Below is **exactly how I would train a fresh graduate** to *master* any **embedded LRU technical specification document**, not just read it.
This is a **job-ready, engineer's method**, used in avionics/defense programs where mistakes are expensive.

I'll structure this as a **mentoring playbook**, moving from **basic → advanced → real-world application**.

---

0. First Rule I Teach the Junior (Mindset Reset)

📌 **A technical specification is NOT a document to read.**
It is a document to "interrogate".

Your job as an embedded hardware engineer is to:

- Extract **design intent**
- Predict **design challenges**
- Identify **risks**
- Convert text into **hardware decisions**

I tell the junior:

> "If you can explain *why* each requirement exists and *what it forces us to design*, you've mastered the spec."

---

1. How I Teach Them to Read the Spec (Very Important)

❌ Wrong way (Fresh Graduate instinct)

- Start from page 1
- Read line by line
- Try to "understand everything"

✅ Correct Engineer Way (I train them this)

We read the spec in **multiple passes**, each with a **different goal**.

Pass	Goal	Outcome
Pass 1	System understanding	What is this box?

Pass	Goal	Outcome
Pass 2	Functional mapping	What must it do?
Pass 3	Interface ownership	What signals go where?
Pass 4	Design constraints	What are we allowed / not allowed to do?
Pass 5	Verification mindset	How will this be tested & certified?

---

2. PASS-1: System-Level Understanding (Big Picture First)

What I make the junior do

Before touching any section deeply, I ask them to answer **ONLY these questions**:

A. What is this LRU?

- Is it:
  - Mission computer?
  - Display processor?
  - Video processing unit?
  - Data concentrator?

B. Where does it sit?

- Aircraft?
- Ground system?
- Naval platform?

C. What does it interact with?

- Sensors?
- Displays?
- Other mission computers?
- Video sources?

📌 **Exercise I give**

👉 "Draw a **block diagram** of the system using only the *intro* and *functional overview* sections."

No schematics. Just boxes and arrows.

If they can't draw this → they don't understand the unit yet.

---

3. PASS-2: Functional Requirements (Core of the Spec)

How I teach them to decode Functional Requirements

I tell them:

> "Every functional requirement will become either hardware, firmware, or software."

I make them classify each requirement into:

Requirement Type	Example
Hardware-driven	Redundancy, I/O count
Firmware-driven	Initialization, monitoring
Software-driven	Graphics rendering
Cross-domain	Power-up sequencing

Example (How I walk them through)

> "The unit shall process and display 4 independent video streams at 1080p"

I ask:

- What bandwidth is required?
- What graphics processor is needed?
- What memory size?
- What thermal impact?

📌 **Rule I teach**

Every "shall" → ask **HOW will we implement this physically?**

---

4. PASS-3: Interfacing Details with Avionics Subsystems

This is where juniors usually panic. I simplify it.

I teach them to break interfaces into 4 layers

1️⃣ Electrical Layer

- Voltage levels
- Differential / single-ended
- Impedance, termination

2️⃣ Protocol Layer

- ARINC 429 / 664
- MIL-STD-1553
- Ethernet / PCIe / UART

3️⃣ Data Layer

- Message format
- Video format
- Frame rate

4️⃣ Responsibility Layer

- Are we master or slave?
- TX only / RX only?
- Redundancy handling?

📌 **Exercise**

I ask them to create an **Interface Control Table**:

Interface	Direction	Protocol	Speed	Owner

If they can do this → they are thinking like a system engineer.

---

5. PASS-4: Performance, Design, Test, Manufacture Requirements

This section teaches **engineering discipline**.

What I highlight to the junior

Performance

- Throughput
- Latency
- Frame rates
- Boot time

Ask:

> “What will break first if this is violated?”

Design Constraints

- Power limits
- Thermal limits
- Form factor
- Redundancy

I explain:

> “This section silently kills bad design ideas.”

Manufacturing & Acceptance

- Environmental stress screening
- Burn-in
- Production tolerances

📌 **Real-world lesson**

A design that works in lab but fails here is a *bad design*.

---

6. Qualification Requirements (Very Critical in Avionics)

This is where juniors usually skim — I **don’t allow that**.

I teach them to ask:

- DO-160 category?
- Temperature range?
- Vibration / shock?
- EMI/EMC?

📌 **Key lesson**

Qualification requirements:

- Decide **component selection**
- Decide **PCB stackup**
- Decide **connector choice**

> “You don’t design freely in avionics. You design inside a cage.”

---

7. Embedded Graphics Computing Requirements

This is a **specialized section**, so I guide carefully.

I ask them to extract:

- GPU vs FPGA vs CPU?
- Supported APIs (OpenGL, Vulkan)
- Video resolution & scaling
- Memory bandwidth

📌 **Golden Rule**

Graphics requirements = **power + thermal + memory problem**, not just software.

---

8. Video Interface & Communication Video Interfaces

I teach them to separate:

Video Interface

- HDMI / DVI / DisplayPort
- LVDS / SDI

Communication Video Interface

- Ethernet-based video
- ARINC 818
- Compressed vs raw

📌 **Key engineering question**

> “Where does timing control live — source or sink?”

This determines:

- Clocking
- Jitter tolerance
- PLL design

---

9. Packaging Specification (Often Ignored, Very Important)

I tell them:

> "This decides whether your design survives the real world."

They must note:

- Form factor (VME, VPX, custom)
- Cooling (conduction / air)
- Connector mating cycles
- Maintenance access

📌 **Exercise**

Ask:

> "Can I physically route all required signals inside this package?"

---

10. FINAL PASS: Convert Spec → Design Checklist

At the end, I make the junior produce **one master document**:

Embedded LRU Design Readiness Checklist

- System block diagram
- Interface summary
- Key constraints
- High-risk requirements
- Qualification impacts
- Open questions / ambiguities

If they can do this **without my help**, they are ready for **any embedded unit spec**.

---

11. How This Helps Them in Their Career

After this training, the junior can:

✅ Read specs confidently
✅ Talk to system engineers
✅ Ask intelligent questions
✅ Avoid design rework
✅ Transition from "reader" to "designer"

If you want, next I can:

- Walk through a **real LRU spec (simulated)** step by step
- Teach **how to ask questions during design reviews**
- Convert a spec into **schematic-level decisions**
- Train how **interviewers expect you to talk about specs**

Just tell me where you want to go next.