How ARM systems are booted

An introduction to the ARM boot flow

Rouven Czerwinski – r.czerwinski@pengutronix.de

https://www.pengutronix.de

About Me

Rouven Czerwinski

Pengutronix e.K.

Emantor

rcz@pengutronix.de

Testing

OP-TEE

System Integration

Media Systems

Short Disclaimer

How some ARM systems are booted (mostly looking at i.MX8M*)

Consult your vendor documentation!

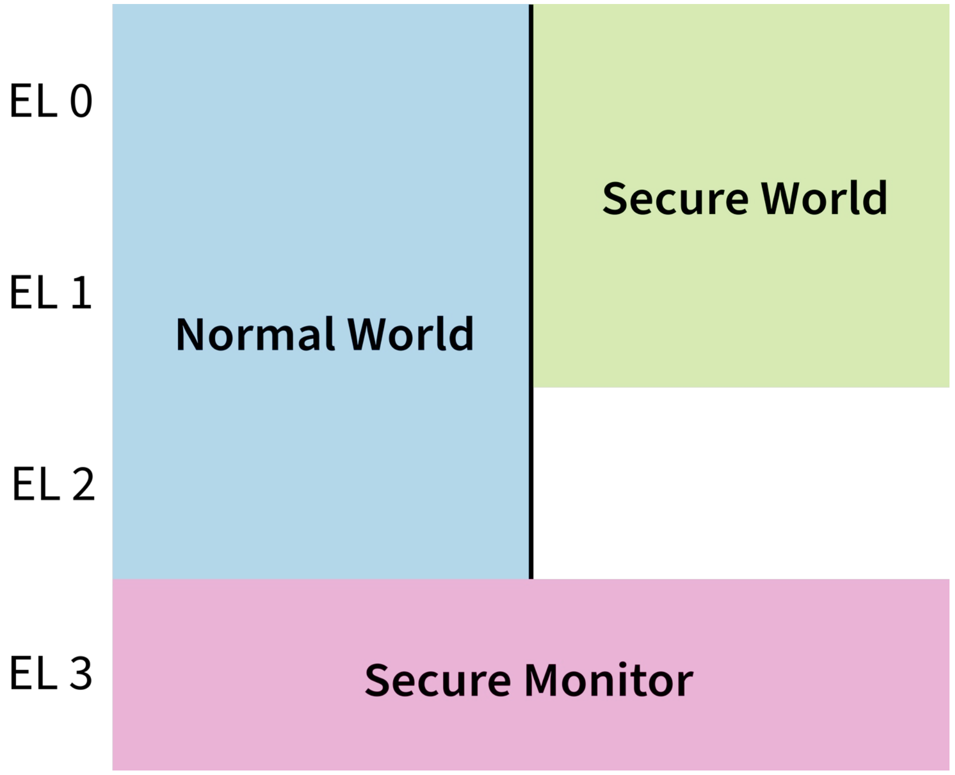
Implementations

- Implementations are vendor specific
    - NXP, Nvidia, Qualcomm, Samsung, ARM
- ARMv7 is not looked at
    - Vendor specific kernel drivers, specific suspend/idle drivers
- ARMv8 de-factro standardized on ARM Trusted Firmware (TF-A)

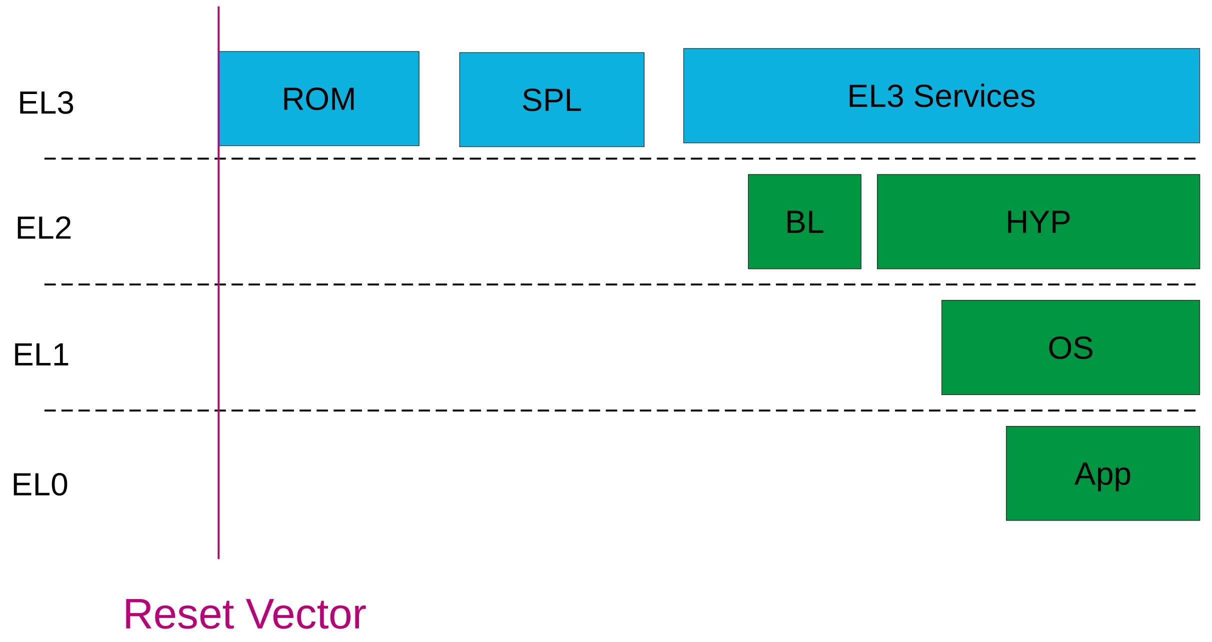
Table of Contents

- Introduction
- Exception Levels
- Requirements for booting
- TF-A services
- Kernel Start Sequence

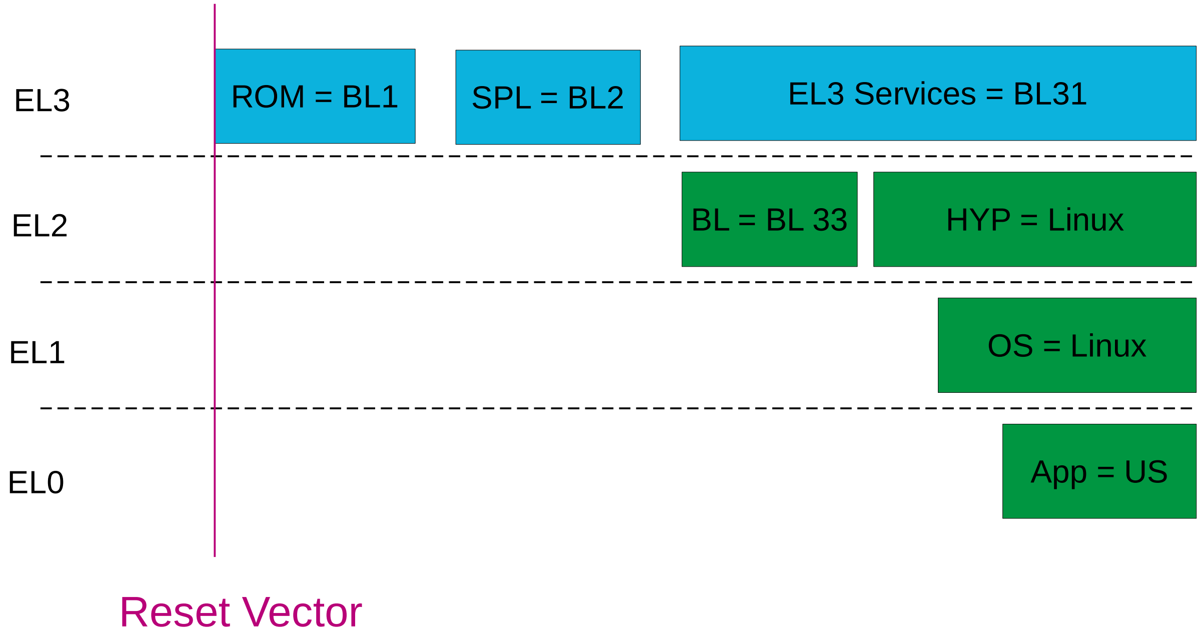
Exception Levels



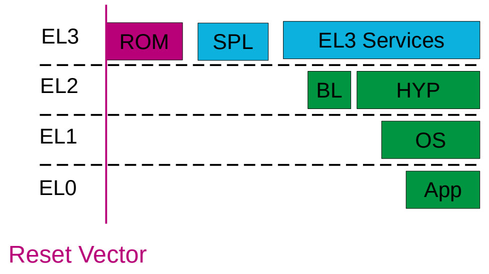
Exception Levels & Binary Naming Overview



TF-A naming scheme

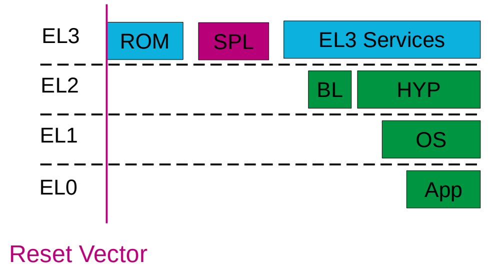


First Stage (BL1): ROM code



- Fused into the SoC
- Uses SRAM for memory if required
- Implements vendor specific storage access/next stage loading
- May implement USB/Serial Upload of Next Stages, networking
- Smaller = better (less size == less cost)
- Use sane default settings: Clocks, Power,

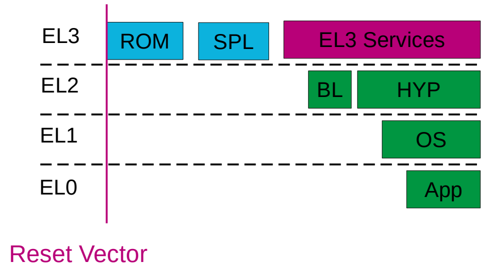
Second Stage (BL2) : TF-A/U-Boot SPL/Barebox PBL



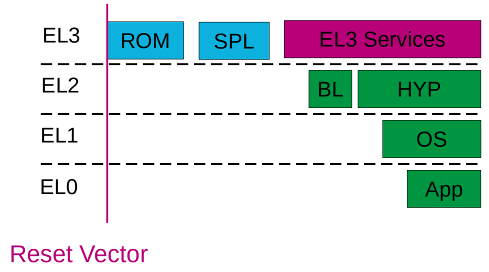
- Loaded by first stage, less restrictions
- Needs to setup RAM training
- Will load the Next Stage again from storage medium
- Can be:
  - U-Boot SPL → TF-A → U-Boot
  - Barebox PBL → TF-A → Barebox
  - TF-A BL2 → BL33 (U-Boot, Barebox)

Arm Trusted Firmware (TF-A)

- Framework to implement standard firmware services
  - PSCI, SCMI,...
  - EL3 Secure Monitor and SIP Router
- Can be used as BL2 and/or BL3* dispatcher
- MIT Licensed
  - Which means the source code is not always available...



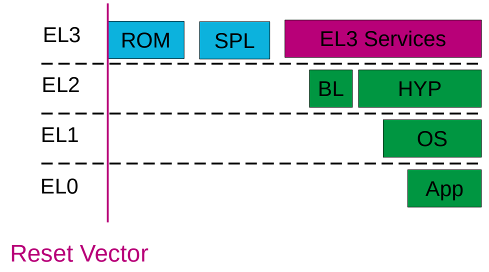
TF-A: SiP

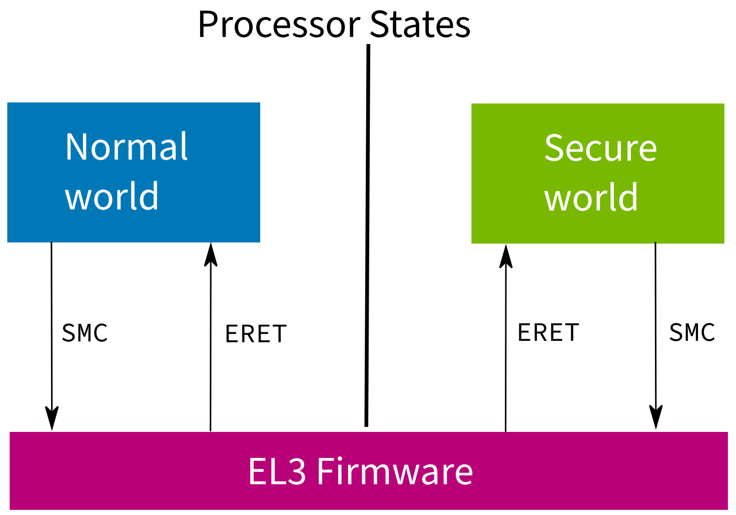


- Silicon Provider Service
- Implement SoC specific Services
  - HAB (Secure Boot) API for i.MX8M*
  - DDR Frequency Scaling
- Often require higher exception level access

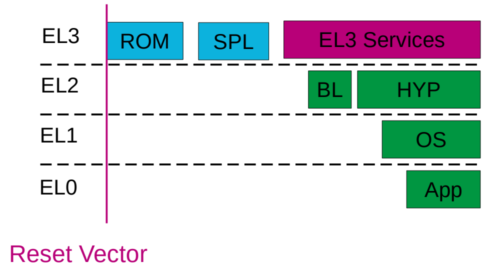
ARM SMC Calling Convention

- How do Exception Levels communicate?
  - SMC – Secure Monitor Call
  - ERET – Exception Return
- Communication via Registers
- Defines argument register placement



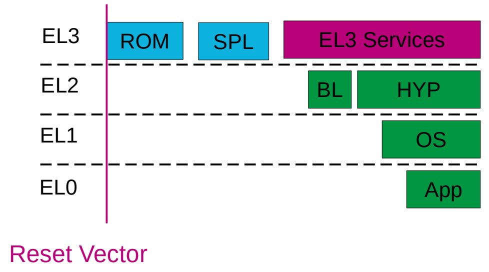


TF-A Services: PSCI



- Power State Coordination Interface
- Unfified discoverable API for
  - CPU on/off
  - System Sleep
  - CPU Idle
- Standard on ARMv8 Systems, usage on ARMv7 possible

TF-A Services: SCMI



- System Control and Management Interface
- Unified discoverable API for clocks, power,. . .
- Useful since Normal and Secure World may require clocks
- Simplified control interface for the linux kernel

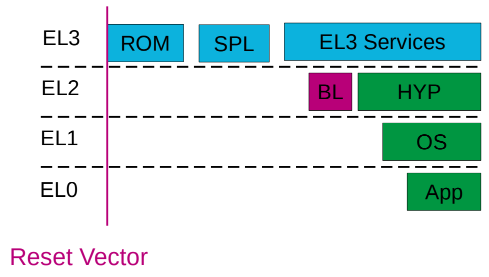
Excursion: Device Trees

- Many slightly different SoC versions
- Shared components across SoC generations
  - i.MX6 → i.MX8 UART
  - i.MX8 HW-Encoder → RK Encoder
- Solution: Device Trees

```
/dts-v1/;

#include "imx8mq.dtsi"

/ {
	model = "NXP i.MX8MQ EVK";
	compatible = "fsl,imx8mq-evk", "fsl,imx8mq";
[…]
&fec1 {
	pinctrl-names = "default";
	pinctrl-0 = <&pinctrl_fec1>;
	phy-mode = "rgmii-id";
	phy-handle = <&ethphy0>;
	status = "okay";

	mdio {
		#address-cells = <1>;
		#size-cells = <0>;

		ethphy0: ethernet-phy@0 {
```

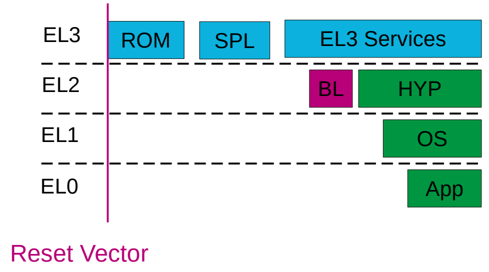
BL33: Barebox Proper

- We are in EL2 now
- Barebox provides additional services
  - Networking/NFS Boot
  - Bootspec Parsing
  - USB Gadget Support for
    - Serial
    - Mass storage
    - fastboot

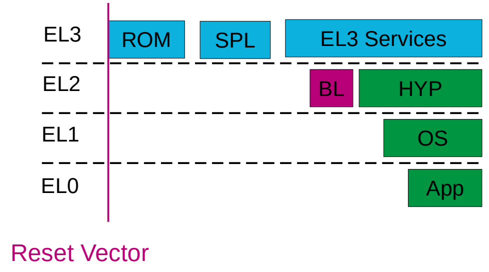


BL33: Kernel Start

- Decompress the kernel
  - Aarch64 does not implement decompression in the kernel
- Copy DTB into memory
- Mask interrupts
- Initialize standard ARM timer



BL33: Kernel Start 2



- Load kernel at offset specified in header
- Disable MMU, Data Caches
- Initialize CPU registers for either EL2 or EL1
  - e.g. x0 for device tree blob
- Jump to it

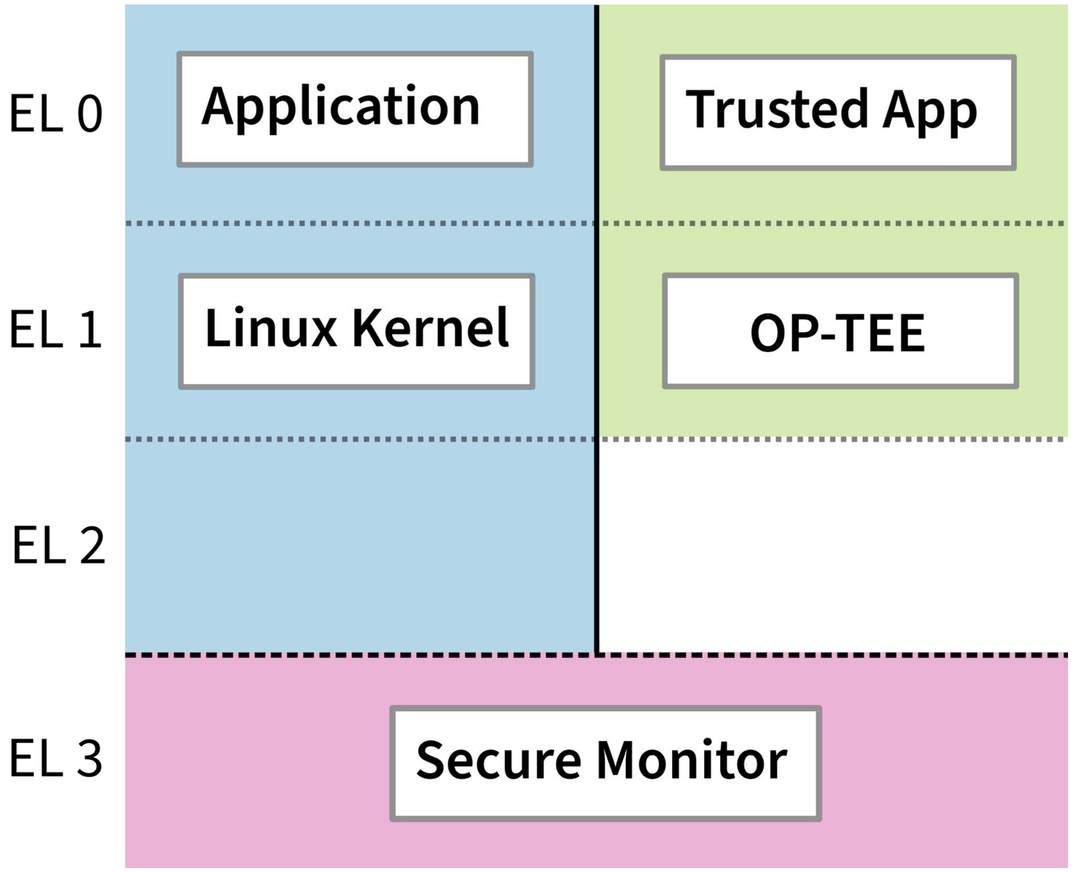
Live Demo

Live Demo

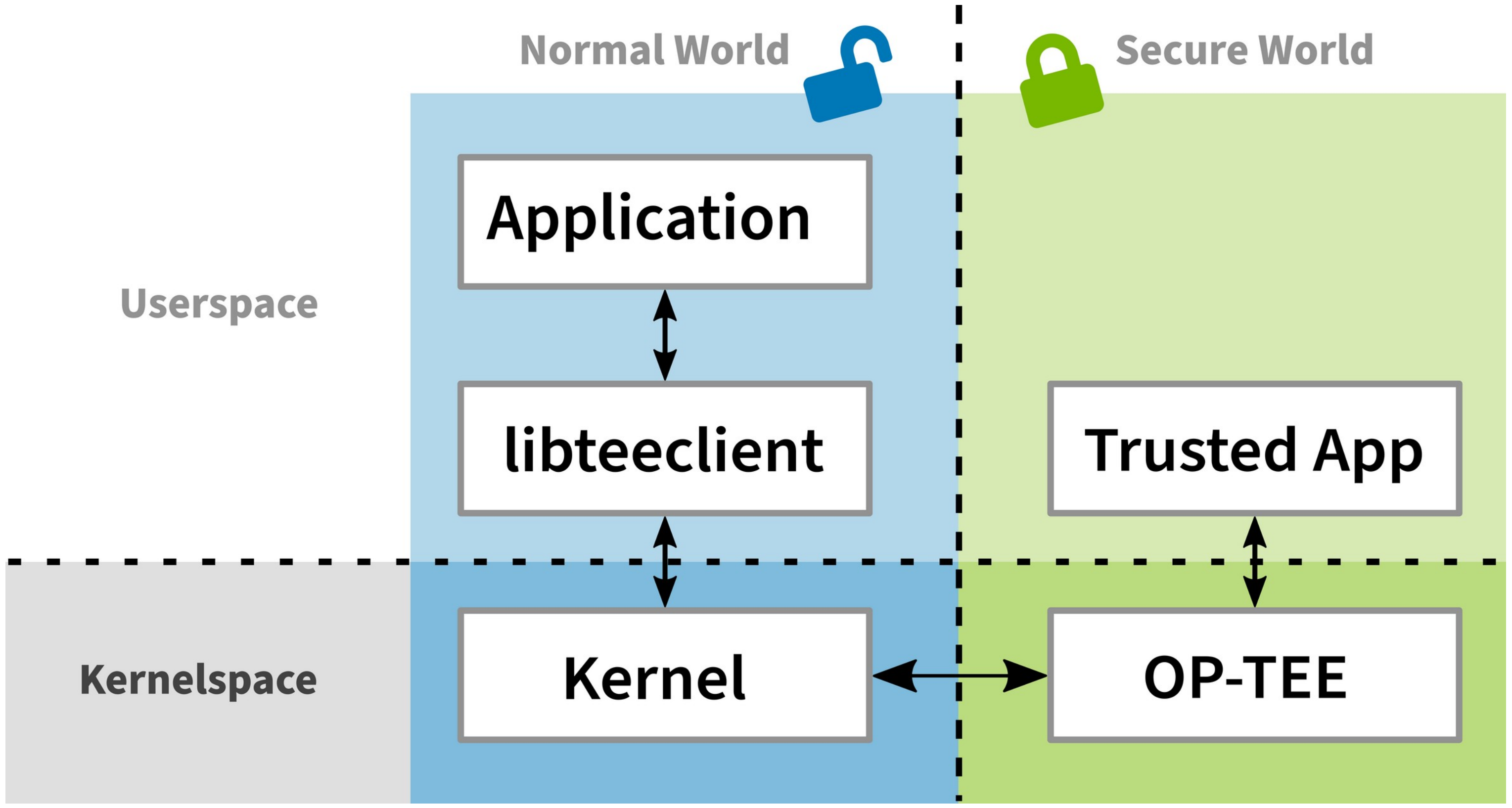
Thanks

Thanks! Questions?

Extra: OP-TEE


EL 0
Application
Trusted App
EL 1
Linux Kernel
OP-TEE
EL 2
EL 3
Secure Monitor

Extra: OP-TEE


Normal World
Secure World
Userspace
Kernelspace
Application
libteeclient
Trusted App
Kernel
OP-TEE