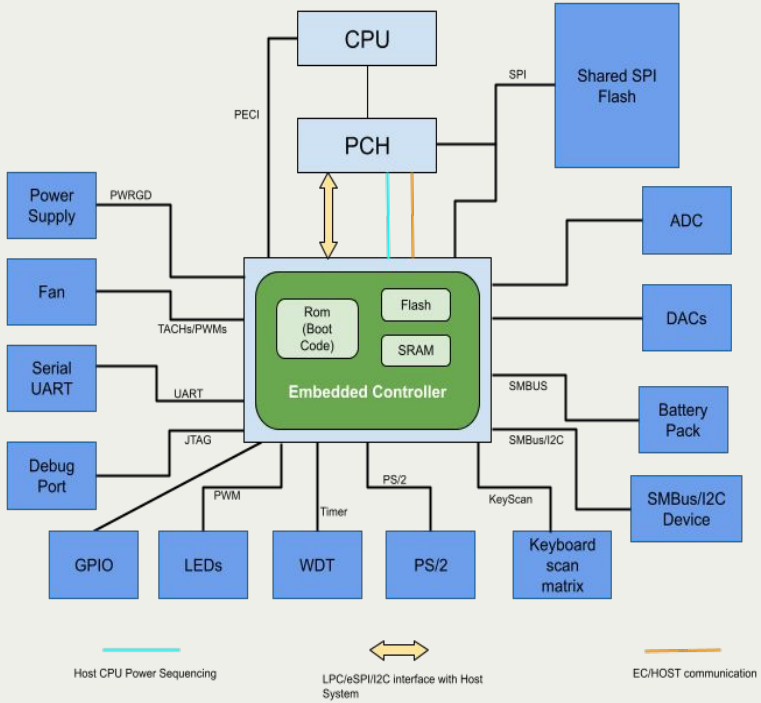
Responsibility of EC Firmware are:

- **AP Power Management**
  - Controls the **startup sequence** and timing of the application processor.
  - Monitors power status signals from the application processor.
- **Peripheral Management**
  - Handle Button Array Devices like Power Button, Vol Up/Down etc.
  - **Sensors** input like accelerometer, gyroscope etc.
  - Track the state change of switches (LID, Dock, Screen Rotate Lock etc.)



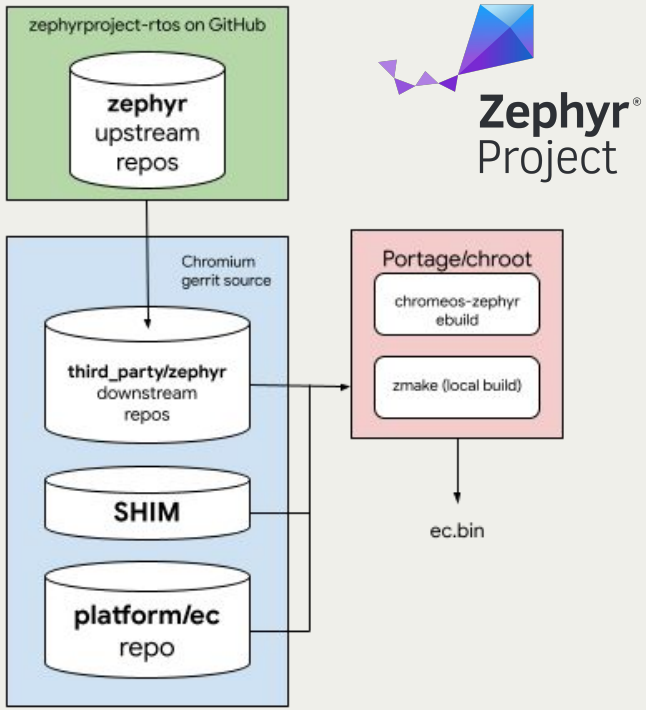
Embedded Controller Overview

- **Keyboard Control**
  - Scans **keyboard matrix** input.
  - Sends keypress data to the operating system.
- **Thermal Management**
  - Monitors device **temperature** (processor, RAM, PMIC's).
  - **Throttles fan** using PWM for cooling.
- **Power Management**
  - Oversees **battery and charging** status.
  - Controls power distribution to USB-A ports.
  - Manages **USB-C connections** (power negotiation, DisplayPort, USB4 modes).

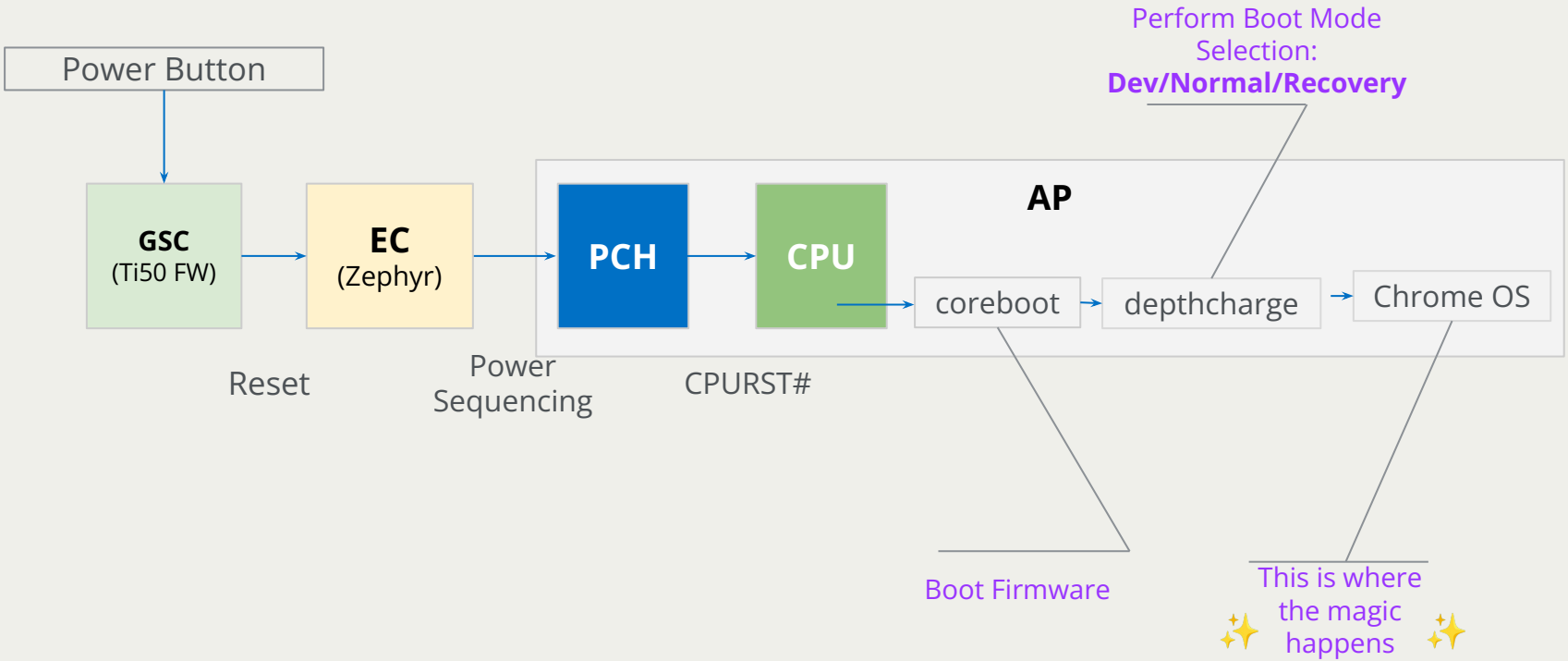


Zephyr RTOS in Embedded controller

- EC firmware for Chrome OS device is Open Source [chromiumos/platform/ec]
- Google added a **SHIM** layer to convert legacy Chrome EC application feature APIs to map into the Zephyr APIs.
- The legacy Chromium EC meets our technical requirements, but has non-technical limitations:
  - In-house RTOS, limited features.
  - Limited number of community contributors
  - Google has to implement all features
  - Vendors (EC chips, sensors, USB-C, etc.) have to write drivers specific to Chromebooks.
  - Google often writes these drivers.
  - Security vulnerabilities.
  - Useful frameworks like device tree, test framework, Hardware free testing etc.



Boot Flow - ChromeOS Device



Introduction to Application Processor (AP) Firmware

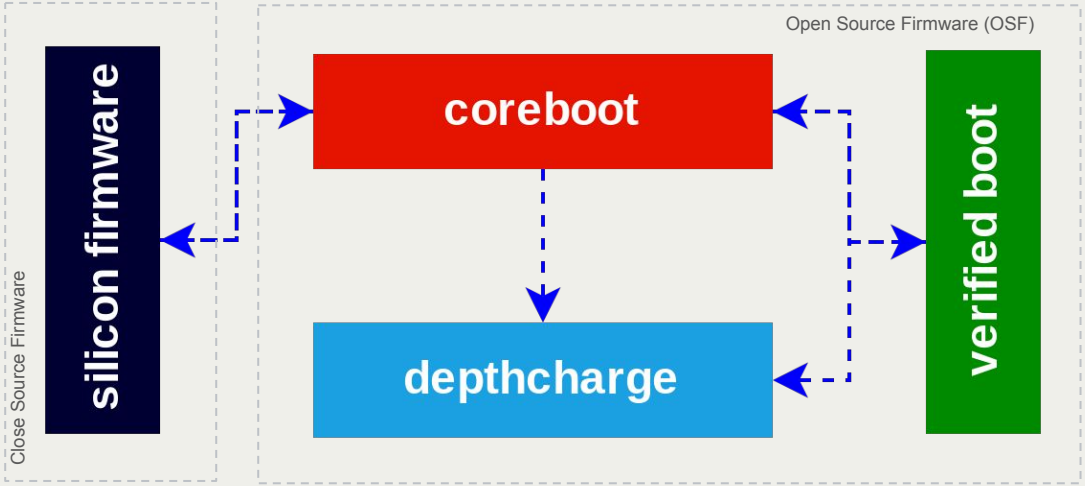
Application Processor (AP)

AP refers to the **Application Processor** or the CPU, AP Firmware refers to the firmware that runs on the AP Processor.

Goal of AP firmware is to perform **mandatory HW initialization** prior **boot to the OS**.

AP Firmware Components

- coreboot
- depthcharge
- verified boot (vboot)
- silicon firmware



coreboot

Core (essential, stripped-down firmware) boot (to boot a platform)

- GPLv2 BIOS replacement
  - Started as LinuxBIOS in 1999 by Ron Minnich
  - Renamed to coreboot in 2008 by Stefan Reinauer
- Mostly C, assembly and ASL
- NOT a bootloader
  - Support for various payloads.
  - Payload can boot any OS. For Chrome OS, this payload is depthcharge.

chromiumos/third_party/coreboot.git

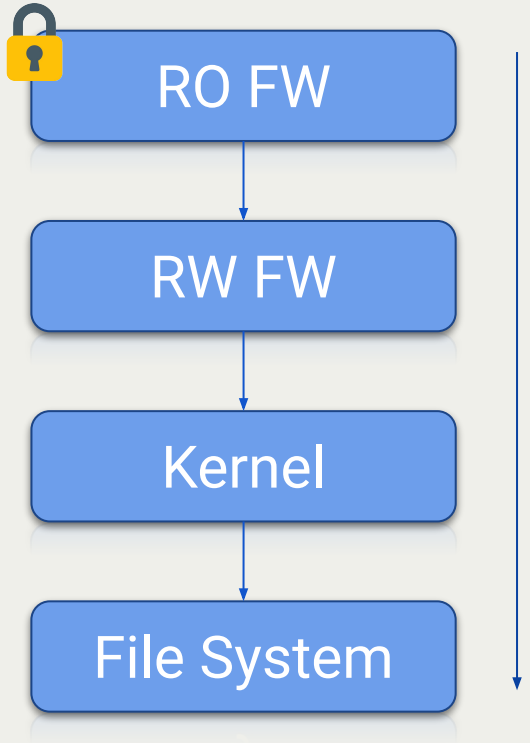
coreboot

- Speed
- Security
- Simplicity
- Open Source

chromiumos/third_party/coreboot.git

Verified Boot (vboot)

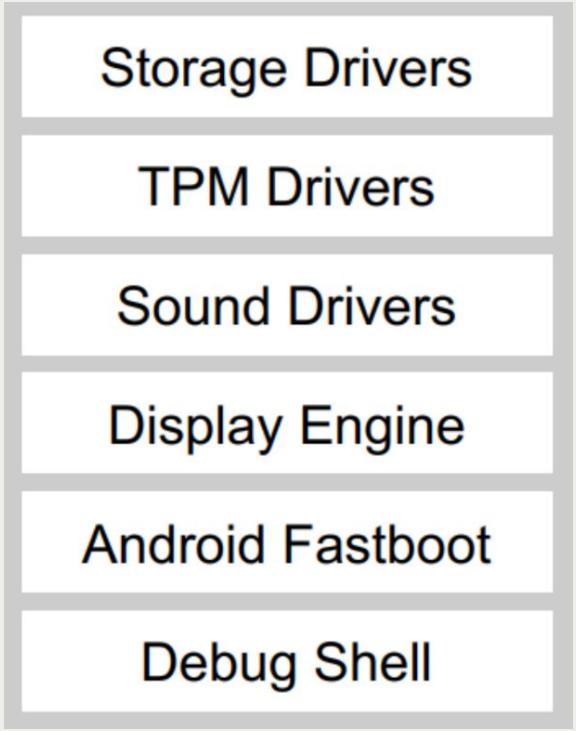
- Makes Chromebook a **secure** computing device.
- Only execute Google signed code.
- **Root of Trust** is in Read-Only (RO) Firmware
  - Hardware Write Protection (aka Locked)
  - Reset vector must be in RO flash
- RO firmware verifies signed RW firmware
- RW firmware verifies signed kernel



chromiumos/platform/vboot_reference.git

depthcharge

- GPLv2 license
- Payload designed to boot Chrome OS
- Goal: Boot ChromeOS quickly
- Verifies Kernel Slot and jumps to it
  - Verified Mode
  - Recovery Mode
  - Developer Mode



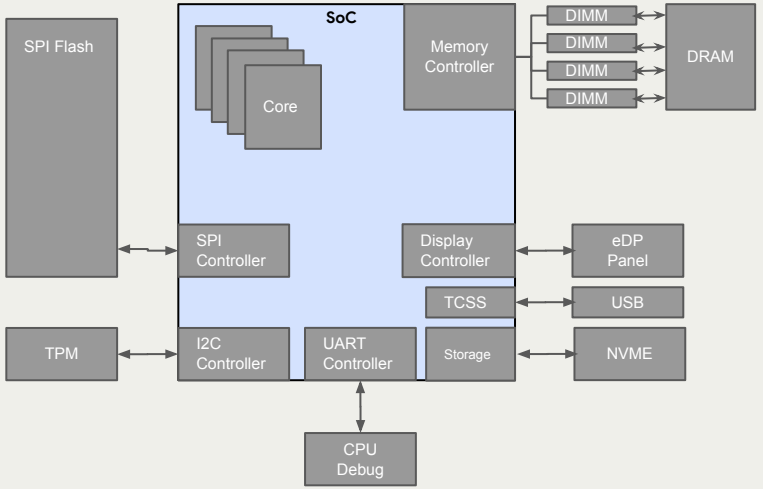
chromiumos/platform/depthcharge.git

Google

Intel SoC Platform Boot Flow

Figure shows Reference Hardware Block

- **CPU**: Intel 12th Generation Heterogeneous Processor.
- **Peripheral Attached:**
  - Dual Channel Hynix 8GB LPDDR4 Memory Down solution
  - USB device configuration
  - LPSS device configuration
    - I2C0-2 Enable
    - UART2 Enabled for serial debug
  - Display Configuration - eDP
  - Storage device configuration - NVME
- **Chrome AP Firmware:**
  - coreboot and silicon binary

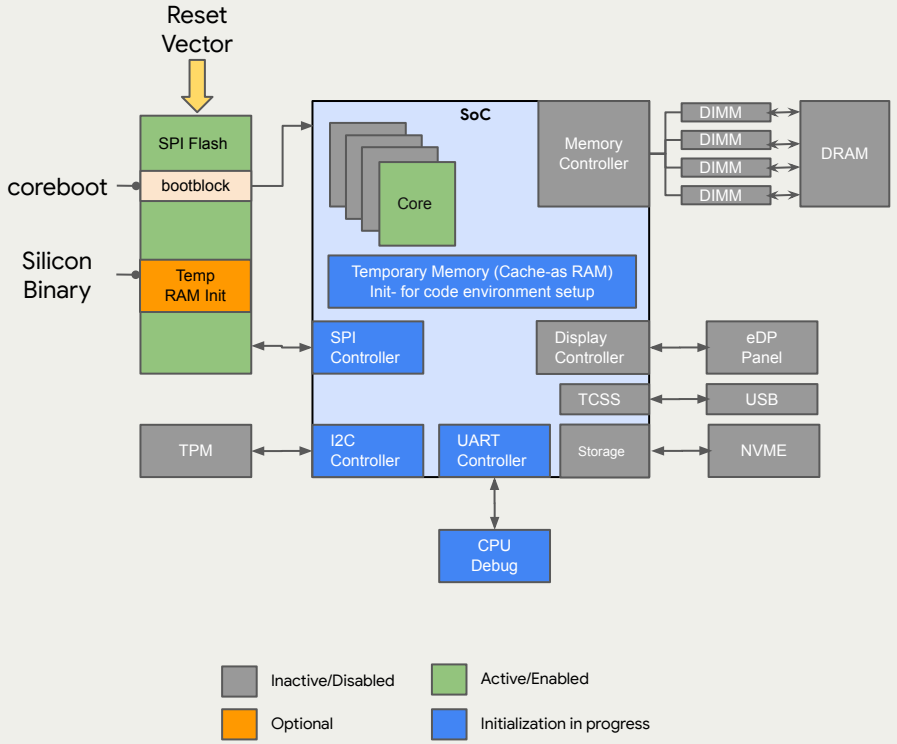


Google

Step 1: Pre-Memory Initialization Phase

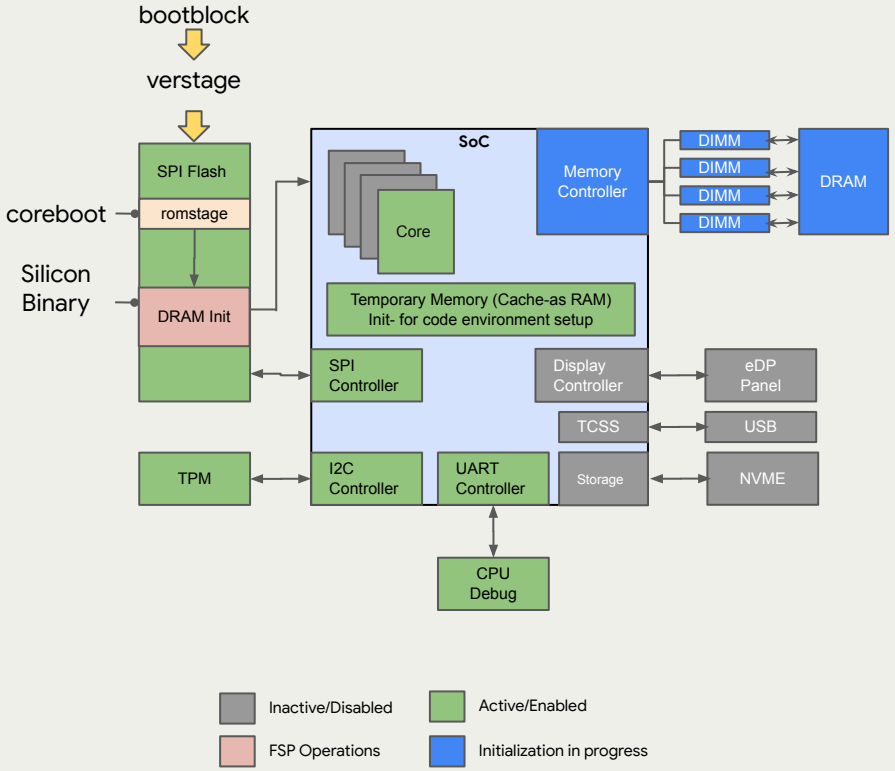
- CPU comes out from reset and starts execution from coreboot (bootblock).
- Setting up the temporary memory (a.k.a CAR) for code execution.
- Early initialization of boot critical devices.
- Decompress and load the next stage.

Note: “Silicon Binary Temp RAM Init” is capable of performing the CAR setup but excluded in Chrome AP Firmware.



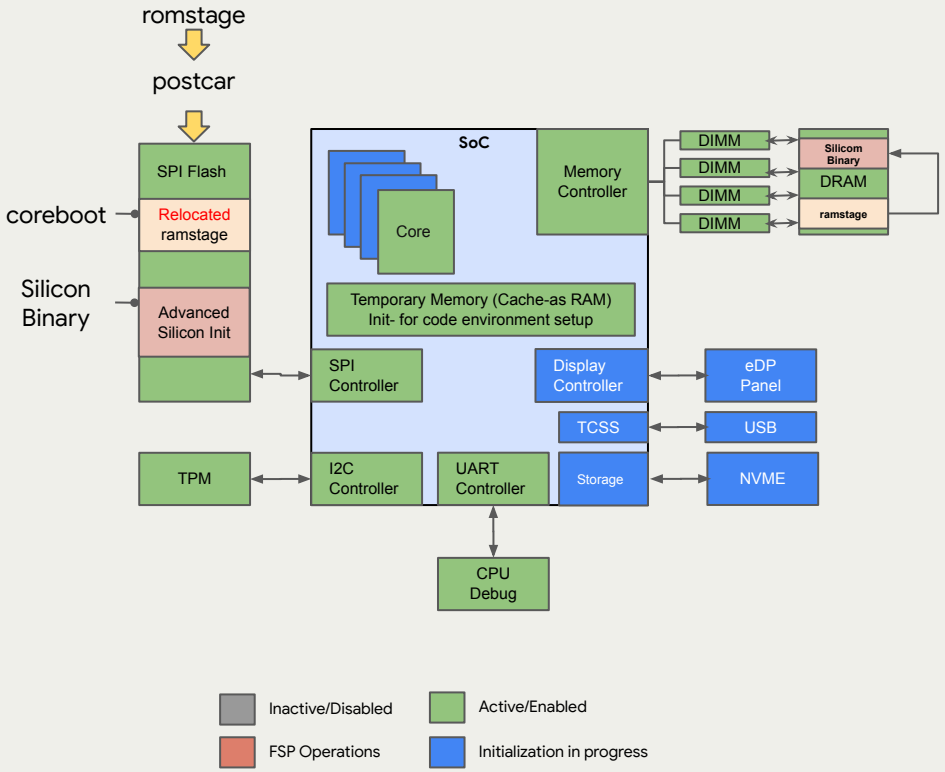
Step 2: Memory Initialization Phase

- coreboot (romstage) performs basic hardware initialization before calling into silicon binary APIs.
- Load and execute DRAM Memory Init API.
- Silicon Binary performs DRAM Initialization by running “closed-source” SoC vendor specific routines.
- coreboot creates cbmem based on DRAM resources.
- Migrate program execution into physical memory.



Step 3: Post-Memory Initialization Phase

- coreboot (ramstage) performs multiprocessor initialization.
- Load and execute advanced Silicon Init using silicon binary.
    - Like chipset Initialization (i.e., Graphics, Storage, Display, USB-C etc.).
- Transfer control to payload (depthcharge) at end of ramstage.



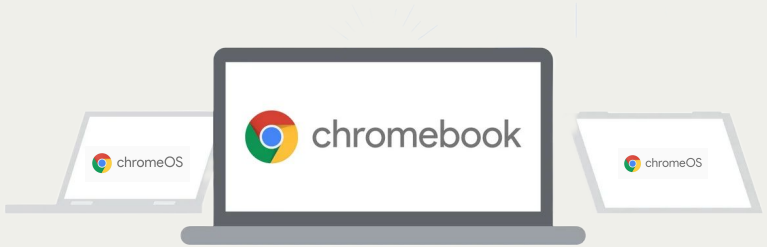
What Makes Chromebook Different ?

Boot Modes

Verified Mode	Recovery Mode	Developer Mode

Verified Mode

- Off the shelf Chromebook boots in Verified Mode.
- Popularly known as normal mode or secure mode.
- Can only boot Google signed OS images.

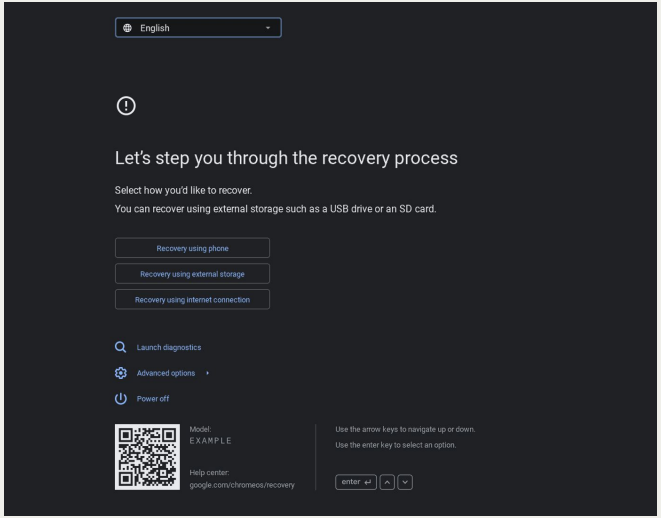


Boot Modes

Verified Mode	Recovery Mode	Developer Mode

Recovery Mode

- Recovery Mode; recovers the device
- Possible reason:
  - Verification Error
  - Corruption of images
  - Hardware issue
- Use recovery media (USB, SD card, network boot) to restore the device.



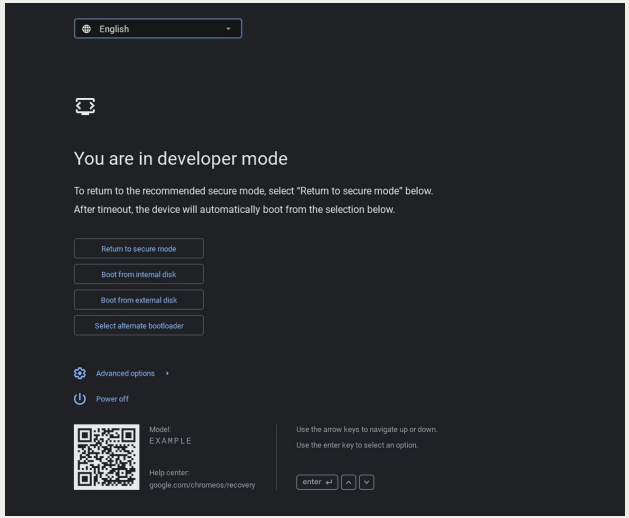
Recovery Screen

Boot Modes

Verified Mode	Recovery Mode	Developer Mode

Developer Mode

- Kernel Verification is disabled
  - Developer warning screen at every boot
- Physical presence is required to enable
  - Confirmation through a trusted source
  - Use for debugging and running more advanced debug/developer tools
- Enable with Ctrl+D on recovery screen.



Developer Screen

Introduction to Power Delivery (PD) Firmware

Why USB and USB PD?

Laptop ports before USB

Chargers before USB PD

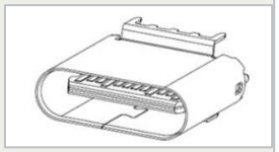
USB Basics

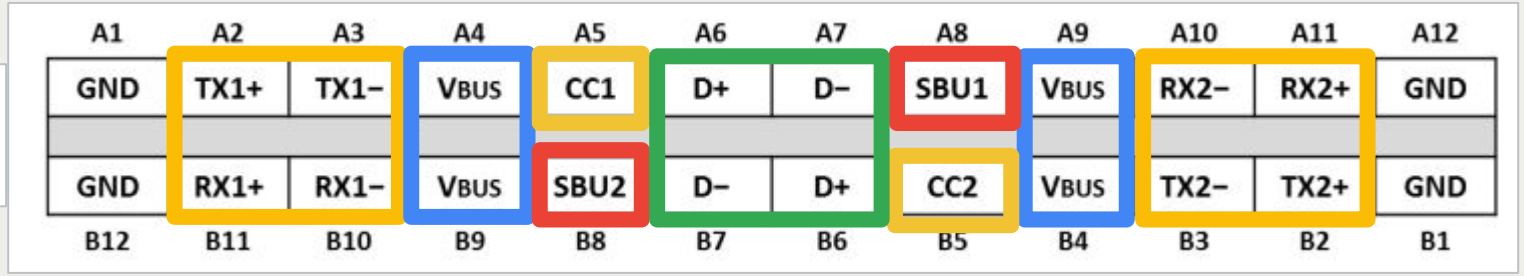
- Universal Serial Bus (USB) is an industry standard that defines cables, connectors, ports, and communication protocols used to connect wide range of devices to a host
- Set of specifications published by USB Implementers Forum (USB-IF)
- Provides testing and certification infrastructure to ensure interoperability
- Terminologies
  - USB Type A, Type B (micro, mini), & Type C are the connectors
  - USB 1.0, USB 2.0, USB 3.0, & USB4 are protocol revisions
  - USB Power Delivery (USB PD) is a specification

Evolution of USB

USB 1.0	USB 2.0	USB 3.0	USB 3.2 Gen 2x2	USB4
Full Speed Data rate: 12 Mbps Power: 2.5W (5V, 500mA)	High Speed Data rate: 480 Mbps Power: 2.5W (5V, 500mA)	Super Speed Data rate: 5 Gbps Power: 4.5W (5V, 900mA)	Super Speed + Data rate: 20 Gbps *Requires USB-C Power: Upto 240W (48V, 5A) **Requires USBPD	Data rate: 40 Gbps Power: Upto 240W (48V, 5A) **Requires USB-C and USBPD

USB Type-C Connector





D+, D-: USB 2.0 Functionality

V_{BUS}: USB bus power

CC1, CC2: Configuration channel

TX, RX: USB 3.2 SuperSpeed and USB4

SBU1, SBU2: Sideband Use

Figures from USB Type-C Cable and Connector Specification, Release 2.0 available on usb.org

USB Power Delivery (USB PD)

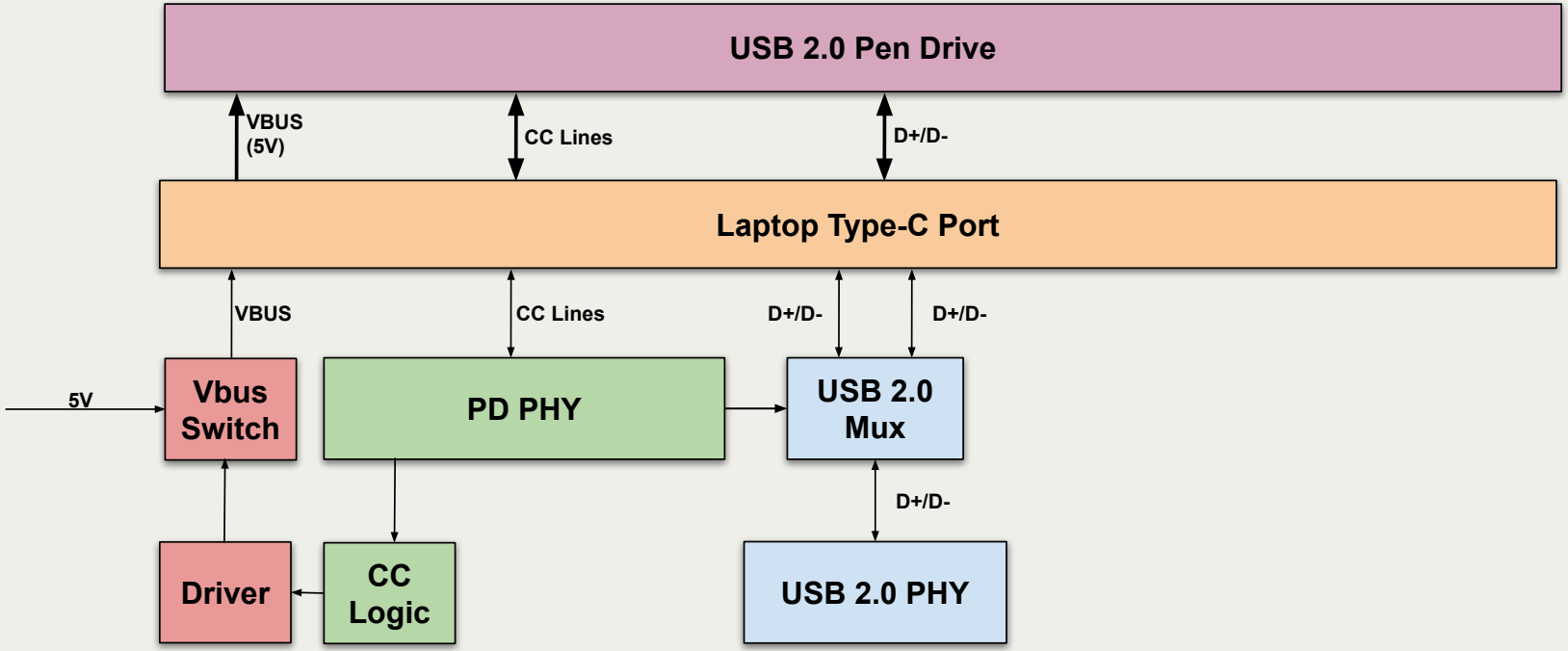
- A specification that enables maximum functionality of USB using the USB Type-C connectors

- Features of USB PD:
  - Protocol communication over Configuration Channel (CC) lines
  - Bi-directional power - A port can act as both power Source and Sink
  - Bi-directional data - A port can act as both Host and Device
  - Supports different voltages (3.3V - 48V) and different currents (Upto 5A)
  - Alternate mode support (Thunderbolt, DisplayPort, HDMI, VirtualLink)

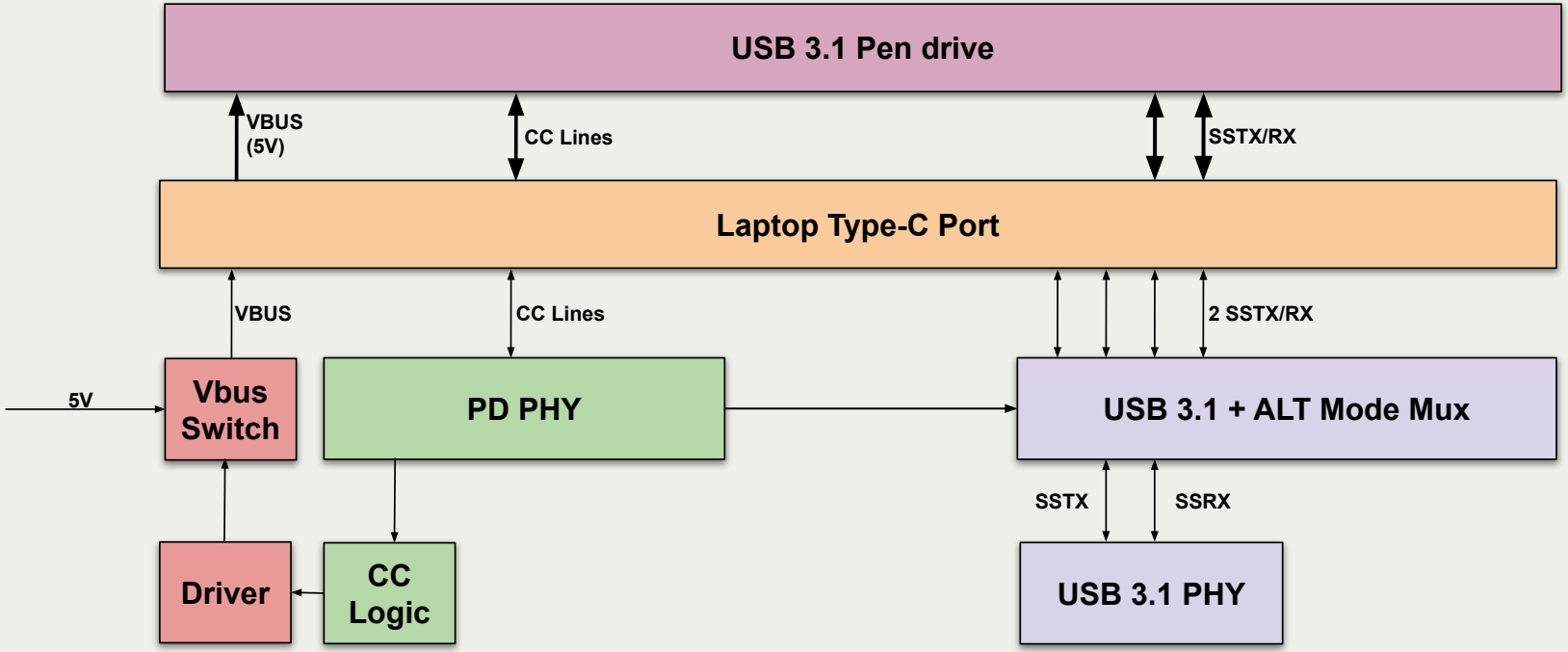
USB PD Messages

- Source capabilities
  - Advertised by power source.
- Request
  - Sent by power sink. E.g. (12V, 3A)
- Data role swap
  - Sent by data host or device to swap data roles
- Accept or Reject
  - In response to the Request or Data role swap message
- Discover Identity
  - To discover additional information of the port partner like alternate modes supported etc.
- Enter/Exit Alternate mode
  - To enter/exit alternate mode like Thunderbolt, DisplayPort, etc.
- GoodCRC
  - Sent in response to every valid PD message

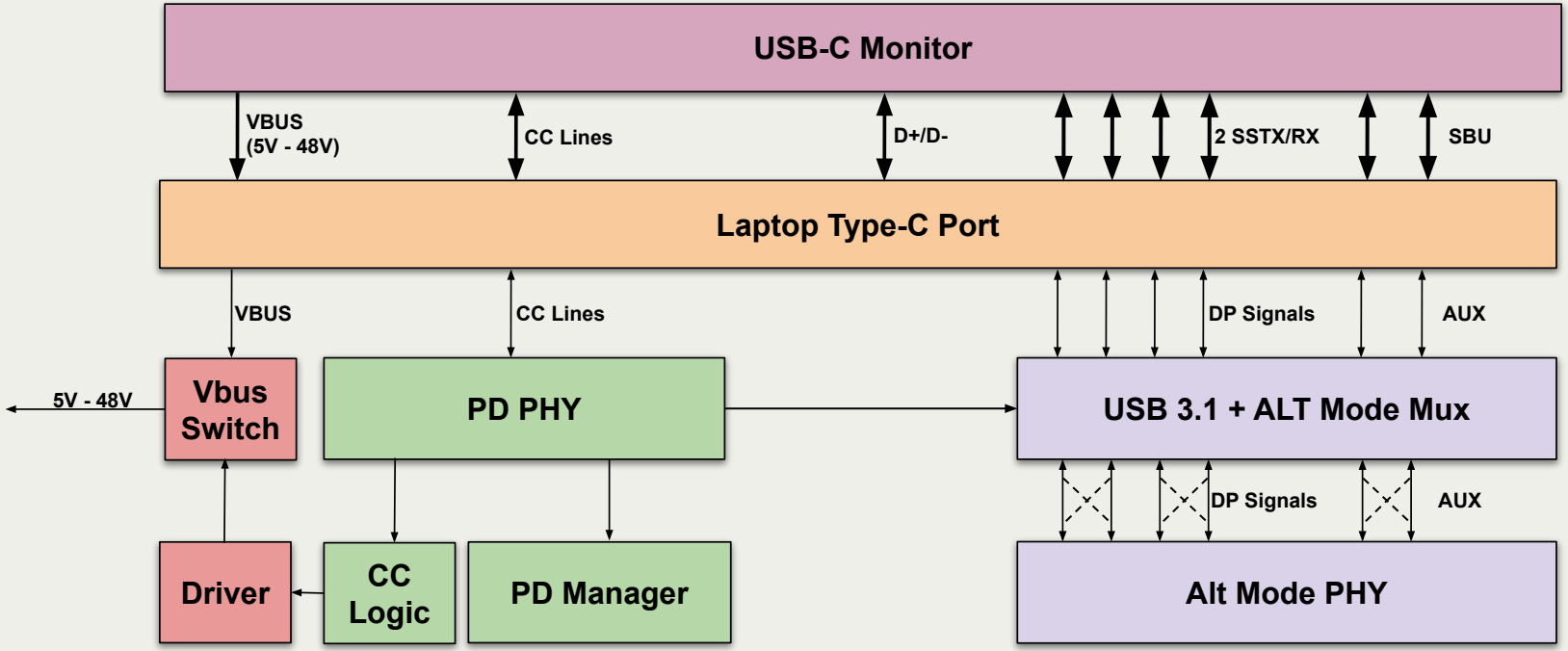
USB 2.0 Device



USB 3.1 Device



USB-C Monitor (DisplayPort)



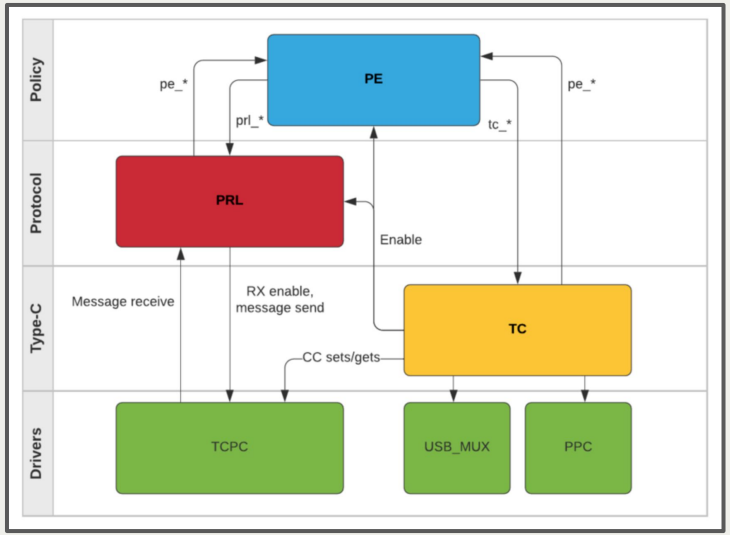
TCPC and TCPM

- **Type-C Port Controller (TCPC)**
  - Dedicated IC for PD PHY layer
  - Exposes necessary terminations based on the port power role
  - Provides PD PHY layer to send and receive USB PD messages
  - Validates CRC of USB PD messages received

- **Type-C Port Manager (TCPM)**
  - Implemented as a part of EC
  - Interprets PD messages received by TCPC and frames a response
  - Advertises port capabilities to the port partner upon connection
  - Determines connection orientation and configures the MUXes
  - Implements device level policy management
  - Exposes communication interface to AP for OS level policy management

ChromeOS firmware for USB PD

- Source code can be found at “src/platform/ec/common/usbc”
- State machine based implementation
- Device Policy Manager Layer
  - Usb_pd_dpm.c, usb_mode.c, *_alt_mode.c
- Policy Engine Layer
  - usb_pe_*_sm.c
- Protocol Layer
  - usb_prl_*_sm.c
- Type-C Layer
  - usb_tc_*_sm.c



Google

Thank you!