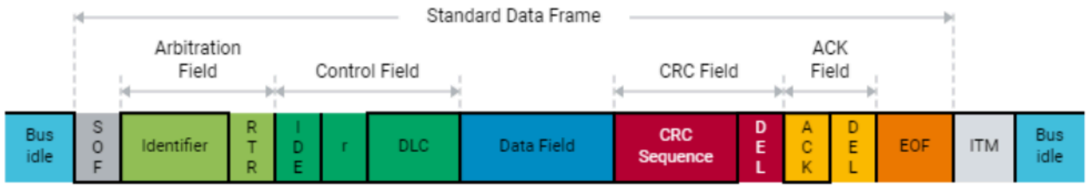
CAN Protocol Design Flow

Structure

Field	Bits	Description
Start of Frame (SOF)	1	Indicates the start of a new CAN frame.
Identifier	11	Identifies the message priority (lower ID = higher priority).
RTR (Remote Transmission Request)	1	0 for data frame, 1 for remote frame request.
IDE (Identifier Extension Bit)	1	0 for standard frame, 1 for extended frame.
Reserved Bit (r0)	1	Reserved for future use.
DLC (Data Length Code)	4	Specifies the number of data bytes (0-8 bytes).
Data Field	0-64	Actual message data (up to 8 bytes in classic CAN, up to 64 in CAN FD).
CRC (Cyclic Redundancy Check)	15	Error detection checksum for data integrity.
CRC Delimiter	1	Must be recessive (1) to confirm CRC transmission.
ACK Slot	1	The receiver sends a dominant bit (0) as acknowledgment.
ACK Delimiter	1	Must be recessive (1).
End of Frame (EOF)	7	Marks the end of the CAN message.

Intermission (IFS)	3	Space between consecutive frames.

Standard Data frame



Problems Encountered While Designing the Flow of CAN Protocol

During the development of the CAN-based communication flow between the object detection module and the braking system, we encountered several challenges that required careful consideration and problem-solving. Below, we outline the key issues we faced and the solutions we implemented.

1. Effective Use of Identifier Bits

One of the primary challenges we faced was determining how to effectively utilize the **identifier bits** in the CAN protocol. Since the braking system module processes each received frame individually, we initially struggled to define a clear prioritization strategy. Without a well-structured identifier scheme, critical signals, such as **object detection alerts**, might have been delayed due to lower-priority messages.

Solution: To address this, we assigned a **higher-priority identifier** to frames carrying the **"Object Detected"** signal whenever an obstacle was detected. By ensuring that these frames have the lowest numerical identifier (higher priority in CAN arbitration), they are transmitted and processed ahead of less critical messages, minimizing the reaction time of the braking system.

---

2. Managing the Complexity of CAN Protocol

The CAN protocol is inherently **large and complex**, incorporating multiple fields such as arbitration, control bits, CRC checks, acknowledgment mechanisms, and error-handling features. While implementing the entire protocol, we realized that

integrating CRC (Cyclic Redundancy Check) within the CAN frame handling logic significantly increased design complexity.

Solution: To simplify our implementation, we **excluded CRC computation from the core CAN protocol logic and implemented it separately**. By handling CRC independently, we streamlined the data transmission flow while still ensuring data integrity. This modular approach also made debugging and validation more manageable.

---

3. Verifying Data Integrity After Transmission

Initially, we faced uncertainty regarding **how to verify the accuracy of transmitted data after applying CRC at the sender side**. Since the CRC field is included in the frame before transmission, we needed a mechanism to validate whether the received data was correctly transmitted without corruption.

Solution: We implemented a **CRC checker on the receiver side**. This module recalculates the CRC for the received data and compares it to the transmitted CRC value. If the computed and received CRC values match, the data is considered valid; otherwise, an error is flagged. This approach ensures reliable error detection while keeping the sender's implementation straightforward.

Flow of Design

The **CAN-based communication flow** in our system is designed to ensure efficient and reliable transmission of critical data between the **Obstacle Detection Module** and the **ABS (Anti-lock Braking System) Module**. Below is a structured breakdown of the design flow:

1. **Obstacle Detection Module Activation**
    - The **Obstacle Detection Module** continuously monitors the surroundings for objects.
    - When an obstacle is detected, it generates an **object_detected signal**.
    - The module also reads and transmits the **current speed of the vehicle** to provide context for braking decisions.

2. **Frame Construction in the CAN Module**
    - The **CAN controller** receives the **object_detected signal** and **vehicle speed**.
    - It constructs a **CAN frame**, embedding:
        - **Identifier** (prioritization for arbitration)
        - **Object detection status**
        - **Vehicle speed data**
    - If an object is detected, the frame is assigned a **higher priority** to ensure immediate transmission.

3. **Transmission of the Frame over CAN Bus**
    - The CAN module **transmits** the constructed frame onto the **CAN bus**.
    - Arbitration ensures that **higher-priority messages (object detected)** are sent before less critical messages.

4. **Frame Reception by the ABS Module**
    - The **ABS module** receives the frame from the CAN bus.
    - It extracts the **object_detected signal** and **vehicle speed** for further processing.

5. **CRC Validation at the ABS Side**
    - The **ABS module performs CRC checking** to ensure the received frame is error-free.
    - If the CRC is invalid, the frame is **discarded or a retransmission is requested**.

6. **Decision Making by the ABS Module**
    - Based on the **object_detected status and vehicle speed**, the ABS system **calculates the required braking force**.
    - If an object is detected and the speed is above a critical threshold, the system **activates the braking mechanism**.
    - Otherwise, normal driving conditions continue.

7. **Braking Action Execution**
    - If the ABS module decides to **apply braking**, it sends a **control signal to the braking actuators**.
    - The actuators **modulate brake pressure** to **prevent wheel lock-up**, ensuring controlled and safe deceleration.

Conclusion

By systematically addressing the challenges in our CAN communication design, we developed a **robust, efficient, and reliable** system for the braking module. The structured flow ensures **real-time and accurate data exchange** between the **Obstacle Detection Module** and the **ABS**, enabling timely and critical decision-making. The integration of **priority-based identifiers**, **modular CRC implementation**, and **receiver-side error checking** enhances both the **safety and responsiveness** of the braking system. These design choices significantly improve the overall effectiveness and reliability of the CAN-based communication framework.

References

[1] A. R. Mohamed, A. M. Mohamed, E. A. El-din Zarif, M. A. Mohamed, and R. S. Abdlmonem, **"Automobile Security System Based on Face Recognition and CAN Bus Security Implementation,"** B.Sc. thesis, Faculty of Engineering, Cairo University, Giza, Egypt, Aug. 2020.

[2] Typhoon HIL, **"CAN Bus Protocol,"** [Online]. Available: https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/can_bus_protocol.html. [Accessed: Mar. 2025].

[3] University of California, Riverside, **"CAN Specification 2.0,"** [Online]. Available: http://esd.cs.ucr.edu/webres/can20.pdf. [Accessed: Mar. 2025].