Excellent. Now we'll **do**, not just talk.
I'll simulate a **realistic embedded LRU** exactly the way I would train a junior on the job and **walk you step-by-step from zero knowledge to system-level mastery**.

Think of this as **day-1 to day-30 system induction on a real defense/aerospace program**.

🛩️ CASE STUDY

Mission Computer LRU (Avionics-Grade Embedded Unit)

Application (given to junior on Day-1)

> "This LRU is a **Mission Computer** used in an aircraft.
> It collects sensor data, communicates with other avionics systems, performs processing, and sends commands to actuators."

You are NOT given schematics initially.

📦 LRU OVERVIEW (What the junior is told)

Hardware Summary

- **Chassis**: 3U VPX conduction-cooled
- **Slots**: 6
- **Backplane**: VPX switched fabric
- **Power Input**: 28V DC aircraft supply

Cards in the LRU

Slot	Card Type
Slot 1	Power Supply Card
Slot 2	SBC (PowerPC-based)
Slot 3	FPGA I/O Card
Slot 4	Ethernet Switch Card
Slot 5	ARINC-429 I/O Card

Slot	Card Type
Slot 6	Maintenance & BIT Card

🧠 STEP 1 – SYSTEM THINKING (Before Electronics)

First thing I ask the junior:

❓ *"What is the LRU supposed to DO?"*

Junior must answer:

- Receive sensor data (ARINC-429)
- Communicate with other LRUs (Ethernet)
- Process mission algorithms (SBC)
- Output control commands
- Monitor its own health

📌 **Key lesson:**

> Never open schematics until you understand **why the system exists**.

🧱 STEP 2 – PHYSICAL & BACKPLANE ANALYSIS

I show the physical chassis

Questions I ask:

- Why conduction-cooled?
  👉 No airflow allowed in aircraft bay
- Why VPX?
  👉 High-speed serial + rugged + modular

Backplane understanding:

- **Data plane**: PCIe / Ethernet
- **Control plane**: Ethernet
- **Utility plane**: Power, clock, reset

🧠 **System insight:**

Backplane is the **nervous system**, not just wiring.

⚡ STEP 3 – POWER ARCHITECTURE WALKTHROUGH

Given info:

- Input: 28V DC
- Redundant feeds: A & B

Power card responsibilities:

- EMI filtering
- Surge protection
- DC-DC conversion
- Power sequencing

I make the junior draw:

```
28V IN
 ├─> 12V (Backplane)
 │      ├─> SBC
 │      ├─> FPGA Card
 │      └─> Switch
 ├─> 5V
 └─> Standby 3.3V (BIT card)
```

📌 **Real-world check:**

> “Which card powers up first and why?”

Correct answer:

➡️ BIT card (to monitor power-up health)

⏱️ STEP 4 – RESET, CLOCK & BOOT SEQUENCE

Reset philosophy:

- Power card asserts GLOBAL_RESET
- SBC releases reset after power-good
- FPGA cards wait for SBC config

Boot flow:

```
Power ON
 ↓
```

```
Power rails stable
 ↓
Global Reset released
 ↓
SBC boots from Flash
 ↓
FPGA bitstreams loaded
 ↓
I/O cards enabled
```

🧠 **Debug skill:**

If system hangs → identify **where the boot sequence stops**.

🧠 STEP 5 – PROCESSING & CONTROL FLOW

SBC details:

- PowerPC processor
- DDR3 memory
- NOR Flash for bootloader
- RTOS (VxWorks)

Responsibilities:

- System master
- Health management
- Data processing
- FPGA configuration

FPGA I/O Card:

- ARINC decoding
- Timing critical tasks
- DMA data to SBC via PCIe

📌 **Key lesson:**

CPU does **decisions**, FPGA does **deterministic work**.

🔌 STEP 6 – DATA FLOW SIMULATION

Example data path (Sensor input):

```
Sensor
 → ARINC-429 I/O Card
 → FPGA
 → PCIe over VPX
 → SBC DDR
 → Mission Software
 → Ethernet Switch
 → Other LRU
```

Junior exercise:

Ask them to identify:

- Electrical standard
- Protocol layer
- Buffering points
- Latency sources

🧠 **This builds integration confidence.**

🌡️ STEP 7 – HEALTH MONITORING & FAULT HANDLING

Monitoring points:

- Voltage rails
- Card temperatures
- Watchdog timers
- Communication timeouts

Fault example:

ARINC card overheats

System reaction:

1. BIT card detects temp fault
2. SBC logs fault
3. ARINC channel disabled
4. Fault reported via Ethernet
5. System continues in degraded mode

📌 **Aerospace mindset:**

> Never just “fail” — always **fail safe**.

🧪 STEP 8 – DEBUGGING A REAL FAILURE (Simulation)

Reported issue:

> “LRU powers on, but no Ethernet communication.”

How I guide the junior:

1. Check power rails on switch card
2. Verify reset released
3. Check clock presence
4. Verify SBC config of switch
5. Check backplane lanes

🧠 **Skill learned:**

Debug **systematically**, not randomly.

🔁 STEP 9 – MAKE IT TRANSFERABLE

Final task I give the junior:

> “You are given a **new LRU** tomorrow.
> You have 1 week.
> Prepare a system understanding presentation.”

They must cover:

- System purpose
- Card roles
- Power & boot flow
- Data flow
- Failure handling

If they can do this → **they are no longer a fresher**.

🎓 What You Just Learned (Meta)

You didn’t learn *this* LRU.
You learned **HOW TO THINK ABOUT ANY LRU**.

This is exactly how:

- Defense PSUs
- Avionics mission computers
- Radar processors
- Industrial controllers
  are understood in real programs.

If you want next, I can:

- Do a **fault-injection case study**
- Simulate **first power-up day on lab bench**
- Show **what seniors silently test in juniors**
- Map this to **interview questions & job tasks**

Just tell me the next scenario you want to simulate 🔧 🧠