


EMBEDDED SYSTEMS REVERSE ENGINEERING

The background of the slide is a dark blue circuit board with glowing cyan and purple lines representing traces. A central microcontroller chip is highlighted with a magnifying glass. Inside the magnifying glass, there is a grid of binary code (0s and 1s).

101010100
1011010110
0101010010
1011010101
11010010

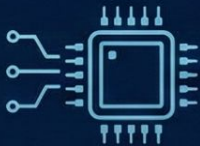
Week 3

Deep Dive into the RP2350 Boot Architecture

- Tracing the complete boot sequence (Power-on to main())
 - Understanding the Bootrom & Vector Table
 - XIP (Execute In Place) from Flash
 - Analyzing startup code (crt0.S) with GDB & Ghidra
 - Thumb Mode addressing quirks
- 
- A small white four-pointed star icon is located in the bottom right corner of the slide.

The “Before main()” Sequence Overview

1. Power On: Cortex-M33 core wakes up, starts at 0x00000000.



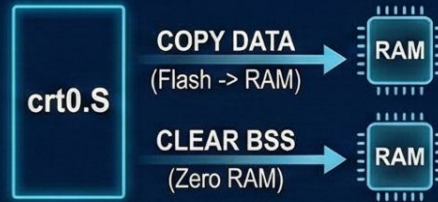
0x00000000

1. Power Onu: Cortex-M33, 0x0000000.

3. Boot Stage 2 (boot2): Configures flash for fast XIP access.



5. C Runtime Startup (crt0.S): Prepares RAM (Data copy, BSS clear).



2. Bootrom Executes: On-chip immutable ROM verifies flash firmware.



BOOTROM
(Immutable)

SP
(Stack Pointer)

**Reset
Handler**

4. Vector Table Read: CPU find the Stack Pointer and Reset Handler addresses.



**VECTOR
TABLE**
(at 0x10000000)

SP
(Stack Pointer)

**Reset
Handler**

6. Platform Entry: Initializes SDK and finally calls main().

**Platform
Entry**

**SDK
Init**

main()

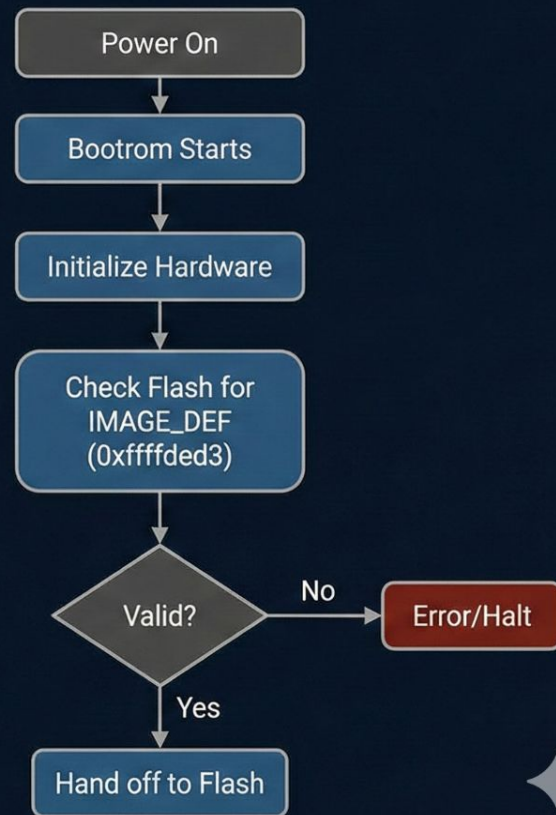
Stage 1: The Bootrom (The Trust Anchor)

The immutable foundation of the RP2350.



Security Note:

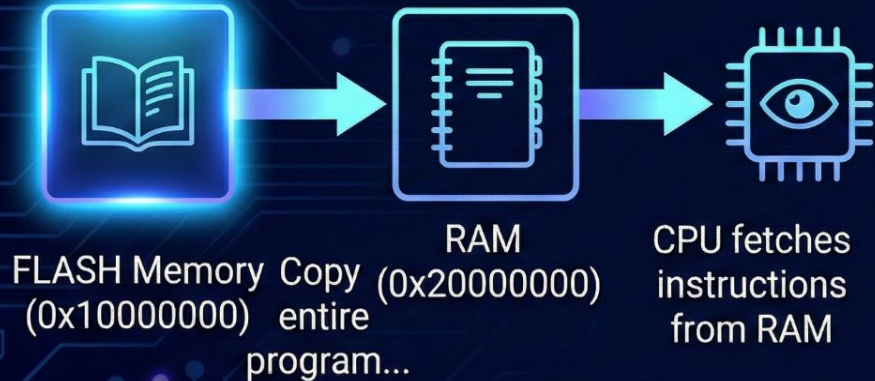
Because it cannot be changed, it is the root of trust for Secure Boot chains.



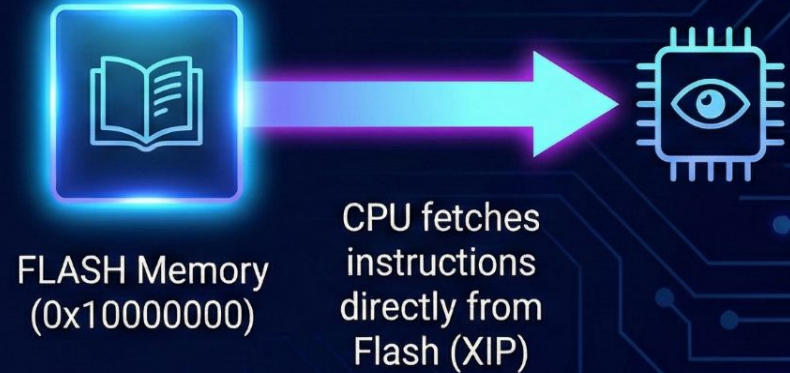
Understanding XIP (Execute In Place)

How the CPU runs code without RAM.

Non-XIP: Copying to RAM



XIP: Reading directly from Flash



Benefit: Saves precious SRAM for volatile data.

The Vector Table (The CPU's Roadmap)

Located at the start of XIP Flash (0x10000000).

It tells the CPU where crucial things are located immediately after bootrom finishes.




A diagram on the left shows a box labeled "XIP Flash (0x10000000)" with an arrow pointing to the first row of the Vector Table.




Offset	Address Content	Description
0x00	0x20082000	 Initial Stack Pointer (SP). Sets top of the stack in RAM.
0x04	0x1000015d	 Reset Handler. The entry point for the startup code.
0x08+	...	 Exception Handlers (NMI, HardFault, etc.).

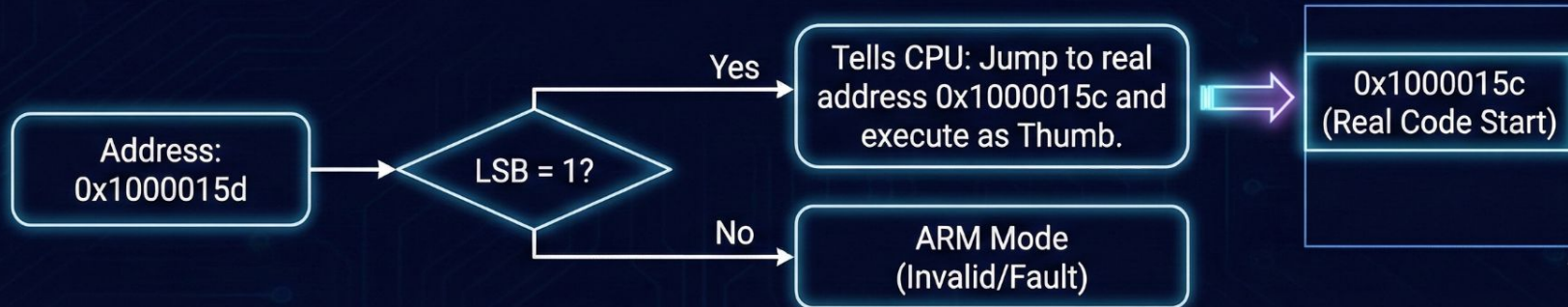
Critical Concept: Thumb Mode Addressing

Why do addresses end in odd numbers?

- ARM Cortex-M architecture executes in Thumb mode. 
- The Least Significant Bit (LSB) of an address indicates the instruction set.

1 1 1 0 0 1 1 | 0 1 1 1 1 1 1

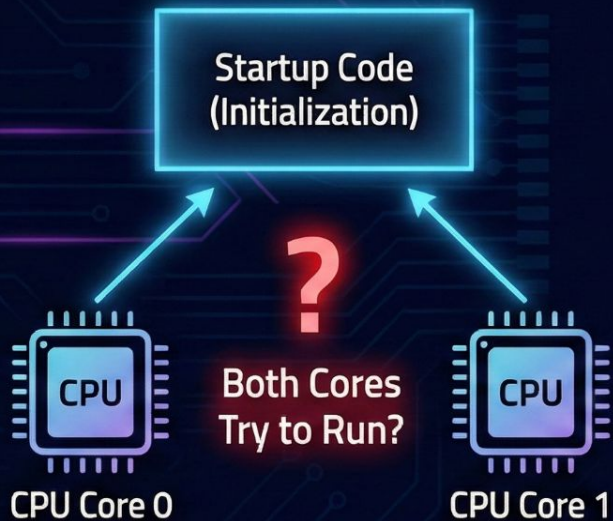
- LSB = 1 (Odd): Thumb Mode (Required for RP2350).  
- LSB = 0 (Even): ARM Mode (Not supported here). 
- Example: Reset Handler address 0x1000015d



The Reset Handler: Phase 1 (Core Check)

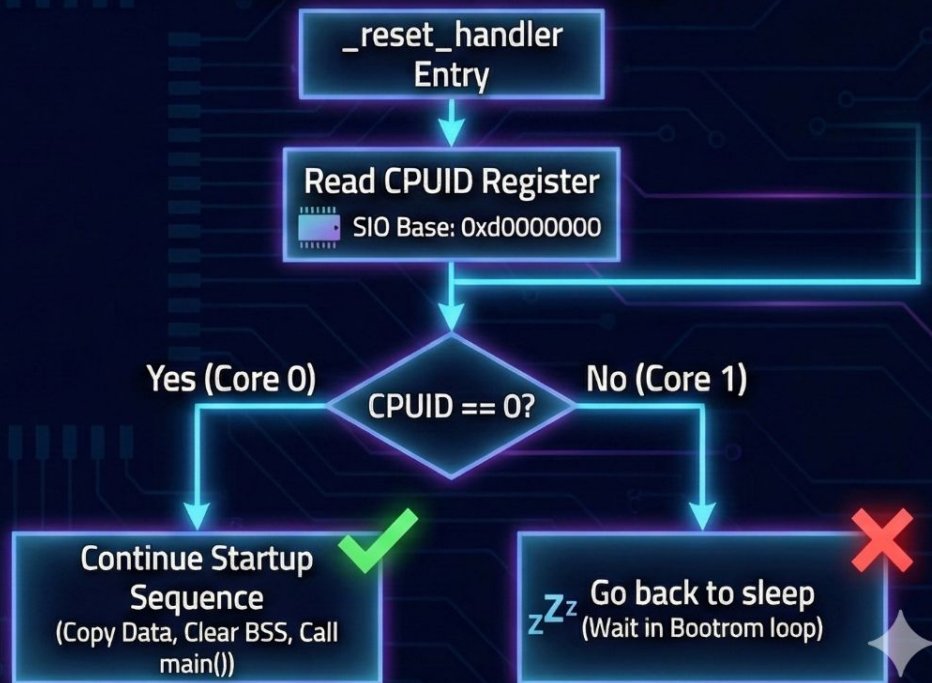
The first code that runs from your flash binary (`_reset_handler`).

The Problem:



The RP2350 has two cores, but only one should run startup initialization.

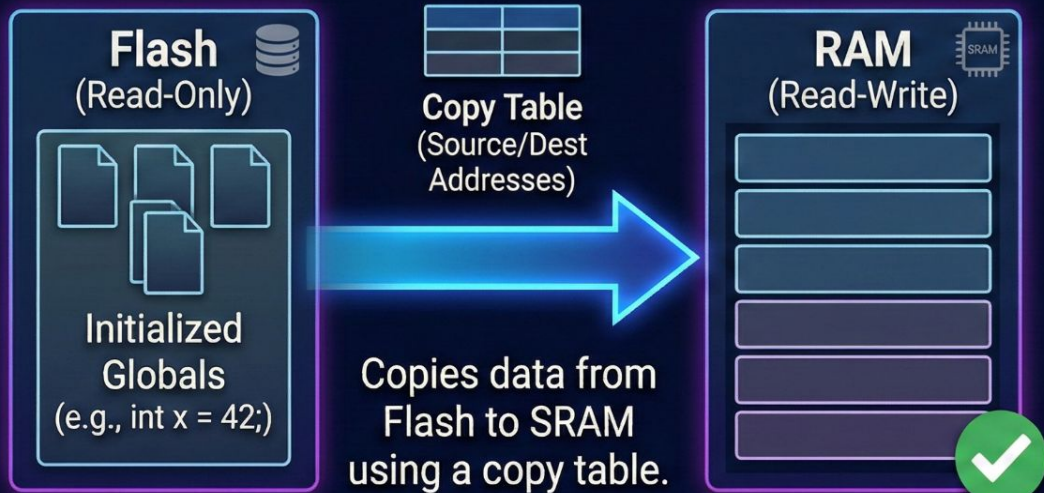
The Check (Solution):



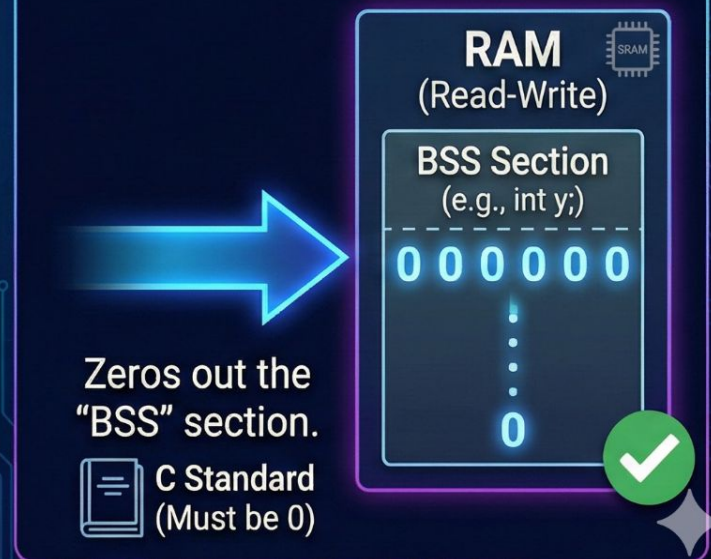
The Reset Handler: Phases 2 & 3 (Memory Prep)

Setting up the RAM environment defined in the linker script.
Since Flash is Read-Only (XIP), volatile data must be moved to RAM.

Phase 2: Data Copy



Phase 3: BSS Clear



The Final Stretch: Platform Entry

The bridge between startup assembly and C code.



Execution moves to **platform_entry** after memory is prepped.

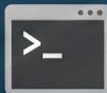


Tools of the Trade: Static vs. Dynamic

We use different tools for different perspectives on the boot process.



GDB (Dynamic Analysis)



Action:

- Stepping through live execution (si)
- Examining registers (i r)
- Watching memory change

```
> |
(gdb) si
(gdb) i r
r0      0x20000000
sp      0x20082000
```

Best for:

Seeing exactly what value is in the **Stack Pointer** or CPUID register at a specific moment.

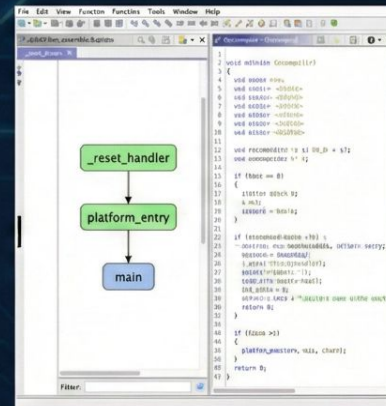


Ghidra (Static Analysis)



Action:

- Decompiling assembly into C-like pseudocode
- Viewing function call graphs



Best for:

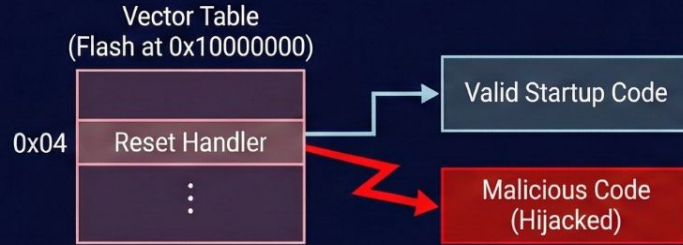
Visualizing the entire flow from **_reset_handler** down to **main** without needing hardware connected.

Security Implications of the Boot Sequence

Understanding boot is vital for both attack and defense.

Attack Vectors:

Vector Table Hijacking



Modifying address 0x04 to point to malicious startup code.

Glitching



Interrupting the Bootrom's signature check to bypass validation.



Defense Strategies (Secure Boot):

Immutable Bootrom



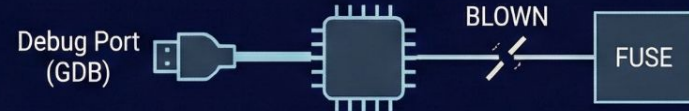
The hardware root of trust cannot be modified.

Signature Verification



Bootrom checks cryptographically signed IMAGE_DEF before executing.

Disable Debug



Blowing fuses to prevent GDB attachment in production.

