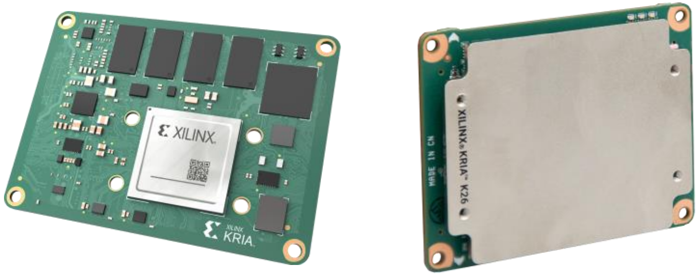
Linaro Connect
MADRID 2024 | MAY 14-17 2024

Linaro Connect
Kria Dynamic Board-ID & Device Tree Selection

Wes Skeffington
Michal Simek

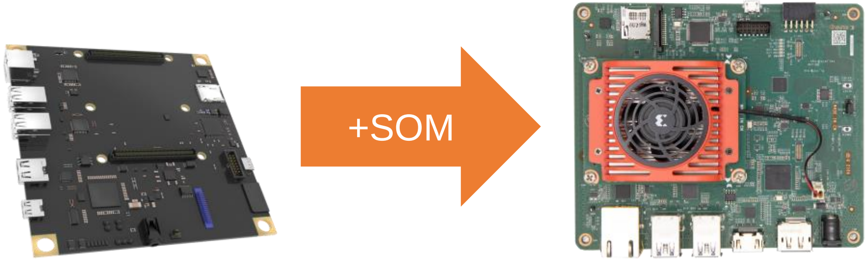
What is a SOM vs. Carrier Card?

System-on-Module (SOM)



- HW platform abstraction for product development
- Plugged to carrier card for physical interfaces
- Kria SOM is small form-factor, flexible, embedded computer based on MPSoC
- Processor, runtime memory, nonvolatile boot memory, core clocks and power supplies
- Heat-spreader for thermal solution interface

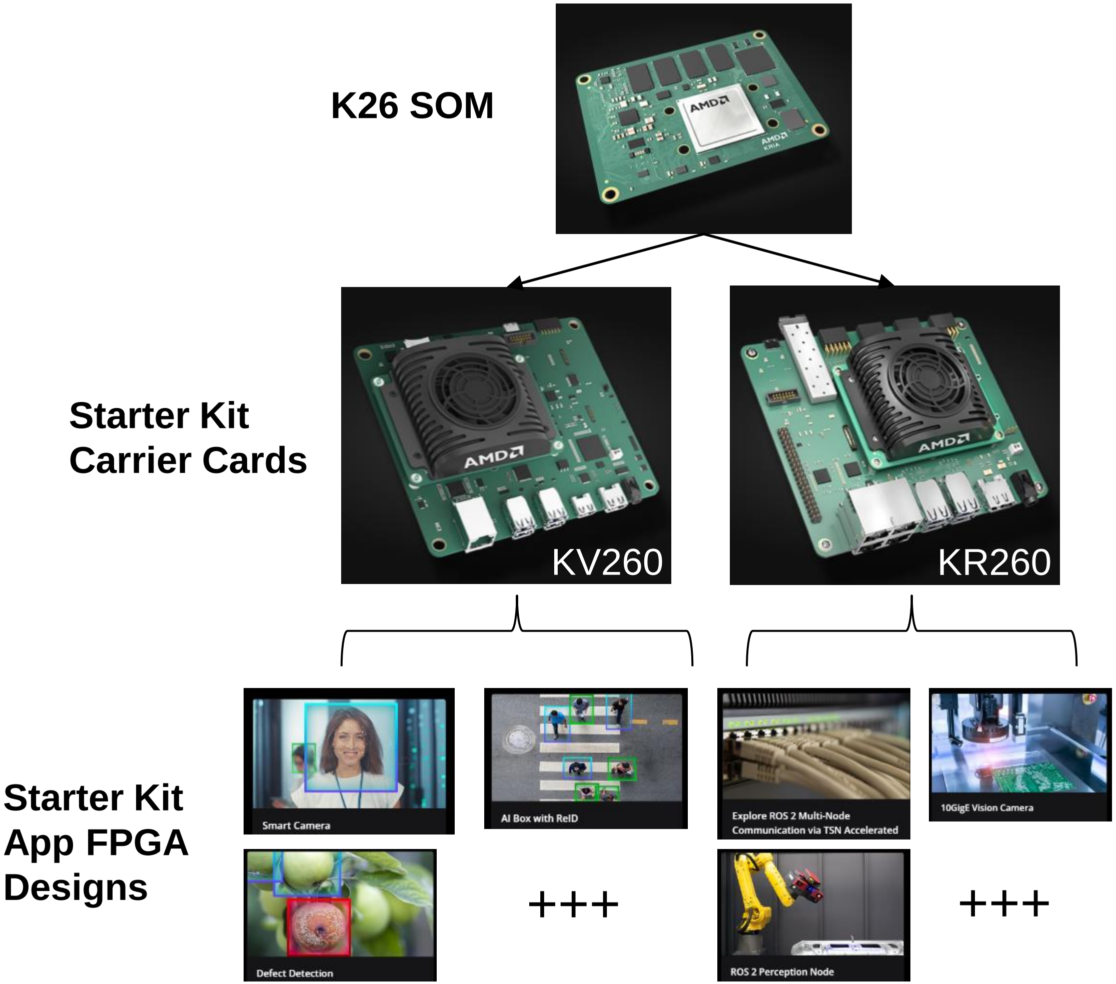
Carrier Card (CC)



- Carrier card implements application specific physical peripherals & any unique HW requirements
- Simplified HW design as high-density digital components abstracted to SOM (e.g. DDR4 layout)
- CC provides a single 5V power input to SOM
- VCCO rails customized by CC design
- Dual 240 pin connectors provide access to CC defined PS & PL interfaces

Problem Statement

- **Goal**: Minimize number of unique FW design artifacts for supporting HW combinations
- **Solution:** Boot time HW identification, peripheral enablement, & dynamic PL loading
- Kria SoC SOMs present two layers of required HW abstraction
  - SOM to multiple Carrier Cards (CC)
  - FPGA programmable hardware



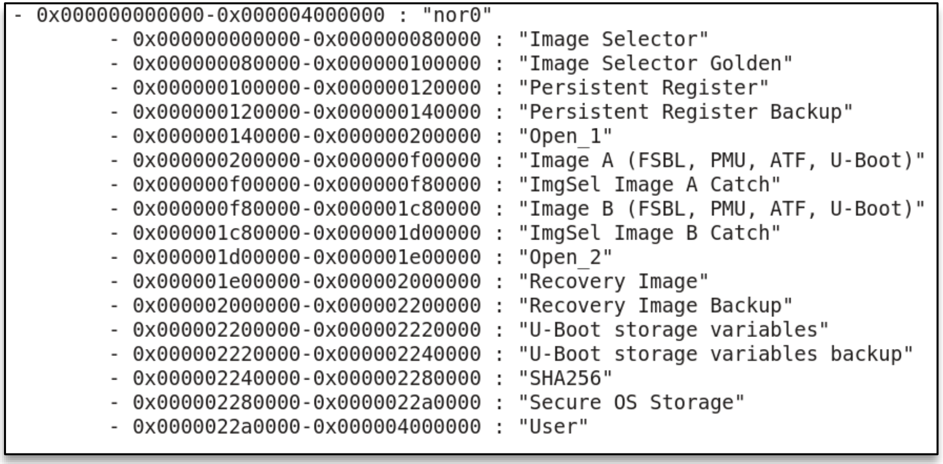
K26 SOM

Starter Kit Carrier Cards

Starter Kit App FPGA Designs

Technical Architecture – HW setup

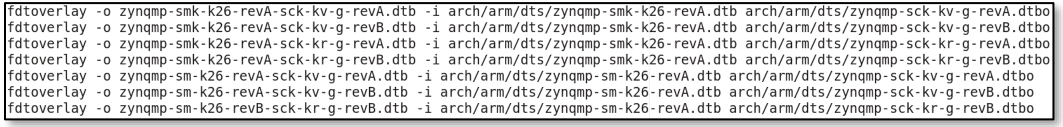
- BOOT.BIN
    - Single image with PMUFW, TF-A, U-Boot, DTBs
- Boot out of QSPI RAW (MTD layout)
- BootROM steps sectors with 32kB offset
    - Image Selector
        - Boot logic based on Persistent Register (A/B)
        - Duplicated ImgSel Image to catch broken Flash sectors
        - Two catch partitions to protect against incorrect boot images
    - Recovery Image
        - Web based application for board recovery
        - Initiated by invalid Boot image via ImgSel Catches
        - Via pressing FWUPD button
- Image A/B
- SOMs and CCs have i2c eeprom for identification
    - In FRU format with using vendor extensions



```
- 0x000000000000-0x000004000000 : "nor0"
    - 0x000000000000-0x000000080000 : "Image Selector"
    - 0x000000080000-0x000000100000 : "Image Selector Golden"
    - 0x000000100000-0x000000120000 : "Persistent Register"
    - 0x000000120000-0x000000140000 : "Persistent Register Backup"
    - 0x000000140000-0x000000200000 : "Open_1"
    - 0x000000200000-0x000000f00000 : "Image A (FSBL, PMU, ATF, U-Boot)"
    - 0x000000f00000-0x000000f80000 : "ImgSel Image A Catch"
    - 0x000000f80000-0x000001c80000 : "Image B (FSBL, PMU, ATF, U-Boot)"
    - 0x000001c80000-0x000001d00000 : "ImgSel Image B Catch"
    - 0x000001d00000-0x000001e00000 : "Open_2"
    - 0x000001e00000-0x000002000000 : "Recovery Image"
    - 0x000002000000-0x000002200000 : "Recovery Image Backup"
    - 0x000002200000-0x000002220000 : "U-Boot storage variables"
    - 0x000002220000-0x000002240000 : "U-Boot storage variables backup"
    - 0x000002240000-0x000002280000 : "SHA256"
    - 0x000002280000-0x0000022a0000 : "Secure OS Storage"
    - 0x0000022a0000-0x000004000000 : "User"
```

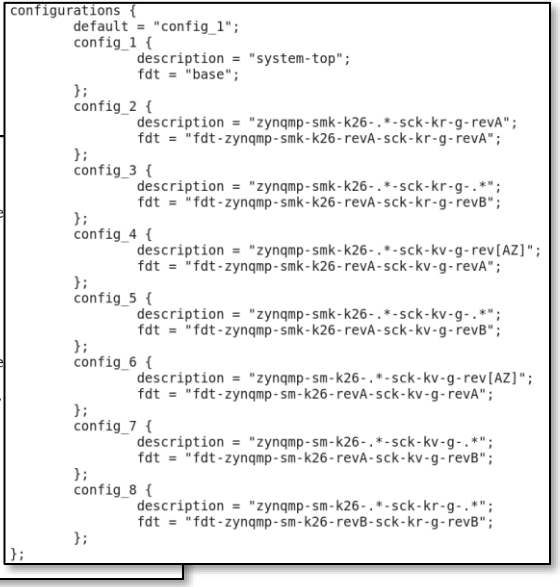
Technical Architecture – U-Boot Image preparation

- Combine SOM + CC DTBs



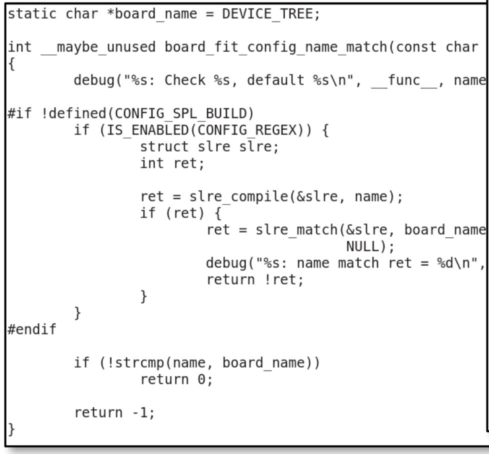
```
fdtoverlay -o zynqmp-smk-k26-revA-sck-kv-g-revA.dtb -i arch/arm/dts/zynqmp-smk-k26-revA.dtb arch/arm/dts/zynqmp-sck-kv-g-revA.dtbo
fdtoverlay -o zynqmp-smk-k26-revA-sck-kv-g-revB.dtb -i arch/arm/dts/zynqmp-smk-k26-revA.dtb arch/arm/dts/zynqmp-sck-kv-g-revB.dtbo
fdtoverlay -o zynqmp-smk-k26-revA-sck-kr-g-revA.dtb -i arch/arm/dts/zynqmp-smk-k26-revA.dtb arch/arm/dts/zynqmp-sck-kr-g-revA.dtbo
fdtoverlay -o zynqmp-smk-k26-revA-sck-kr-g-revB.dtb -i arch/arm/dts/zynqmp-smk-k26-revA.dtb arch/arm/dts/zynqmp-sck-kr-g-revB.dtbo
fdtoverlay -o zynqmp-sm-k26-revA-sck-kv-g-revA.dtb -i arch/arm/dts/zynqmp-sm-k26-revA.dtb arch/arm/dts/zynqmp-sck-kv-g-revA.dtbo
fdtoverlay -o zynqmp-sm-k26-revA-sck-kv-g-revB.dtb -i arch/arm/dts/zynqmp-sm-k26-revA.dtb arch/arm/dts/zynqmp-sck-kv-g-revB.dtbo
fdtoverlay -o zynqmp-sm-k26-revB-sck-kr-g-revB.dtb -i arch/arm/dts/zynqmp-sm-k26-revA.dtb arch/arm/dts/zynqmp-sck-kr-g-revB.dtbo
```

- Create FIT image from DTBs with using regular expressions for configuration selection



```
configurations {
        default = "config_1";
        config_1 {
                description = "system-top";
                fdt = "base";
        };
        config_2 {
                description = "zynqmp-smk-k26-.*-sck-kr-g-revA";
                fdt = "fdt-zynqmp-smk-k26-revA-sck-kr-g-revA";
        };
        config_3 {
                description = "zynqmp-smk-k26-.*-sck-kr-g-.*";
                fdt = "fdt-zynqmp-smk-k26-revA-sck-kr-g-revB";
        };
        config_4 {
                description = "zynqmp-smk-k26-.*-sck-kv-g-rev[AZ]";
                fdt = "fdt-zynqmp-smk-k26-revA-sck-kv-g-revA";
        };
        config_5 {
                description = "zynqmp-smk-k26-.*-sck-kv-g-.*";
                fdt = "fdt-zynqmp-smk-k26-revA-sck-kv-g-revB";
        };
        config_6 {
                description = "zynqmp-sm-k26-.*-sck-kv-g-rev[AZ]";
                fdt = "fdt-zynqmp-sm-k26-revA-sck-kv-g-revA";
        };
        config_7 {
                description = "zynqmp-sm-k26-.*-sck-kv-g-.*";
                fdt = "fdt-zynqmp-sm-k26-revA-sck-kv-g-revB";
        };
        config_8 {
                description = "zynqmp-sm-k26-.*-sck-kr-g-.*";
                fdt = "fdt-zynqmp-sm-k26-revB-sck-kr-g-revB";
        };
};
```

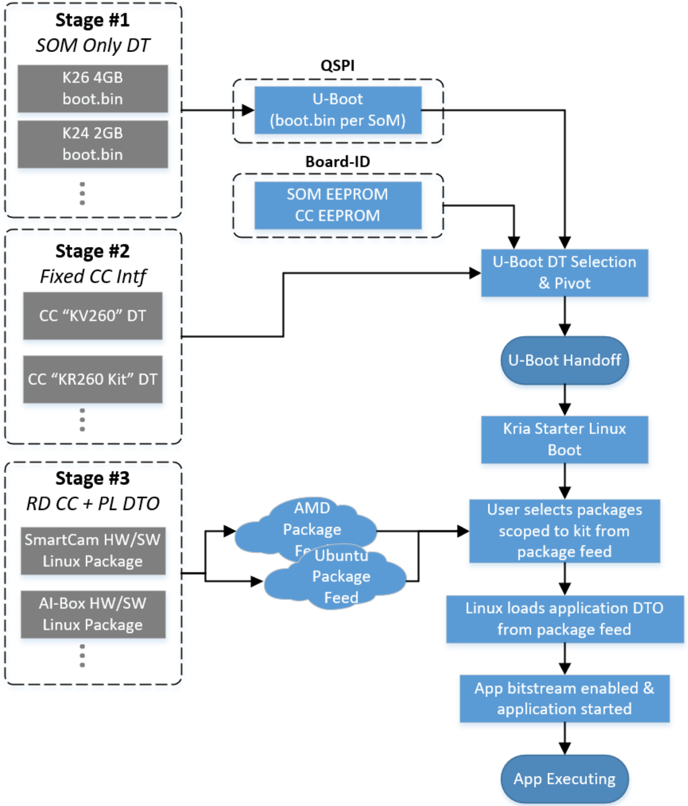
- Boot SOM with minimal configuration
  - Do SOM and Carrier Card detection (based on i2c eeprom)
  - And do DTB RESELECTION in U-Boot



```
static char *board_name = DEVICE_TREE;

int __maybe_unused board_fit_config_name_match(const char
{
        debug("%s: Check %s, default %s\n", __func__, name

#if !defined(CONFIG_SPL_BUILD)
        if (IS_ENABLED(CONFIG_REGEX)) {
                struct slre slre;
                int ret;

                ret = slre_compile(&slre, name);
                if (ret) {
                        ret = slre_match(&slre, board_name
                                         NULL);
                        debug("%s: name match ret = %d\n",
                        return !ret;
                }
        }
#endif

        if (!strcmp(name, board_name))
                return 0;

        return -1;
}
```

Technical Architecture – Peripheral Enable, DT Selection, DT Overlays

- HW/SW boundaries split into three incremental domains:
1. Initial boot DT for U-Boot (SOM only)
2. CC DT swap (SOM + CC)
3. FPGA DT overlay (dynamic load/unload)
- Key SW components
  - U-Boot + PMU config object loading
  - Decoupled Linux dynamic DT selection & loading
  - Linux *fpga-mgr* + AMD *libdfx*
  - Linux DT overlays
- Challenges
  - PL peripheral driver clean load/unload from active DT
  - Upstream support of DT overlays
  - Limited to references to customers for production deployments



Stage #1
SOM Only DT
K26 4GB boot.bin
K24 2GB boot.bin
⋮

Stage #2
Fixed CC Intf
CC "KV260" DT
CC "KR260 Kit" DT
⋮

Stage #3
RD CC + PL DTO
SmartCam HW/SW Linux Package
AI-Box HW/SW Linux Package
⋮

QSPI
U-Boot (boot.bin per SoM)

Board-ID
SOM EEPROM
CC EEPROM

AMD Package Fe
Ubuntu Package Feed

U-Boot DT Selection & Pivot
U-Boot Handoff
Kria Starter Linux Boot
User selects packages scoped to kit from package feed
Linux loads application DTO from package feed
App bitstream enabled & application started
App Executing

Example Applications

- SOM has high-adoption in areas with high application I/O diversity within a given customer
  - Industrial I/O
  - Industrial controllers
  - Healthcare
  - Vision systems
- Dynamic platform capabilities allow for reuse not just in HW design but also in shared FW and SW infrastructure components.
  - Reduced design artifact development & maintenance
  - Reduced life-cycle costs through shared/common updates
  - Faster time to market through abstracted HW and SW coupling

Industrial I/O

Industrial controllers

Vision systems

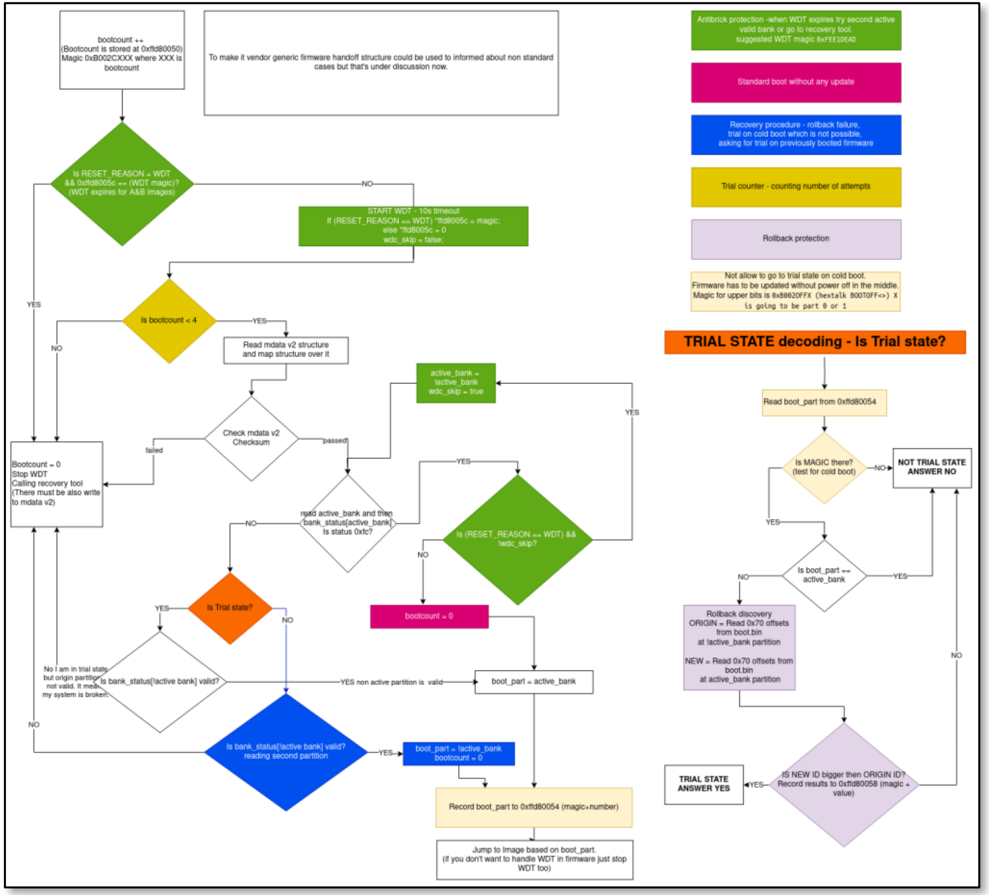
Next Steps

- Collaboration with Linux community for DT overlays in upstream
- Canonical Ubuntu:
  - libdfx + dfx-mgr inclusion in Ubuntu archive
  - AMD FPGA SoC stable (PS & PL) driver integration in Ubuntu LTS
  - DKMS based integration for customer PL & new FPGA IP drivers
- Linaro collaborations:
  - fpga-mgr, libdfx, dfx-mgr inclusion in Linaro TRS
  - Dynamic FPGA load/unload testing inclusion in LAVA
  - SystemReady-IR (LVFS, fwupd integration)
  - DTB FIT generation via binman (but without placing binman node to DT)
  - A/B update based on mdata v2 (in progress)

Thank you

A/B Update – mdata v2

- Reusing persistent register MTD
- ImgSel changes - adopt mdata v2
- ImgSel catch changes
- Image Recovery changes
- Roll back protection with WDT
- Using mdata protection for SOC extension



bootcount ++
(Bootcount is stored at 0xffd80050)
Magic 0xB002CXXX where XXX is bootcount

To make it vendor generic firmware handoff structure could be used to informed about non standard cases but that's under discussion now.

Is RESET_REASON = WDT && 0xffd8005c == (WDT magic)? (WDT expires for A&B images)

START WDT - 10s timeout
If (RESET_REASON == WDT) *ffd8005c = magic;
else *ffd8005c = 0
wdc_skip = false;

Is bootcount < 4

Read mdata v2 structure and map structure over it

Check mdata v2 Checksum

Bootcount = 0
Stop WDT
Calling recovery tool
(There must be also write to mdata v2)

active_bank = !active_bank
wdc_skip = true

read active_bank and then bank_status[active_bank]
Is status 0xfc?

Is (RESET_REASON == WDT) && !wdc_skip?

bootcount = 0

Is Trial state?

Is bank_status[!active bank] valid?

No I am in trial state but origin partition not valid. It means my system is broken.

boot_part = active_bank

Is bank_status[!active bank] valid?
reading second partition

boot_part = !active_bank
bootcount = 0

Record boot_part to 0xffd80054 (magic+number)

Jump to Image based on boot_part.
(if you don't want to handle WDT in firmware just stop WDT too)

Antibrick protection -when WDT expires try second active valid bank or go to recovery tool.
suggested WDT magic `0xFEE1DEAD`

Standard boot without any update

Recovery procedure - rollback failure,
trial on cold boot which is not possible,
asking for trial on previously booted firmware

Trial counter - counting number of attempts

Rollback protection

Not allow to go to trial state on cold boot.
Firmware has to be updated without power off in the middle.
Magic for upper bits is `0xB002OFFX (hextalk BOOTOFF<>) X is going to be part 0 or 1`

TRIAL STATE decoding - Is Trial state?

Read boot_part from 0xffd80054

Is MAGIC there?
(test for cold boot)

NOT TRIAL STATE ANSWER NO

Is boot_part == active_bank

Rollback discovery
ORIGIN = Read 0x70 offsets from boot.bin
at !active_bank partition

NEW = Read 0x70 offsets from boot.bin
at active_bank partition

IS NEW ID bigger then ORIGIN ID?
Record results to 0xffd80058 (magic + value)

TRIAL STATE ANSWER YES