CAN Transmit Setup

1. Purpose of This Document

This document describes how **CAN transmit software** is used to generate test traffic for the project. The goal is to establish a **controlled, repeatable way to place known CAN frames on the bus** so that reception, validation, and later decoding can be verified step by step.

The setup described here is **vendor-agnostic** and focuses on *what must be transmitted* and *why*, not on any specific commercial tool.

This is the second document in **Phase 2 – CAN Fundamentals & Validation**.

---

2. Why a Controlled Transmit Setup Is Needed

Before connecting embedded hardware, it is essential to:

- Know exactly which CAN frames are present on the bus
- Control transmission rates and payloads
- Separate bus-level issues from software issues
- Create repeatable test conditions

Without a controlled transmit source, it becomes difficult to distinguish between: - Wiring problems - Timing issues - Driver bugs - Misinterpreted data

A CAN transmit setup acts as a **known-good reference**.

---

3. Role of CAN Transmit Software

CAN transmit software is responsible for:

- Sending CAN data frames onto the bus
- Defining message identifiers (IDs)
- Defining Data Length Code (DLC)
- Defining payload values
- Controlling message transmission rates

In this project, the transmit software is considered an **external system** and is not modified as part of the embedded design.

---

4. Transmit Configuration Elements

A minimal CAN transmit configuration consists of the following elements:

4.1 CAN Bitrate

- Must match the bitrate expected by the CAN controller
- Common values include 125 kbps, 250 kbps, and 500 kbps
- All nodes on the bus must use the same bitrate

Bitrate mismatches result in no communication or continuous error frames.

4.2 Message Identifiers (IDs)

- IDs should be chosen deliberately and documented
- Lower numerical IDs have higher priority
- Initial testing should use a small number of known IDs

Example (illustrative): - ID 0x100 – periodic status message - ID 0x200 – slower diagnostic-style message

---

4.3 Data Length Code (DLC)

- DLC defines how many bytes are transmitted
- Classical CAN supports 0–8 bytes
- DLC should remain constant during early testing

Changing DLC dynamically complicates validation and is avoided initially.

4.4 Payload Values

Payload bytes should be:

- Easy to recognize (e.g. counters, fixed patterns)
- Predictable over time
- Documented alongside the ID and DLC

Using simple patterns makes validation and debugging much easier.

4.5 Transmission Rate

- Defines how often a message is sent
- Expressed as period (e.g. 10 ms, 100 ms)
- Must be realistic relative to bus load

Early testing should avoid saturating the CAN bus.

---

5. Recommended Initial Test Traffic

For early validation, a minimal setup is recommended:

- 1–2 CAN IDs
- Fixed DLC (e.g. 8 bytes)
- Simple payload patterns
- Known, constant transmission rates

This keeps behavior observable and repeatable.

---

6. Observability and Verification

The transmit setup should allow:

- Visual confirmation that messages are being sent
- Inspection of transmitted IDs, DLC, and payloads
- Start/stop control of message transmission

This observability is essential for validating receive-side behavior later.

---

7. Relationship to System Boundaries

The CAN transmit software:

- Is **outside the system boundary**
- Acts as a test stimulus
- Provides a reference for validation

The embedded system must adapt to the transmitted traffic, not the other way around.

---

8. What This Document Does Not Define

This document intentionally does **not** define:

- Specific CAN transmit tools
- Graphical user interface layouts
- Vendor-specific configuration steps
- DBC files or signal decoding

These details are either tool-dependent or introduced in later phases.

---

9. Next Steps

With CAN transmit setup defined, the next document will describe:

- **PDF 2.3 – Test Frames & Validation Strategy**
- What constitutes a valid frame
- How reception is verified
- How expected vs observed behavior is compared

This completes Phase 2 – CAN Fundamentals & Validation.