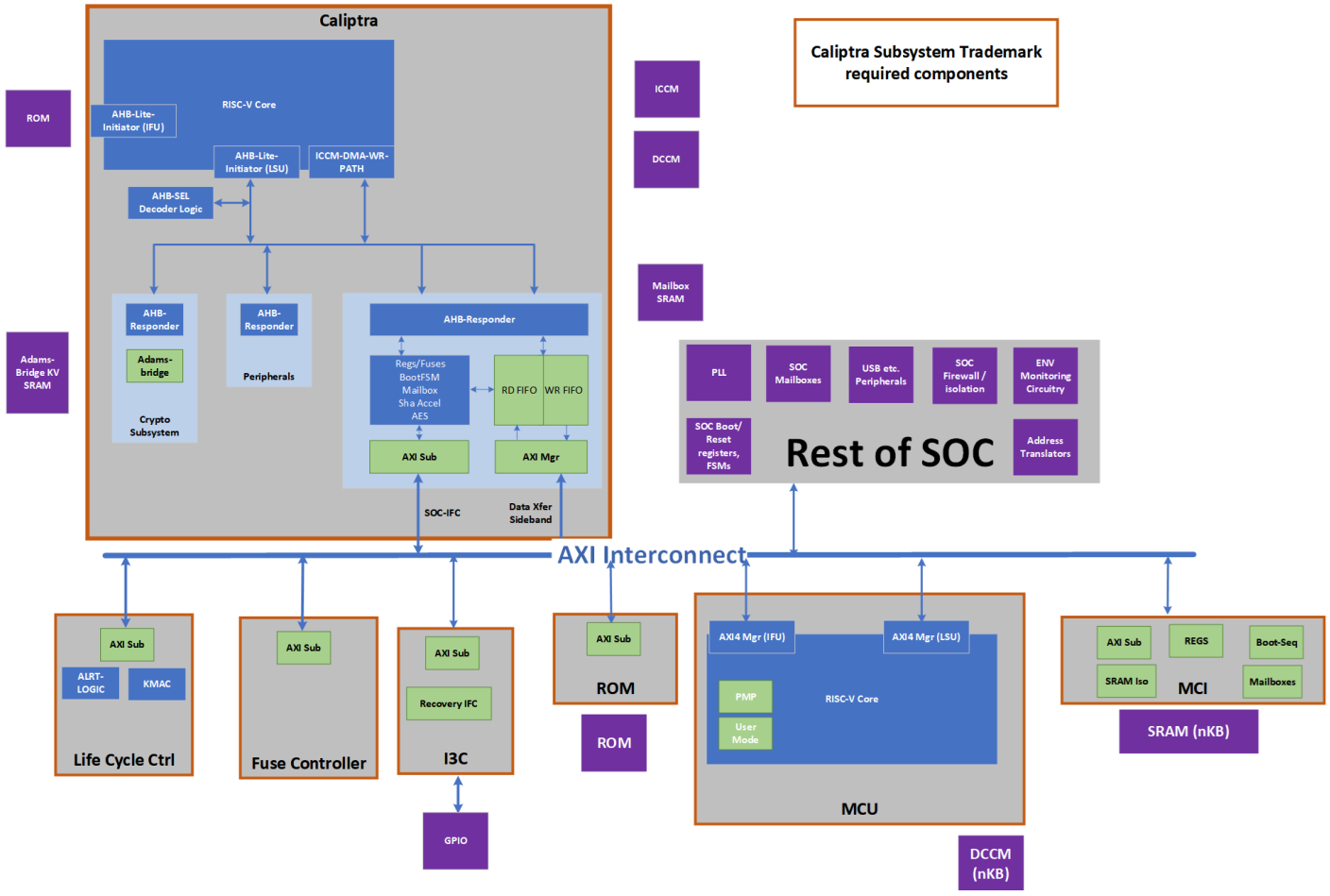
Microsoft

Caliptra – Subsystem Firmware Stack

Caliptra Subsystem

- Caliptra Subsystem
- Enables Caliptra Integrators to build fully featured RoT



Caliptra Subsystem Features

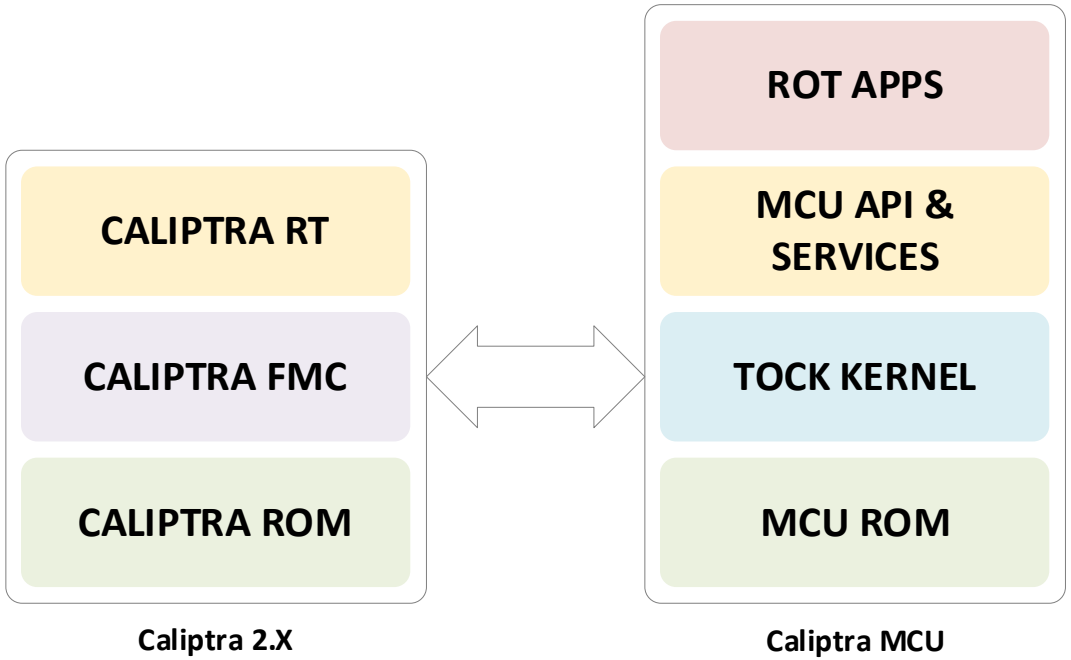
- Caliptra 2.x - Security Features
  - ML-DSA (with Adams bridge integration), DICE extension w/ PQC, AES, Key Vault Extensions for PQC, PCR Signing with PQC
  - Subsystem Mode Support: AXI DMA Assist, Manufacturing & Product Debug Unlock, UDS programming, Streaming boot support in Caliptra
  - Integration of Lifecycle controller & Fuse controller
  - OCP Streaming boot support over I3C
  - MCU & corresponding HW support for running SOC specific FW (whose FW is loaded/bootstrapped by Caliptra)

Design Principles

- Open and Extensible
  - Developed in open on GitHub
  - Provided as SDK to build RoT Applications
  - Extensible and customizable by integrators
- Secure & Safe
  - Follows established security & isolation best practices
  - Memory safe – Developed in Rust
- Consistent
  - Consistent implementation of Secure Boot, Measured Boot, Attestation, Recovery, Streaming Boot, etc.
  - Standards Based: TCG, DMTF, OCP, PCIe
- Compliant
  - OCP SAFE Audited
  - FIPS 140-3 Ready
  - Caliptra Trademark

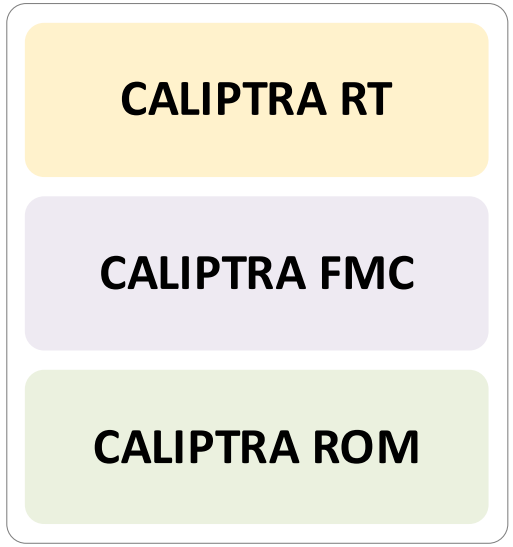
Subsystem Stack

- Combination of Caliptra 2.0 and MCU
- Caliptra 2.0 provides the security services
- Caliptra MCU provides the RoT services for the SoC and Platform



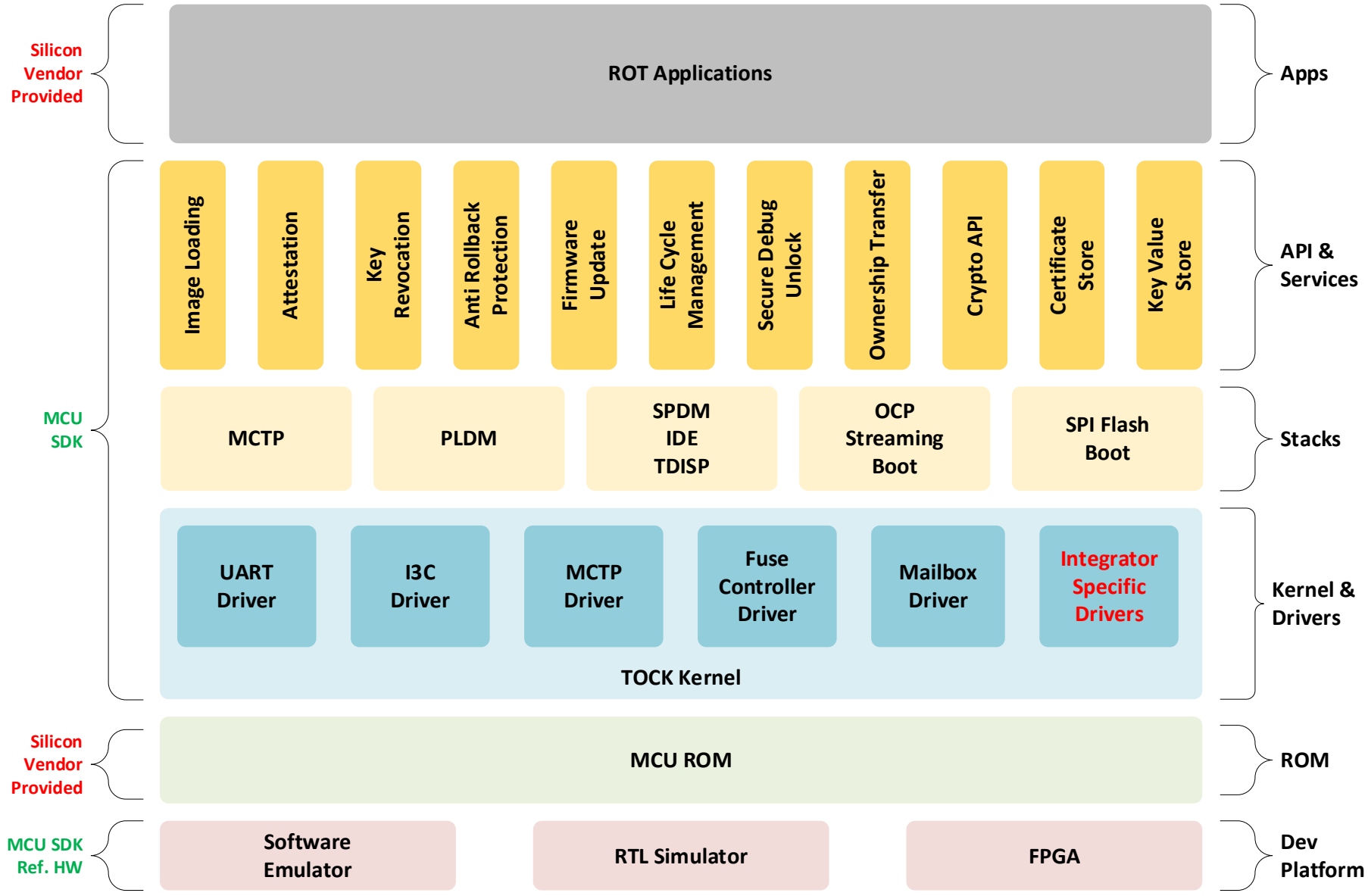
Caliptra 2.x

- Caliptra 2.x comprises of consortium governed ROM, FMC and Runtime
- Caliptra 2.x provides foundational security features for MCU
  - Quantum Resilient DICE
  - OCP Recovery Boot
  - Classical and Post Quantum Cryptography API support
  - Debug Unlock support
  - Quantum Resilient DPE



Caliptra 2.X

MCU Stack


Silicon Vendor Provided
ROT Applications
Apps
Image Loading
Attestation
Key Revocation
Anti Rollback Protection
Firmware Update
Life Cycle Management
Secure Debug Unlock
Ownership Transfer
Crypto API
Certificate Store
Key Value Store
API & Services
MCU SDK
MCTP
PLDM
SPDM IDE TDISP
OCP Streaming Boot
SPI Flash Boot
Stacks
UART Driver
I3C Driver
MCTP Driver
Fuse Controller Driver
Mailbox Driver
Integrator Specific Drivers
TOCK Kernel
Kernel & Drivers
Silicon Vendor Provided
MCU ROM
ROM
MCU SDK Ref. HW
Software Emulator
RTL Simulator
FPGA
Dev Platform

MCU SDK

- Development Tools
  - Emulators, RTL Simulators & FPGA
- Reference MCU ROM
  - Reference MCU ROM implemented in Rust
  - Illustrates integration b/w Caliptra 2.x and MCU
- Tock Kernel
  - Provides user/kernel mode isolation via RISCV MPU
  - Driver Model for extensibility
- Stacks & API
  - MCTP & PLDM
  - PLDM Firmware Update
  - OCP Streaming Boot
  - SPDM based Attestation
  - PCIe IDE & TDISP support
  - SPI Flash Boot
  - Image Loading
  - Firmware Update infrastructure
  - Ownership Transfer

Integrator Extensibility

- ROM Development
  - MCU ROM is integrator specific
  - Ability to extend the reference Rust ROM to build MCU-specific ROM
- Drivers
  - Flexible & async driver model provided by Tock kernel
  - SoC specific drivers (UART, SPI, IDE/TDISP, etc.)
- SOC API & Stacks
  - Ability for SOC to provide their own stack and user mode interfaces
- RoT Applications
  - Allows integrators to synthesize ROT applications specific to their use cases
  - Allows extensibility points for integrator specific message handlers for external communication
  - Enables running of legacy C code
  - Allows multiple application isolation via Tock kernel & MMU

Compliance & Trademark

- MCU SDK will be reviewed, audited and compliant to various standards

- Current Plan is to support compliance with:
  - Caliptra Trademark
  - OCP SAFE
  - FIPS 140-3

- Provide ability to run FIPS ACVP test suite

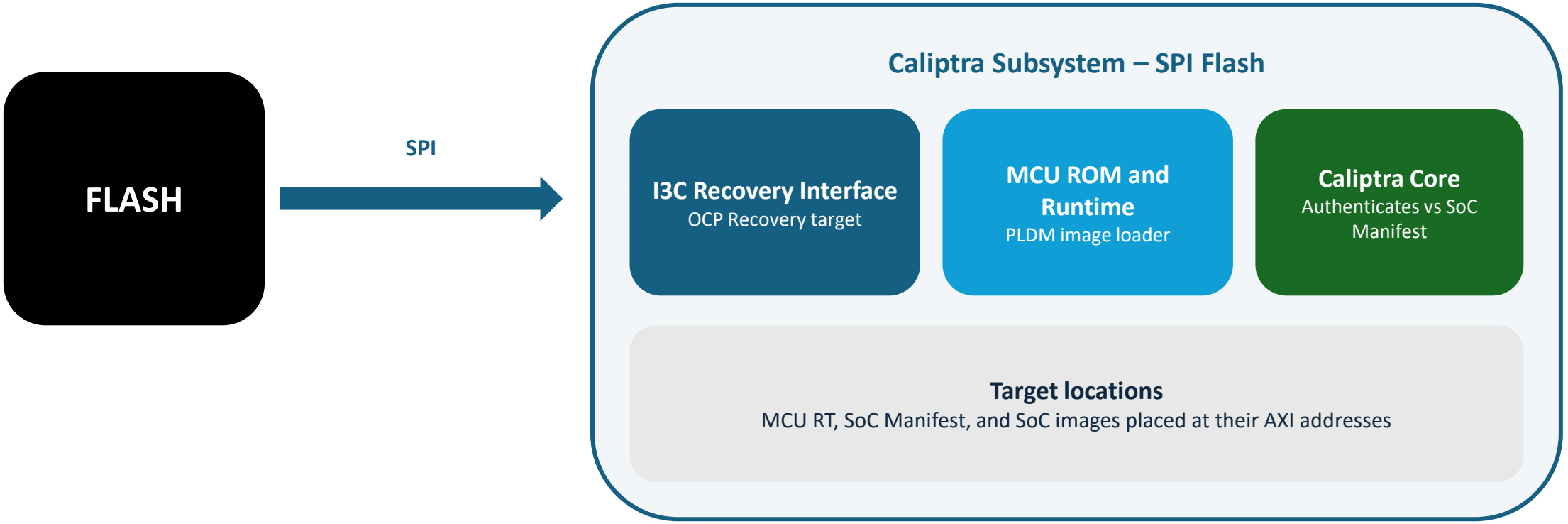
OCP S.A.F.E.

SPI Flash Boot

- Supported Topologies
  - ➢Single flash: one device split into primary (active image) and recovery halves
  - ➢Dual flash: two devices on separate chip selects - flash 0 primary, flash 1 recovery

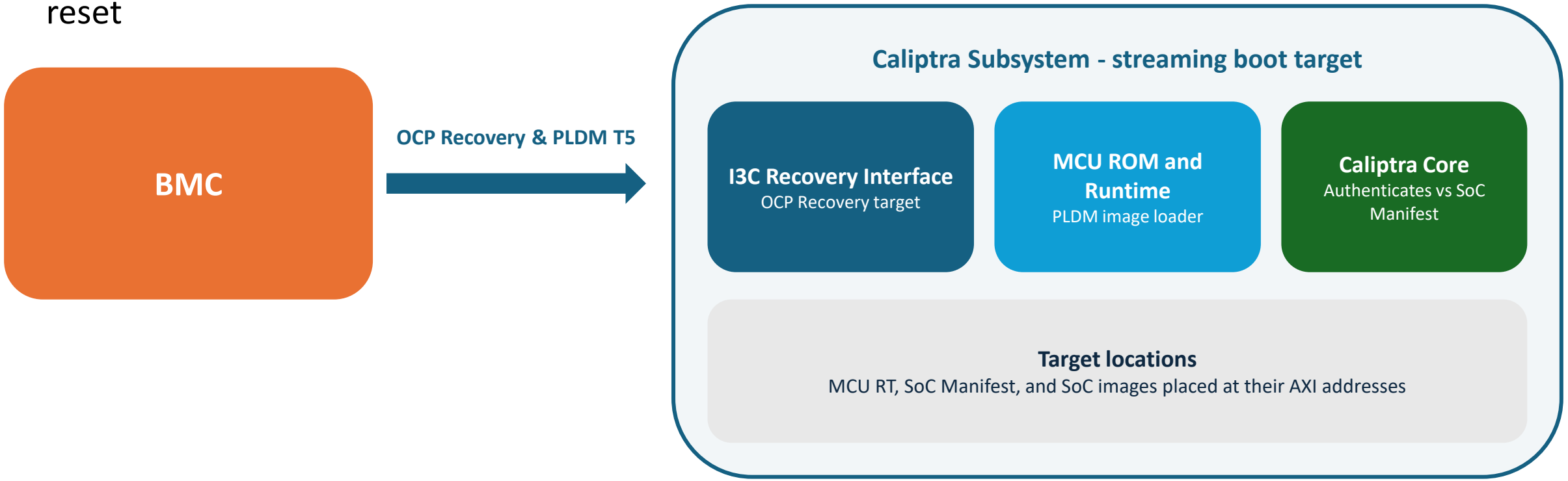
Scaling and Partitions

- ➢Multi flash: multiple controllers and devices for scale and redundancy
- ➢Primary flash may also hold staging, certificate, and log partitions
- ➢Recovery flash always carries a known-good image



I3C Streaming Boot

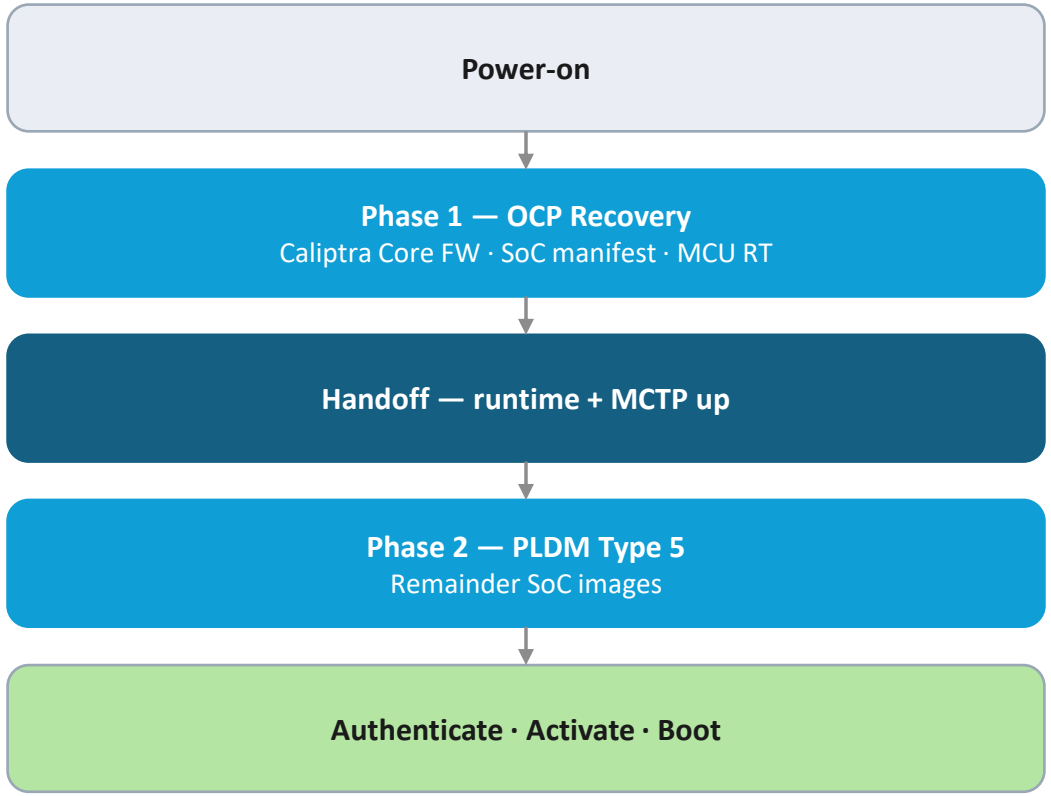
- Recovery Interface
  - I3C interface acts as a target device implementing the OCP Recovery specification
  - MCU ROM initializes I3C and the recovery interface before releasing Caliptra from reset
- Streaming the Images
  - A recovery agent, typically a BMC, streams the PLDM T5 package over I3C
  - AXI streaming bypass lets the MCU write images directly, skipping I3C protocol logic



Streaming Boot: A Two-Phase Secure Boot

The BMC establishes the trusted runtime, then the live MCU pulls the rest.

- **Phase 1 — OCP Recovery:** BMC pushes an ordered image set (Caliptra FW, SoC manifest, MCU runtime) over the register interface. No MCTP/PLDM/runtime yet.
- **Handoff:** MCU runtime starts, exposes an MCTP endpoint; bus ownership changes.
- **Phase 2 — DMTF PLDM Type 5:** MCU pulls remaining SoC images via RequestFirmwareData; BMC acts as Update Agent.
- Caliptra authenticates and gates every stage — streaming boot is the normal boot path.



Contribute to Caliptra

- Install the host tools
- Clone the repository
- Check the development environment
- Run the Caliptra Core and MCU emulators
- Run the emulator integration tests

Step 1: Install the host tools

What to do

- Commands target Linux or WSL.
- Install git, curl, and a C build toolchain with your OS package manager.
- Install rustup, which manages the Rust compiler and Cargo; choose the default option.
- Open a new terminal or load Cargo into the current shell, then confirm the version.

Console output

```
$ sudo apt update
$ sudo apt install -y build-essential curl git
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
Rust is installed now. Great!
$ source "$HOME/.cargo/env"
$ cargo --version
cargo 1.85.1 (d73d2caf9 2024-12-31)
```

Step 2: Clone the repository

What to do

- Clone with --recursive so the Caliptra hardware submodule comes down too.
- Confirm submodules with git submodule status --recursive; each initialized line begins with a space.

Console output

```
$ git clone --recursive \
  https://github.com/chipsalliance/caliptra-mcu-sw.git
Cloning into 'caliptra-mcu-sw'...
remote: Total 57699 (delta 1967), pack-reused 54872
Receiving objects: 100% (57699/57699), 45.65 MiB, done.
Submodule 'hw/caliptra-ss' registered for path 'hw/caliptra-ss'
$ cd caliptra-mcu-sw
$ git submodule status --recursive
 b5718a0f hw/caliptra-ss (css-v2.0.2)
 c9d45534 .../caliptra-rtl (v2.0.4)
 db4a6f34 .../i3c-core (i3c-v1.5.0)
```

Step 3: Check the development environment

What to do

- From the repository root, confirm the pinned compiler with rustc --version.
- Optionally run the full pre-submit check: cargo xtask precheckin.
- It builds firmware and runs format, lint, unit, and integration checks, so the first run takes several minutes.

Console output

```
$ rustc --version
rustc 1.85.1 (4eb161250 2025-03-15)
$ cargo xtask precheckin
  Finished `release` profile [optimized + debuginfo] in 11.40s
  Finished `release` profile [optimized + debuginfo] in 2.64s
  Finished `dev` profile [unoptimized + debuginfo] in 14.32s
  Finished `release` profile [optimized + debuginfo] in 8.91s
$ echo $?
0
```

Step 4: Run the emulators

```
$ cargo xtask runtime
```

Ubuntu-26.04

```
elf/devel/runtime-emulator.bin --caliptra-rom /home/vsoni/caliptra-mcu-sw/target/caliptra-rom-a0bf91a33752368870000bf7255976452160db5a-emulator.bin --caliptra-firmware /home/vsoni/caliptra-mcu-sw/target/c
aliptra-fw-bundle-a0bf91a33752368870000bf7255976452160db5a-emulator-0-key1.bin --soc-manifest /home/vsoni/caliptra-mcu-sw/target/soc-manifest --vendor-pk-hash b17ca877666657ccd100e6926c7206b60c995cb68992c
6c9baefce728af05441dee1ff415adfc187e1e4edb4d3b2d909 --hw-revision 2.0.0 --rom-offset 0x80000000 --rom-size 0x10000 --dccm-offset 0x50000000 --dccm-size 0x4000 --sram-offset 0x40000000 --sram-size 0x100000
 --pic-offset 0x60000000 --i3c-offset 0x20004000 --i3c-size 0x1000 --mci-offset 0x21000000 --mci-size 0xe00000 --mbox-offset 0x30020000 --mbox-size 0x28 --soc-offset 0x30030000 --soc-size 0x5e0 --otp-offs
et 0x70000000 --otp-size 0x140 --lc-offset 0x70000400 --lc-size 0x8c --device-security-state 3`
Loaded ROM File "/home/vsoni/caliptra-mcu-sw/target/riscv32imc-unknown-none-elf/release/mcu-rom-emulator.bin" of size 65536
Starting I3C Socket, port 0
Starting DOE mailbox transport thread
[MCI] Registering outgoing events
[MCI] Registering outgoing events
Active mode enabled with 3 recovery images
[mcu-rom] Hello from ROM
[mcu-rom] Device lifecycle: Production
[mcu-rom] MCI generic input wires[0]: 00000000
[mcu-rom] MCI generic input wires[1]: 00000000
[mcu-rom] MCI RESET_REASON: 0x00000000
[mcu-rom] Cold boot detected
[mcu-rom] Starting cold boot flow at time 2921
[mcu-rom] Active I3C core for recovery: 0
[mcu-rom] Setting Caliptra boot go
[emulator bmc recovery] Recovery state transitioo: ReadProtCap -> ReadDeviceStatus
[mcu-lcc] Initializing Lifecycle controller..
[mcu-lcc] Lifecycle state: 00000011 (count 1)
Lifecycle controller inittialization done
[mcu-rom-otp] Initializing OTP controller...
[mcu-rom-otp] Setting check timeout to 1048576
[mcu-rom-otp] Disabling check modifications
[mcu-rom-otp] Done init
[mcu-rom] OTP initialized
[mcu-rom] Configurating Caliptra watchdog timers for streaming boot: 800000000 800000000
[mcu-rom] Initializing I3C
[mcu-rom-i3c] HCI version: 120
[mcu-rom-i3c] Set TTI RESET_CONTROL: 0000003F
[mcu-rom-i3c] Initialize timing registers
[mcu-rom-i3c] Timing registers t_r: 0, t_f: 0, t_hd_dat: 0, t_su_dat: 0, t_higl.. 0, t_low: 0, t_hd_sta: 0, t_su_sta: 0, t_su_sto: 0, t_free: 200
[mcu-rom-i3c] Setup HCI queue thresholds
[mcu-rom-i3c] Setting main target DCR to cc
[mcu-rom-i3c] Enable the target transaction interface
[mcu-rom-i3c] STBY_CR_CONTROL: 8000d000
[mcu-rom-i3c] STBY_CR_CAPABILITIES: 1000
[mcu-rom-i3c] Setting static address to 3a
[mcu-rom-i3c] Setting virtual device static address to 3b
[mcu-rom-i3c] Set TTI queue thresholds
[mcu-rom-i3c] TTI queue thresholds: 01000101
[mcu-rom-i3c] TTI data buffer thresholds ctrl: 01010101
[mcu-rom-i3c] Reset indirect fifo ctrl
[mcu-rom-i3c] Enable PHY to the bus
[mcu-rom] Waiting for Caliptra to be ready for fuses: true
[mcu-rom] Writing fuses to Caliptra
[mcu-rom] Setting Caliptra mailbox user 0 to CCCCCCCC
[mcu-rom] Locking Caliptra mailbox user 0
[mcu-rom] Setting Caliptra mailbox user 1 to DDDDDDDD
[mcu-rom] Locking Caliptra mailbox user 1
[mcu-rom] Setting fuse user
[mcu-rom] Locking fuse user
[mcu-rom] Setting TRNG user
[mcu-rom] Locking TRNG user
[mcu-rom] Setting DMA user
[mcu-rom] Populating fuses
[mcu-fuse-write] Setting UDS/FE base address to 48
[mcu-fuse-write] Setting UDS/FE DAI idle bit offset to 22, OTP status reg offset to 16, and direct access cmd reg offset to 96
[mcu-fuse-write] Selected vendor PK slot 0
[mcu-fuse-write] Setting vendor PQC type to 3
```

Step 5: Run the emulator tests

What to do

- Run cargo xtask test to build the required firmware and run the integration tests on the MCU emulator through cargo nextest.
- The run succeeds when it exits with status 0 and the summary reports no failed tests.
- Output below is from one emulator-test shard of a successful upstream run.

Console output

```
$ cargo xtask test
PASS [ 17.910s] (134/146) test_i3c_simple
PASS [ 23.832s] (142/146) test_successful_boot_hw_model
PASS [  2.652s] (144/146) test_usb_ocp_recovery_prot_cap
Summary [1134.711s] 146 tests run: 146 passed (5 slow),
  1144 skipped
PASS [ 19.707s] (1/2) test_defmt_logging_release
PASS [ 38.817s] (2/2) test_flash_soc_boot_successful
Summary [58.527s] 2 tests run: 2 passed, 217 skipped
```

References

Please join us in co-developing an industry compliant RoT SDK

https://caliptra.io/

Specs

- Main Caliptra specification 2.0
- Caliptra Subsystem Hardware Specification
- ROM 2.x Specification
- FMC 2.x Specification
- Runtime 2.x Specification
- MCU Firmware and SDK specification