

Perfect. Now we cross the line from “understanding requirements” into **real engineering responsibility**.

What you’ll see now is exactly how Phase-2 happens on an avionics Mission Computer program. This is where **most freshers collapse**, and where seniors decide whether you are an *engineer* or just a *support resource*.

🔴 PHASE-2 — ARCHITECTURE DEFINITION & TECHNOLOGY TRADE-OFFS

(Applied to an Avionics Mission Computer)

If Phase-1 answered **WHAT** must be guaranteed,

👉 Phase-2 answers **HOW** those guarantees can be met without violating safety, timing, or certification.

1 What Phase-2 REALLY Is

✗ Fresher misunderstanding

“Phase-2 is selecting CPU, bus, and cards.”

✓ Reality

Phase-2 is deciding architecture *philosophy* before touching components.

Once Phase-2 is frozen:

- 70% of design risk is locked
 - Certification cost is largely fixed
 - Recovery from bad choices is painful
-

2 Phase-2 INPUTS (From Phase-1)

You are NOT allowed to start Phase-2 without:

- ✓ Functional decomposition
- ✓ Functional flow
- ✓ Requirement classification
- ✓ Derived requirements
- ✓ Ambiguities list

If any of these are missing → Phase-2 is invalid.

3 Mission Computer Architecture QUESTIONS (Answered in Phase-2)

Before drawing anything, you MUST answer:

1. Single MC or redundant MC?
2. Centralized or distributed processing?
3. Deterministic or best-effort timing?
4. Hardware fault tolerance or software mitigation?
5. Partitioned or monolithic software?
6. Growth margin philosophy?

These are **architecture decisions**, not component choices.

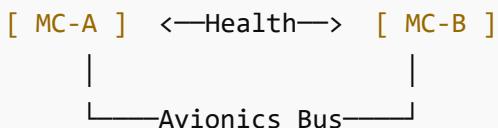
4 High-Level Architecture OPTIONS (Mission Computer)

4.1 Single-Channel Mission Computer (Risky)

[CPU + Memory + I/O]

- Lower cost
- Single point of failure
- Rarely acceptable in avionics MCs

4.2 Dual-Channel (Active / Standby)



- Fault tolerance
- Controlled failover
- Complexity in synchronization

👉 Most military MCs live here

4.3 Federated vs Integrated

Model	Meaning
Federated	Separate boxes per function
Integrated	One MC handles many functions

Modern MCs → **Integrated Modular Avionics (IMA)** style thinking.

5 Functional-to-Architectural Mapping

Now we map Phase-1 functions → architectural blocks.

5.1 Architectural Blocks (Abstract)

Block	Responsibility
Compute Complex	Mission processing
I/O Complex	Avionics interfaces
Memory Subsystem	Data & code storage
Health Monitor	BIT, watchdogs
Power & Reset	Safe startup & recovery
Timing	Deterministic execution

⚠ Still no part numbers.

6 Timing Architecture (Most Dangerous Area)

Mission Computer ≠ IT computer.

Architecture must define:

- Hard real-time tasks
- Soft real-time tasks
- Background tasks

Example:

Task	Timing
Sensor processing	Hard RT
Data fusion	Hard RT
BIT logging	Soft RT
Maintenance	Background

🔥 Phase-2 failure here → missed deadlines in flight

7 Redundancy & Fault-Tolerance Philosophy

This is **non-negotiable** for MCs.

7.1 Fault Classes

Fault	Strategy
CPU hang	Watchdog reset
Memory error	ECC + retry
Interface failure	Switch path
Software crash	Partition restart

You must decide:

- Detect?
 - Isolate?
 - Recover?
 - Degrade?
-

8 Partitioning Strategy (Hardware vs Software)

8.1 Hardware Partitioning

- Separate buses
- Isolated power domains

8.2 Software Partitioning

- RTOS partitions
- Time & space isolation

🔥 MCs almost always require **software partitioning**, even if not explicitly stated.

9 Interface Architecture (Not Signal Level Yet)

You define:

- Which interfaces are **critical**
- Which are **non-critical**
- Which need **isolation**

Example:

Interface	Criticality
Sensor input	Safety-critical
Display output	Mission-critical
Maintenance port	Non-critical
This drives isolation, buffering, redundancy later.	

10 Growth & Margin Philosophy (Senior-Level Thinking)

You must answer:

- CPU margin (%)?
- Memory margin (%)?
- I/O expansion capability?

Avionics rule:

Design for future payloads you don't know yet

Freshers never think this far.

1 1 Phase-2 COMMON FAILURES (Observed in Real Programs)

✗ Failure 1: Component-First Thinking

"Let's use this processor."

→ Architecture should force processor choice, not reverse.

✗ Failure 2: Ignoring Failure Modes

"Software will handle it."

→ Hardware must support recovery.

✗ Failure 3: No Partitioning Strategy

→ Certification nightmare.

Failure 4: No Timing Budget

→ Missed deadlines discovered too late.

1 2 Phase-2 EVALUATION CHECKS (Seniors Use These)

They may ask:

Check 1

"Where is the single point of failure?"

Check 2

"What happens if MC-A dies mid-mission?"

Check 3

"Which architectural choice is hardest to certify?"

Check 4

"What did you over-design, and why?"

1 3 Phase-2 OUTPUTS (MANDATORY)

You must produce:

1. High-level architecture diagram
2. Redundancy concept
3. Timing architecture
4. Partitioning strategy
5. Fault handling philosophy
6. Growth & margin assumptions

No outputs → Phase-2 is invalid.

1 4 Phase-2 COMPLETION CHECK (Mission Computer)

You are Phase-2 complete if you can answer:

"Which architectural decision, if changed later, would cause the most rework?"

Correct answer:

 Partitioning + redundancy philosophy

What Comes Next

👉 Phase-3: Detailed Hardware Architecture (Card-Level, Backplane, Power, Clocks)

This is where:

- VPX / CPCI decisions appear
- PPC vs ARM becomes concrete
- XMC / mezzanine cards enter
- Power, reset, clock trees are defined

If you want:

- Proceed directly to **Phase-3 (Mission Computer)**
- Simulate a **real design review interrogation**
- Train freshers using **Phase-2 evaluation tasks**
- Convert this into a **program lifecycle playbook**

Just say where to go next.