Test Frames & Validation Strategy

1. Purpose of This Document

This document defines the **test frames** and **validation strategy** used during Phase 2 to verify correct CAN reception and system behavior *before* any embedded software is written.

The goal is to establish **what "correct" looks like** when observing CAN traffic and to define objective checks that can be repeated throughout later phases of the project.

This document completes **Phase 2 – CAN Fundamentals & Validation**.

---

2. Why Validation Comes Before Implementation

Validating CAN traffic early helps to:

- Separate protocol and wiring issues from software issues
- Detect configuration errors (bitrate, IDs, DLC)
- Establish trust in test tools and measurements
- Create a known-good reference for later debugging

Without this step, problems discovered later are harder to isolate and often misattributed to firmware or application logic.

---

3. Definition of Test Frames

A **test frame** is a deliberately designed CAN data frame whose properties and behavior are fully known in advance.

Each test frame is defined by: - CAN Identifier (ID) - Data Length Code (DLC) - Payload pattern - Transmission rate

Test frames are simple by design and avoid encoded signals or complex behavior.

---

4. Recommended Initial Test Frame Set

For initial validation, the following minimal set is recommended:

4.1 Frame A – High-Frequency Frame

- **ID:** Low numerical value (high priority)
- **DLC:** Fixed (e.g. 8 bytes)
- **Payload:** Incrementing counter pattern
- **Rate:** Fast periodic transmission (e.g. 10–20 ms)

Purpose: - Validate continuous reception - Detect dropped frames - Observe timing stability

---

4.2 Frame B – Low-Frequency Frame

- **ID:** Higher numerical value (lower priority)
- **DLC:** Fixed
- **Payload:** Static or slowly changing pattern
- **Rate:** Slow periodic transmission (e.g. 100–500 ms)

Purpose: - Validate coexistence of different priorities - Confirm arbitration behavior - Check multi-ID handling

---

5. Validation Criteria

For each test frame, validation is performed against the following criteria:

5.1 Identifier Validation

- Observed IDs match configured IDs
- No unexpected IDs appear on the bus

5.2 DLC Validation

- Observed DLC matches expected value
- No truncated or malformed frames are observed

5.3 Payload Validation

- Payload patterns match expected values
- Incrementing counters behave correctly

5.4 Timing Validation

- Transmission intervals match configured rates
- No unexpected jitter or bursts are observed

---

6. Error Observation and Interpretation

During validation, the following conditions should be actively checked:

- Missing frames
- Irregular timing
- Unexpected arbitration outcomes
- Error frames reported by tools

At this stage, any anomalies should be attributed to: - Transmit configuration - Bus wiring or termination - Bitrate mismatch

Embedded software is **not** yet a factor.

---

7. Documentation of Results

Validation results should be recorded as:

- Screenshots or logs of observed traffic
- Tables comparing expected vs observed behavior
- Notes on any anomalies or assumptions

This documentation serves as a **baseline reference** for all later phases.

---

8. Relationship to Later Phases

The validation strategy defined here will be reused in:

- Phase 3 (MCU software validation)
- Phase 4 (inter-processor communication validation)
- Phase 5 (Linux decoding and processing validation)

Having a stable test reference reduces uncertainty as system complexity increases.

---

9. What This Document Does Not Cover

This document intentionally does **not** define:

- Embedded firmware tests
- Driver-level diagnostics
- Signal decoding or DBC usage
- Performance optimization

These topics are introduced only when relevant.

---

10. Phase Completion Summary

With CAN fundamentals, transmit setup, and validation strategy defined:

- Phase 2 objectives are complete
- A known-good CAN reference environment exists
- The project is ready to move into **Phase 3 – MCU Software Architecture**

The next document will be:

- **PDF 3.1 – MCU Responsibilities & Task Design**