Flight software, watchdog, FDIR, redundancy and safe mode

Essential answer

Flight software, watchdog, FDIR, redundancy and safe mode. The question to solve is: How does the spacecraft protect itself when a sensor becomes inconsistent or a computer hangs? Here, understanding the system matters more than one equation. We reason with states, interfaces, margins and success criteria. The rest of the course turns that idea into an auditable line of reasoning: explicit units, stated assumptions, reproducible calculations, order-of-magnitude checks and interpretation limits. A result is useful only when the reader can explain what it measures, where every input came from and which engineering decision it can support.

Reasoning map

1. 1 — The concrete scene
2. 2 — Essential words, explained before using them
3. 3 — See the architecture before calculating
4. 4 — Formulas, only when they answer a question
5. 5 — What units and margins mean

Logical configuration allowing certain actions.

Monitor expecting a sign of life.

Before calculating

Quantity:	Flight software, watchdog, FDIR, redundancy and safe mode
Check:	units + order of magnitude + assumptions
Decision:	state what the result actually supports

Flight software, watchdog, FDIR, redundancy and safe mode

Logical configuration allowing certain actions.

Monitor expecting a sign of life.

Fault Detection, Isolation and Recovery.

Additional means able to take over a function.

Dependency able to defeat multiple channels.

Simplified state prioritising survival and diagnosis.

3 — See the architecture before calculating

Thresholds, consistency, voting and timeouts.

Identify likely region, not merely raise an alarm.

Retry, restart, switch or disable.

Reduce complexity to preserve power, thermal control and communication.

4 — Formulas, only when they answer a question

Here, understanding the system matters more than one equation. We reason with states, interfaces, margins and success criteria.

5 — What units and margins mean

Thresholds keep the monitored unit; timeouts use s or ms. Logic should state value, duration and applicable state.

6 — Three concrete demonstrations, calculated step by step

Exceedance 0.2 s, logic requires 2 s.

21.0 °C; 21.3 °C; 85 °C.

85 °C channel is suspect, root cause not proven.

Function maintained, further fault tolerance lost.

7 — Deepening: what the simplified diagram hides

Explicit transitions reduce implicit behaviour.

Too sensitive and too permissive are both risks.

Active reading

Sections:	3 — See the architecture before calculating ; 4 — Formulas, only when they answer a question
Verification:	A value is usable only with its unit and calculation boundary.
Exercise:	change one assumption and predict the direction of change before recalculating

Flight software, watchdog, FDIR, redundancy and safe mode

Identical copies can share the same design error.

A coherent state must be restored after reboot.

Simulation and hardware-in-the-loop test rare cases.

8 — Application to an Earth-Mars spacecraft

9 — Reference dossier: what a real project must still consider

Robust spacecraft software makes modes and transitions explicit: initialization, nominal operation, manoeuvre, communications, safe mode and recovery. Commands may be valid in one mode and forbidden in another. Explicit state machines make rare paths visible and testable.

A watchdog expects evidence that software or a processor is alive and may trigger reset or reconfiguration when that evidence stops. Poor design can cause needless resets or miss logically incorrect behaviour, so the monitored condition must be defined carefully.

Fault Detection, Isolation and Recovery are separate questions. Detect asks whether something is wrong; isolate asks where the likely cause is; recovery asks how to continue or become safe. Acting too quickly on a false alarm can create a real failure, while acting too slowly can propagate damage.

Safe mode deliberately reduces objectives to preserve essential power, thermal state, attitude and minimum communications. It still depends on hardware and software, so designers must analyse which faults could prevent safe mode itself.

Two computers running identical code may make the same wrong decision when the design error is shared. Hardware redundancy and design diversity solve different problems. A deliberately simpler backup function may sometimes be more robust but costs additional development and verification.

Critical problems often occur in uncommon combinations. Software-in-the-loop and hardware-in-the-loop testing can inject sensor faults, communications loss, resets and timing problems. A campaign that tests only nominal operation gives false confidence.

Remote updates need validated images, robust transfer, backup copies and rollback. Mars latency increases the value of local autonomy. A settlement also needs programming tools, signatures, procedures and compatibility records as part of its technical maintenance capability.

10 — Common traps and bad intuitions

Active reading

Sections:	8 — Application to an Earth-Mars spacecraft ; 9 — Reference dossier: what a real project must still consider
Verification:	A value is usable only with its unit and calculation boundary.
Exercise:	change one assumption and predict the direction of change before recalculating

Flight software, watchdog, FDIR, redundancy and safe mode

Thinking two identical copies remove common cause.

Using restart as universal response.

Creating a safe mode never tested end-to-end.

Flight software must lose functions without losing the spacecraft

A watchdog checks that a computer or task remains responsive. Automatic reset can recover a transient lock-up but can become dangerous if the root cause persists. Flight software therefore tracks context, limits repeated recovery actions and can enter a safe mode.

FDIR separates detection, isolation and recovery. One bad measurement does not automatically identify one bad sensor; comparisons and models may be needed. Recovery can disable a load, switch chains, change a setpoint or wait for Earth depending on the authorized autonomy.

Remote updates add configuration risk. A vehicle needs a known fallback image, integrity checks and the ability to return to a previous version if activation fails. In deep space, rollback capability is as important as the new code itself.

11 — Guided exercises

Question : What question comes before choosing hardware?

Question : Why is a nominal result insufficient?

12 — What to remember

Explain the topic in simple words before symbols.

Connect at least four interfaces with other subsystems.

Redo the three numerical examples without reasoning gaps.

13 — NASA sources for further study

NASA - 2026 State-of-the-Art: Small Spacecraft Avionics

NASA - 2026 State-of-the-Art: Ground Data Systems and Mission Operations

Active reading

Sections:	Flight software must lose functions without losing the spacecraft ; 11 — Guided exercises
Verification:	NASA - 2026 State-of-the-Art: Small Spacecraft Avionics | NASA Systems Engineering Handbook | NASA - 2026 State-of-the-Art: Ground Data Systems and Mission Operations
Exercise:	change one assumption and predict the direction of change before recalculating