POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Functional Safety in Can Open Communication for Electric Motor Controller

Master's Thesis of:
Andrea Leto

Advisor: Prof. Ferdinando Luigi Mapelli
Co-advisor: Eng. Stefano Mossina
Academic Year: 2023 - 2024
Student ID Number: 989988

Contents

1 Introduction **1**

2 Literature and Background **3**

2.1 Extension to CAN-Open Framework 4

2.2 BS EN 50325-1:2019 7

2.2.1 Definitions 8

2.2.2 Normal service, transport and mounting conditions . . . 10

2.3 BS EN 50325-4:2002 12

2.3.1 Purpose and Application 12

2.3.2 Requirements and Specifications 13

2.3.3 Classifications and Characteristics 14

2.3.4 Service Data Objects 16

2.3.5 Process Data Objects 16

2.3.6 Network Management (NMT) 17

2.3.7 Device Profiles 18

2.3.8 Emergency Messages and Error Control 19

2.4 BS EN 50325-5:2010 19

2.4.1 Functional Safety Requirements 20

2.4.2 Safety-Related Data Objects 21

2.4.3 Global Fail-Safe Command 21

2.4.4 Error Detection and Handling 22

2.4.5 Integration with CANopen 22

2.4.6 Protocol Specifications 23

2.4.7 Compliance and Certification 24

3 Experimental Setup and Methodology **25**
3.1 Object Dictionary . . . 25
3.1.1 Structure . . . 26
3.1.2 Data Types and Access Methods . . . 27
3.1.3 Communication Objects . . . 28
3.1.4 Entries . . . 29
3.1.5 Management . . . 30
3.2 Safety-Related Data Objects . . . 31
3.2.1 Overview . . . 32
3.2.2 Structure . . . 33
3.2.3 Timing Requirements . . . 34
3.2.4 COB-ID . . . 36
3.2.5 SRDO Mapping . . . 38
3.3 Global Fail-Safe Command . . . 39
3.3.1 Structure and Functionality . . . 42
3.3.2 Configuration of GFC . . . 43
3.3.3 Error Detection and Mitigation . . . 44
3.3.4 Implementation in CANopen Networks . . . 45
3.3.5 Applications . . . 46

4 Results and Analysis **49**
4.1 Scope . . . 55
4.2 Experimental Implementation . . . 56
4.2.1 GFC Transmit . . . 56
4.2.2 GFC Receive . . . 58
4.2.3 SRVT Timing . . . 59
4.2.4 SCT Timing . . . 61
4.2.5 COB-ID 1 Inconsistency . . . 63
4.2.6 COB-ID 2 inconsistency . . . 66
4.2.7 COB-ID 1 & COB-ID 2 inconsistency . . . 67

5 Conclusion **71**

Bibliography **73**

1 | Introduction

The rapid advancement of technology has led to increased reliance on electric motor controllers in various applications, ranging from industrial automation to automotive systems. Thus ensuring the functional safety of these controllers is very important, especially when they are integrated into complex networks like CANopen. This thesis explores the functional safety mechanisms in CANopen communication, focusing on electric motor controllers.

CANopen, a higher-layer protocol based on the Controller Area Network (CAN), facilitates robust communication in distributed control systems. It extends the basic functionalities of CAN, enabling detailed device management, real-time data exchange, and network-wide error handling. The protocol's flexibility and reliability make it a preferred choice for safety-critical applications.

Functional safety within CANopen is achieved through mechanisms such as Safety-Related Data Objects (SRDOs), Global

Fail-Safe Commands (GFC), and rigorous error detection protocols. SRDOs ensure the integrity of safety-critical data by transmitting redundant information, while GFCs provide a method for immediate system-wide shutdown in the event of a critical fault.

The objective of this research is to analyze and test these safety mechanisms through a series of experiments, using a practical setup involving an electric motor controller, Vehicle Control Unit (VCU), and various diagnostic tools. The aim is to validate the effectiveness of CANopen's safety protocols and their implementation in real-world scenarios, so highlighting the importance of data consistency and timely error detection for maintaining system integrity and safety.

2 | Literature and Background

The Controller Area Network (CAN) protocol, is a robust communication system initially designed for automotive applications and now widely used in industrial automation, medical devices, and mobile machinery. CAN operates on a broadcast communication principle, allowing efficient real-time data exchange. Its architecture features a multi-master, message-oriented system with strong error detection methods like bit monitoring and cyclic redundancy checks, ensuring reliable communication. CAN's non-destructive bitwise arbitration prioritizes critical data transmission, enhancing network efficiency. The protocol supports various data rates and network sizes, making it highly scalable and adaptable.

2.1 Extension to CAN-Open Framework

As industrial automation and other sectors began to adopt CAN, it became clear that additional functionalities were necessary to meet the complex demands of these environments. This necessity led to the creation of CANopen, a higher-layer protocol designed to extend the capabilities of CAN.

The primary motivation for extending CAN with CANopen was the need for a more sophisticated application layer to handle the intricacies of industrial automation. While the basic CAN protocol efficiently managed low-level data transmission with high reliability, it lacked the higher-layer functionalities required for complex industrial applications. Industrial environments demand precise control, real-time data processing, and intricate device interactions that the base CAN protocol alone could not sufficiently address. CANopen was developed to provide these advanced capabilities, enabling seamless integration and communication between various devices in an industrial network.

One of the significant enhancements introduced by CANopen is the standardized object dictionary, which organizes com-

munication objects, device parameters, and application data. This dictionary ensures that all devices on a network adhere to a common structure, facilitating interoperability between devices from different manufacturers.
Manufacturers can develop devices that can easily communicate and work together, reducing integration costs and time. The standardized approach also simplifies the configuration and maintenance of the network, making it more user-friendly and accessible.
Then, another main reason for the creation of CANopen was the need for real-time communication capabilities. Industrial applications often involve time-critical processes where delays or interruptions can lead to significant inefficiencies or hazards. While the CAN protocol provides basic support for prioritizing messages, CANopen enhances this by defining specific communication objects, such as Process Data Objects (PDOs) and Service Data Objects (SDOs). PDOs facilitate the fast and efficient exchange of real-time data without additional protocol overhead, while SDOs manage more complex data exchanges, such as configuration and diagnostic information. This distinction allows CANopen to handle both time-critical and less urgent communications effectively, ensuring that the system can

respond promptly to real-time demands.

Network management is another key area where CANopen improves on the basic CAN protocol. Industrial networks need strong systems to start, monitor, and manage devices to work reliably. CANopen meets this need with its Network Management (NMT) protocol. NMT controls the states of all devices, moving them through Initialization, Pre-operational, Operational, and Stopped states as needed. This process ensures devices are set up and synchronized properly before regular operations begin, boosting the network's overall reliability and stability.

In addition to managing device states, CANopen provides advanced error detection and handling mechanisms. While the basic CAN protocol includes error detection features, CANopen builds on these by defining specific error control protocols that ensure any communication issues are promptly identified and resolved. These mechanisms include heartbeat messages, node guarding, and emergency messages, which continuously monitor the status of devices and the integrity of data exchanges.

The development of CANopen also addressed the need for scalability and flexibility in industrial networks: CANopen supports various device profiles that standardize the behav-

ior and communication of different types of devices, such as sensors, actuators, and controllers. These profiles ensure that devices with similar functions can be easily integrated into the network, regardless of their manufacturer. This scalability is important, as industrial systems grow and evolve, allowing new devices to be added without extensive reconfiguration or compatibility issues.

2.2 BS EN 50325-1:2019

This legislation is a European Standard that provides the general requirements for industrial communication subsystems based on ISO 11898 (CAN) for controller-device interfaces. This standard outlines the specifications for communication protocols, network interfaces, and the necessary electrical and mechanical characteristics to ensure reliable communication and interoperability among devices in industrial environments. It aims to enhance productivity through reduced wiring, improved diagnostics, and robust communication between control devices.

2.2.1 Definitions

The BS EN 50325-1:2019 standard provides crucial definitions for understanding the communication subsystems based on ISO 11898 (CAN) - [BS EN 50325-1:2019, pp.7-9]

CAN (Controller Area Network): CAN is a broadcast-oriented communications protocol designed for cost-effective connectivity and improved device-level diagnostics in industrial automation

Controller: A controller is a network component such as a Programmable Logic Controller (PLC), PC, or similar computing hardware where control applications or process software operate

Controller-device interface: This refers to the structured network of nodes and connections that transport data within an industrial control system, comprising a controller and field devices

Cyclic: Cyclic data exchange occurs when devices or controllers generate data at a predetermined rate suitable for the device and application's needs

Electromechanics: Electromechanics encompasses the physical components specified by a controller-device interface,

including interconnecting wires, cables, media, and physical connectors

Field device: A field device is a physical unit containing application elements and potentially communication elements, such as sensors, actuators, and control devices

Interface: An interface is the boundary between network entities, defined by functional, signal, or relevant characteristics, delineating their interaction

Messaging: Messaging involves the process where a device or controller sends or requests device I/O, diagnostic, or configuration information

Multicast: Multicast refers to a data exchange process where a device or controller sends a network message to multiple devices or controllers for respective action

Network power supply: This is a power supply with characteristics and parameters suitable for the network's functionality and capability

Polling: Polling is a data exchange process where a device, such as a controller, sends or requests data from another specific device

2.2.2 Normal service, transport and mounting conditions

The standard outlines specific normal service conditions that components of a CAN-based network must meet to ensure reliable operation in industrial environments [BS EN 50325-1:2019, pp.11-12]

General Guidelines: Components must adhere to relevant standards or agreements between the manufacturer and user. Information given in the manufacturer's catalog can replace formal agreements if necessary

Deviations from Standard Conditions: If the operational conditions differ from those specified in the standard, users must declare these deviations and consult the manufacturer to ensure the suitability of use under such conditions

Ambient Air Temperature: Components must maintain their operational characteristics within an ambient temperature range of -20°C to +60°C. This requirement ensures that devices function correctly in a variety of environmental conditions typically encountered in industrial settings

Altitude: Network components must be capable of operating at altitudes up to 2000 meters. For operation at higher altitudes, an agreement between the manufacturer and the user is necessary to address any additional requirements

Humidity: Components must operate effectively at 40°C with the relative humidity of the air not exceeding 95%. This specification ensures that devices can withstand high humidity environments without compromising performance

Pollution Degree: Devices and components need to operate under environmental conditions of pollution degree 3, as defined in EN 60947-1:1999. This condition refers to environments where conductive pollution or dry non-conductive pollution that becomes conductive due to condensation may occur

Sealed Connectors: Connectors must be specified and protected to at least IP 67 according to EN 60529, ensuring they are dust-tight and protected against temporary immersion in water. Compliance with Annex D of EN 60947-5-2:1998 is also required to maintain reliability in harsh conditions

These conditions guarantee that CAN-based network com-

ponents are robust enough to operate reliably in diverse and demanding industrial environments, supporting the overall stability and efficiency of the system.

2.3 BS EN 50325-4:2002

The BS EN 50325-4:2002 standard is integral to the CAN suite, specifically addressing the CANopen protocol

2.3.1 Purpose and Application

The primary purpose of BS EN 50325-4:2002 is to standardize the communication between programmable logic controllers (PLCs) and various devices. This standard ensures interoperability, reliability, and efficiency in data exchange. By facilitating the integration of equipment from various manufacturers, CANopen improves system flexibility, scalability, and performance in automation systems, ranging from manufacturing processes to machine control [BS EN 50325-4:2002, p.5]. The CANopen protocol achieves this by providing a robust framework that includes standardized communication objects, device profiles, and network management features.

2.3.2 Requirements and Specifications

Normal Service Conditions: Devices must operate reliably across a broad range of temperatures, altitudes, humidity, and pollution levels. This ensures that equipment can function effectively in diverse environmental conditions commonly found in industrial settings. The standard specifies that components should be designed to withstand ambient air temperatures from -20°C to +60°C, operate at altitudes up to 2000 meters, and function in environments with a relative humidity of up to 95% at 40°C. Additionally, devices should be capable of operating under pollution degree 3 conditions as defined in EN 60947-1:1999 [BS EN 50325-4:2002, p.11].

Mounting Conditions: The installation procedure must follow the standards prescribed in BS EN 50325-1:2019. This includes secure placement, appropriate orientation, and environmental safeguarding of devices to maintain network reliability and stability. Proper mounting is crucial for ensuring that devices are protected from environmental factors that could affect their performance and longevity [BS EN 50325-4:2002, p.12].

Performance Requirements: The system's communica-

tion speed and timing need to meet recommended levels to guarantee consistent and error-free data exchange across different distances. This is critical for maintaining the integrity and efficiency of industrial automation processes. The standard outlines specific performance metrics that must be achieved to ensure high-speed, reliable communication [BS EN 50325-4:2002, p.12].

Electromagnetic Compatibility (EMC): Devices must conform to specific EMC regulations to avoid electromagnetic disturbances that could impact device function and network performance. Ensuring EMC compliance is vital for the smooth operation of industrial networks in environments with potential electromagnetic interference. The standard requires devices to adhere to EMC directives to minimize the risk of interference and ensure reliable operation [BS EN 50325-4:2002, p.12].

2.3.3 Classifications and Characteristics

Reference Model: The CANopen reference model is based on the Open Systems Interconnection (OSI) framework, which standardizes industrial network communication across different layers, from application processes to physical sig-

naling. This model ensures improved interoperability and efficiency among devices. The OSI model helps structure the communication process, ensuring that data is correctly transmitted and received across the network [BS EN 50325-4:2002, p.13].

Device Model: This model describes devices in terms of communication and application functionality. It details the structure of the device, which serves as an interface between communication and application layers. The device model provides a standardized approach to describing the capabilities and behaviors of devices within the network, facilitating easier integration and interaction between different devices [BS EN 50325-4:2002, p.14].

Communication Model: The communication model focuses on the mechanisms for message transmission, including synchronous and asynchronous messages. It defines three types of communication relationships: master/slave, client/server, and producer/consumer. These relationships are fundamental to organizing and managing data exchange in the network. The master/slave model is typically used for straightforward command and response communications, the client/server model for complex interactions, and the

producer/consumer model for efficient data broadcasting to multiple nodes [BS EN 50325-4:2002, p.15].

2.3.4 Service Data Objects

SDOs provide a detailed-data exchange mechanism crucial for device configuration, diagnostics, and monitoring. They operate on a client-server model, allowing one-to-one communication for data exchange and supporting both expedited (single message) and segmented (multi-message) transfers. SDOs are ideal for non-time-critical data exchanges, such as device initialization, parameter setting, or firmware updates. They offer flexible access to complex data structures within the object dictionary, enhancing network management and device configurability. The ability to handle detailed configuration and diagnostics makes SDOs essential for setting up and troubleshooting devices in an industrial network [BS EN 50325-4:2002, p.16].

2.3.5 Process Data Objects

These objects facilitate real-time data exchange between nodes, which is crucial for real-time control tasks and status monitoring. There are two types of process data objects:

Transmit process data objects (TPDOs) for sending data from a device and Receive process data objects (RPDOs) for receiving data. This structure enables efficient and directional data flow. Process data objects allow dynamic mapping of application-specific objects to optimize network bandwidth by selecting only relevant data for transmission. By bypassing the CANopen protocol stack for direct data transfer, they reduce latency, enhancing the system's real-time response capabilities. This immediate data exchange is vital for applications requiring prompt updates and quick reaction times, such as in motion control and safety systems [BS EN 50325-4:2002, p.17].

2.3.6 Network Management (NMT)

Network management is another area where CANopen significantly extends the capabilities of the base CAN protocol. Industrial networks require robust mechanisms for initializing, monitoring, and managing devices to ensure reliable operation. CANopen addresses this need with its comprehensive network management services, including the Network Management (NMT) protocol. NMT oversees the states of all devices on the network, transitioning them

between Initialization, Pre-operational, Operational, and Stopped states as necessary. This functionality ensures that devices are correctly configured and synchronized before starting regular operations, enhancing the overall reliability and stability of the network [BS EN 50325-4:2002, p.18].

2.3.7 Device Profiles

CANopen defines specific device profiles to standardize the behavior and communication of different types of devices, such as sensors, actuators, and controllers. These profiles ensure interoperability between devices from different manufacturers by providing a common framework for device functionalities. Each profile includes specifications for device parameters, communication objects, and application-specific behaviors. This standardization simplifies the integration and maintenance of devices within the network, making it easier to expand and modify the system as needed [BS EN 50325-4:2002, p.19]

2.3.8 Emergency Messages and Error Control

CANopen includes advanced error detection and handling mechanisms to ensure the reliability of the network. Emergency messages are used to report critical errors that require immediate attention. These messages are prioritized to ensure they are quickly received and processed by relevant devices. Additionally, CANopen employs various error control protocols, such as heartbeat messages and node guarding, to continuously monitor the status of devices and the integrity of data exchanges. These mechanisms are essential for maintaining a stable and reliable network, particularly in safety-critical applications where communication failures can have severe consequences [BS EN 50325-4:2002, p.20].

2.4 BS EN 50325-5:2010

BS EN 50325-5:2010 specifies the communication profile and protocols for functional safety communication enhancements in CANopen networks. It aims to ensure that data transmission within safety-critical industrial environments

is reliable and secure. The standard focuses on the safety communication layer, which includes mechanisms for error detection and correction, thus ensuring the integrity and reliability of data exchanges [BS EN 50325-5:2010, p.5]

2.4.1 Functional Safety Requirements

The **Safety Communication Layer** includes structures for ensuring data integrity and detecting communication errors, focusing on Safety-Related Data Objects, Global Fail-Safe Command, and various safety measures. Safety-Related Data Objects serve as the interface for safe data transfer, ensuring integrity through cyclic and event-driven transmissions. Global fail-safe commands enables immediate switching of Safety-Related Logical Devices (SRLDs) to a safe state in case of errors, significantly improving system reaction times. Safety measures include sequence numbering, timestamping, connection authentication, and redundancy checks to ensure comprehensive error detection and mitigation [BS EN 50325-5:2010, p.7]

2.4.2 Safety-Related Data Objects

These objects are critical components of the safety communication layer. They ensure that data transmitted between devices maintains its integrity and is free from errors. Each safety-related data object consists of two CAN messages: one for the actual data and one for the redundant data, which are cross-checked to detect discrepancies. This dual-message system enhances error detection and correction capabilities, ensuring that safety-critical data is reliably transmitted across the network [BS EN 50325-5:2010, p.10]

2.4.3 Global Fail-Safe Command

The global fail-safe command mechanism allows the immediate transition of all SRLDs to a predefined safe state upon detecting a critical error. This functionality is essential for maintaining the safety of the entire system, as it ensures that all devices can be quickly and effectively brought to a safe state, preventing potential hazards. The global fail-safe command enhances system reliability by providing a rapid response to critical failures [BS EN 50325-5:2010, p.12]

2.4.4 Error Detection and Handling

The safety communication layer implements rigorous error detection and handling mechanisms. These include:

- **Cyclic Redundancy Checks (CRC):** Ensures data integrity by detecting errors in transmitted data blocks [BS EN 50325-5:2010, p.15]
- **Heartbeat Monitoring:** Monitors the status of devices and detects any communication failures. If a heartbeat message is not received within a specified time frame, the system can take corrective actions [BS EN 50325-5:2010, p.18]
- **Node Guarding and Life Guarding:** Ensures continuous monitoring of the node's operational status. These mechanisms detect failures in real-time and trigger appropriate responses to maintain system integrity [BS EN 50325-5:2010, p.20]

2.4.5 Integration with CANopen

The CANopen Safety protocol is designed to be fully compatible with the existing CANopen framework. This integration ensures that safety and standard operational data

can coexist within the same network, facilitating seamless communication and interoperability between devices. The extended protocols defined in CANopen Safety do not interfere with standard CANopen operations, allowing for flexible and scalable network designs [CiA 301, p.11]

2.4.6 Protocol Specifications

Safety-Related Data Object Configuration: Each object is configured with specific parameters that define its behavior and operational limits. These parameters include the transmission type, communication cycle period, and timeout values. Proper configuration of safety-related data objects is essential to ensure that they function correctly within the safety communication framework [BS EN 50325-5:2010, p.25]

Global Fail-Safe Command Implementation: The global fail-safe command protocol specifies the format and transmission requirements for fail-safe commands. These commands must be prioritized and transmitted with minimal latency to ensure immediate response from all SRLDs. The protocol includes mechanisms for acknowledging receipt and execution of the fail-safe commands, further en-

hancing reliability [BS EN 50325-5:2010, p.28]

Error Control Protocols: CANopen Safety defines specific error control protocols that govern how errors are detected, reported, and managed. These protocols include detailed procedures for handling various types of errors, from transient communication glitches to persistent hardware failures. The error control protocols are designed to minimize downtime and ensure rapid recovery from errors, maintaining the overall safety and efficiency of the network [CiA 301, p.73]

2.4.7 Compliance and Certification

Devices and systems implementing CANopen Safety must undergo rigorous testing and certification processes to ensure compliance with the standard. These processes involve verifying the implementation of safety features, testing for robustness under various conditions, and ensuring interoperability with other CANopen Safety devices. Certification ensures that devices meet the high safety and reliability standards required for critical applications [BS EN 50325-5:2010, p.30]

3 | Experimental Setup and Methodology

The experimental setup used to test the functional safety mechanisms in CANopen communication is now detailed, by describing the methodology for simulating faults, generating data inconsistencies, and measuring system responses to validate the effectiveness of safety protocols.
The CANopen protocol, standardized under CiA 301, provides a robust framework for communication in embedded systems and industrial automation. The standard formalizes the structure of CANopen applications, focusing on its object dictionary, communication objects, and network management.

3.1 Object Dictionary

The CANopen object dictionary is a crucial component in the CANopen protocol, providing a structured and stan-

dardized method for organizing device parameters and communication objects. It consists of a collection of all parameters accessible via the CANopen network, serving as the interface between the application and communication layers. Each entry in the object dictionary is uniquely identified by a 16-bit index and, optionally, an 8-bit sub-index, allowing for hierarchical organization of data. The object dictionary is divided into several areas, each serving specific purposes in device configuration and communication.

3.1.1 Structure

Communication Profile Area: This area includes entries that define the communication parameters for the CANopen device. It contains standardized objects such as CANopen Node-ID, baud rate, and communication-related objects like COB-IDs (CAN Object Identifiers). These objects are essential for establishing and managing communication within the network.

Manufacturer-Specific Profile Area: This section allows device manufacturers to define proprietary parameters that are unique to their devices. It provides flexibility for manufacturers to include custom functionalities and con-

figurations that are not covered by the standardized object dictionary entries.

Standardized Device Profile Area: This area includes objects defined by CiA standards for specific types of devices, ensuring interoperability between devices from different manufacturers. For example, profiles for devices like encoders, drives, and sensors are standardized here, specifying mandatory and optional objects required for each device type.

3.1.2 Data Types and Access Methods

Entries in the object dictionary can represent various data types, including integers, floating-point numbers, strings, and complex data structures. Each entry is associated with specific access methods that define how data can be read from or written to the object dictionary. These access methods include:

Direct Access: Direct access methods allow immediate reading or writing of simple data types, such as integers or boolean values, from specific object dictionary entries.

Array Access: Array access methods facilitate the handling of arrays or lists of data, enabling efficient manage-

ment of multiple related parameters stored sequentially within the object dictionary.

Record Access: Record access methods are used for complex data structures that consist of multiple sub-entries, each identified by a sub-index. This hierarchical organization allows for detailed configuration and access of multi-parameter objects.

3.1.3 Communication Objects

The object dictionary includes several types of communication objects that facilitate data exchange and network management. Key communication objects include:

Process Data Objects: such objects enable the real-time transfer of process data between devices. Each process data object has a unique identifier and can be configured for cyclic, synchronous, or event-driven transmission. The mapping of application-specific data to process data objects allows for efficient bandwidth usage and minimal latency.

Service Data Objects: they provide a client-server mechanism for accessing the object dictionary. They support both expedited and segmented transfers, allowing for the

configuration and diagnostic data exchange. SDOs are crucial for device setup, parameterization, and monitoring.

Network Management Objects: NMT objects manage the operational state of devices within the network. They control the transition between initialization, pre-operational, operational, and stopped states, ensuring synchronized network operation.

Synchronization Objects: SYNC objects coordinate synchronous operations by sending periodic synchronization signals to all devices on the network. These signals ensure that time-sensitive tasks are performed simultaneously across the network.

Emergency Objects: EMCY objects are used for error reporting. When a device detects an error, it sends an emergency message to notify other devices on the network. This enables quick identification and resolution of issues, enhancing the reliability of the system.

3.1.4 Entries

Detailed entries in the object dictionary specify the parameters and their attributes. Each entry includes:

- **Index and Sub-Index:** The unique identifiers for the entry and its sub-entries.
- **Data Type:** Specifies the type of data stored in the entry, such as integer, float, or string.
- **Access Rights:** Defines whether the entry is read-only, write-only, or read/write.
- **Default Value:** The initial value assigned to the entry upon device startup.
- **Range Limits:** Specifies the valid range of values for the entry, ensuring data integrity.

3.1.5 Management

Managing the object dictionary involves several key tasks, including:

Initialization: Upon device startup, the object dictionary is initialized with default values. This ensures that all parameters are set to known states, providing a reliable starting point for device operation.

Configuration: During device configuration, parameters in the object dictionary can be modified to suit specific application requirements. This can be done via SDOs, which allow for secure and controlled access to configuration data.

Monitoring: Continuous monitoring of the object dictionary entries is essential for maintaining operational integrity. Monitoring tools can read values from the object dictionary to ensure that devices are functioning correctly and within specified limits.

Diagnostics: Diagnostic processes use the object dictionary to store and retrieve error codes and status information. This facilitates troubleshooting and maintenance by providing detailed insights into device performance and issues.

3.2 Safety-Related Data Objects

Safety-Related Data Objects (SRDOs) are a critical component of the CANopen Safety protocol, which extends the CANopen standard to meet the stringent requirements of safety-critical applications. The maximum number of SRDO producers in the system is limited to 64.
Each supported SRDO from SRDO 1 to SRDO 64 shall have its own object with an index from 1301_{h} to 1340_{h}, where SRDO 1 shall correspond to the object at index 1301_{h}, SRDO 2 shall correspond to the object at index 1302_{h}, and so on.

Attribute	Definition
Index	1301_h to 1340_h
Name	SRDO communication parameter
Object code	RECORD
Data type	SRDO communication parameter record (0026_h)
Category	Mandatory for each supported SRDO

Figure 3.1: SRDO Object Definition

3.2.1 Overview

SRDOs are designed to ensure the safe transmission of data in environments where system reliability and safety are paramount. They achieve this by providing a dual-channel communication mechanism that detects and mitigates errors, ensuring that any data transmitted is accurate and reliable. SRDOs consist of two CAN data frames. These frames are transmitted sequentially:

- **First CAN Data Frame:** Carries the actual safety-related data.
- **Second CAN Data Frame:** Contains the bitwise *inverted* version of the data in the first frame.

This dual-frame structure is designed to **enhance data integrity checks** during transmission: by comparing the

two frames, the receiving system can detect any corruption that might occur during the transmission. If the second frame does not exactly invert the first, an error is assumed, and appropriate safety measures are triggered.

3.2.2 Structure

SRDO's implementation, within the object dictionary, consists in specifying the parameters of the communication object, through the respective sub-index field

- **Sub-index 00_h:** defines the "Highest sub-index supported," which is mandatory, read-only, and not mappable to PDO. Its default and value range is set to 06_h.
- **Sub-index 01_h:** "Information direction" specifies whether the SRDO will be used for transmitting (set to 01_h) or receiving (set to 02_h).
- **Sub-index 02_h:** in transmission determines the "Refresh Time", that is the interval at which the SRDO is transmitted cyclically; in reception represents the "Safety Cycle Time" (SCT), i.e the maximum interval before a safety event must be processed

- **Sub-index 03_h:** in reception, it is the "Safety-Related Validation Time" (SRVT) — to specify the time allowed to validate the incoming data. It is not applicable in transmission.
- **Sub-index 04_h:** "Transmission type" describes the type of transmission used, standard or emergency.
- **Sub-index 05_h:** "COB-ID 1" assigns the CAN identifier for the first message of the SRDO, ensuring unique identification within the network. It is specifically calculated as $0x000000FF_h$ + 2*Node-ID_h, and has to be an odd number between 257_d and 383_d.
- **Sub-index 06_h:** "COB-ID 2" assigns the CAN identifier for the bitwise inverted data frame of the SRDO, following immediately after the first. It is specifically calculated as 0x00000100 + 2*Node-ID, and has to be an even number between 258_d and 384_d.

3.2.3 Timing Requirements

An SRDO shall be transmitted cyclically, and such transmission should be monitored. An SRDO may be transmitted event-driven in addition to the cyclic transmission, if required. The event-driven transmission is used to ensure

a fast reaction for NSR application: the following figure shows a cyclic SRDO transmission with cycle time t_{cycle} and an event-driven SRDO in between.

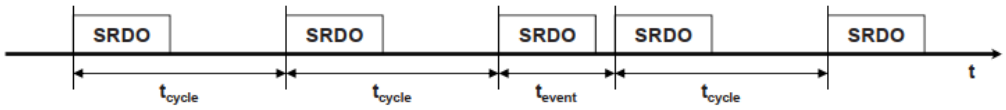

Figure 3.2: Example of SRDO transmission

The SCT is used to monitor the frequency and regularity of the SRDO transmissions over time, ensuring continual communication, then it applies to the sequence of messages to detect loss or delay in expected communication; if an expected SRDO is not received within the SCT interval, it is assumed that a fault has occurred, and the system will take safety actions

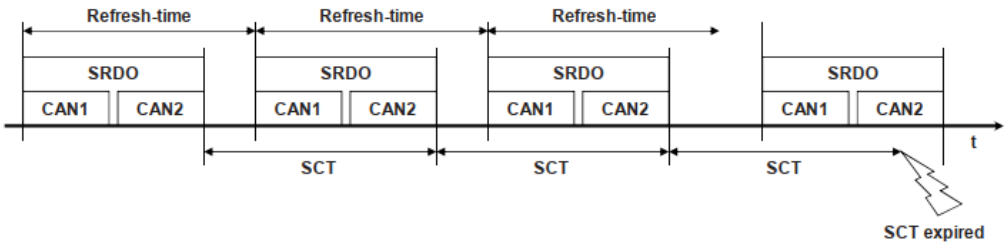

Figure 3.3: SCT Timing

On the other hand, SRVT is concerned with the integrity

of individual SRDO packets within a single transmission event: it applies within a single message's two-part structure to detect data corruption or incomplete transmissions. Then it checks the time between the reception of the two consecutive frames that make up a single SRDO, ensuring that both parts of an SRDO are received within a predefined time window

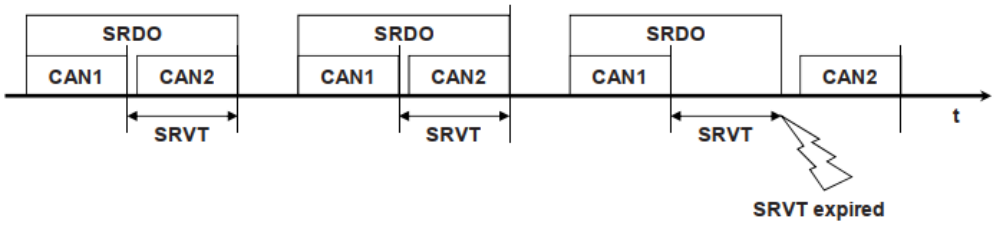

Figure 3.4: SRVT Timing

3.2.4 COB-ID

A COB-ID is a numerical identifier included in every CAN frame transmitted on the network. It consists of either an 11-bit identifier (Standard Frame) or a 29-bit identifier (Extended Frame), depending on the requirements of the network and its configuration. This identifier determines the priority of the message, with lower numerical values having higher priority.
This identifier is subdivided in tw fields

- the **function code** (4 bits for standard frames) is a part of the COB-ID that specifies the type of message being sent. It's used to differentiate between various standardized message types such as Process Data Object (PDO), Service Data Object (SDO), Node Guarding, and Heartbeat messages.
- **Node ID** (7 bits): represents the identifier of the node (device) that is sending or receiving the message

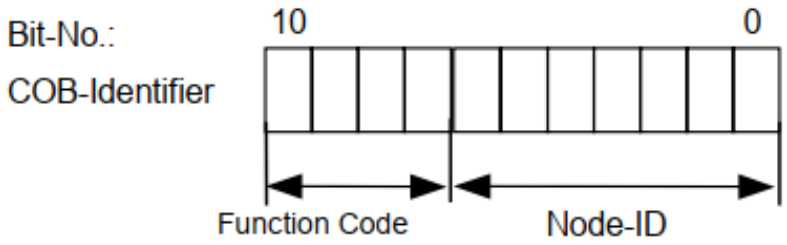

Figure 3.5: COB-ID allocation scheme

The Function Code mapping is provided by the Peer-to-Peer Objects of the Pre-defined Connection Set, in the BS EN 50325-4-2002 standard

The PLC (consumer), knowing the COB-ID, can set up its message filters to receive and process this specific message, ignoring all others not relevant to its operation. This a relevant result in view of data filtering requirements.

Object	Function code (binary)	Resulting COB-IDs	Communication parameters at Index
EMERGENCY	0001	129 (81h) – 255 (FFh)	1014h, 1015h
PDO1 (tx)	0011	385 (181h) – 511 (1FFh)	1800h
PDO1 (rx)	0100	513 (201h) – 639 (27Fh)	1400h
PDO2 (tx)	0101	641 (281h) – 767 (2FFh)	1801h
PDO2 (rx)	0110	769 (301h) – 895 (37Fh)	1401h
PDO3 (tx)	0111	897 (381h) – 1023 (3FFh)	1802h
PDO3 (rx)	1000	1025 (401h) – 1151 (47Fh)	1402h
PDO4 (tx)	1001	1153 (481h) – 1279 (4FFh)	1803h
PDO4 (rx)	1010	1281 (501h) – 1407 (57Fh)	1403h
SDO (tx)	1011	1409 (581h) – 1535 (5FFh)	1200h
SDO (rx)	1100	1537 (601h) – 1663 (67Fh)	1200h
NMT Error Control	1110	1793 (701h) – 1919 (77Fh)	1016h, 1017h

Figure 3.6: Predefined Connection Set

3.2.5 SRDO Mapping

The objects indexed from 1381_h to $13C0_h$ associated with each SRDO (Safety-Related Data Object) typically represent the specific messages or data packets that are to be transmitted through the SRDOs.

Attribute	Definition
Index	1381_h to $13C0_h$
Name	SRDO mapping parameter
Object code	ARRAY
Data type	UNSIGNED32
Category	Mandatory for each supported SRDO

Figure 3.7: Object Definition

Each object in the index range 1381h to 13C0h stores specific data or parameters that the SRDO will transmit across the network. This data could include operational parameters, status information, control commands, or safety-related signals specific to the needs of the safety system (the data payloads that SRDOs are responsible for transmitting).
SRDOs are designed to transmit these data securely and reliably within a network that adheres to safety standards (like EN 50325-5, based on CANopen safety communication protocols). The safety integrity of the data transmission is crucial, and these objects hold the data that must be protected against corruption, loss, or unauthorized alteration.
Each SRDO is mapped to a specific object that contains the data to be transmitted. This mapping ensures that when an SRDO is activated or sent, it correctly references the data stored at a particular index.

3.3 Global Fail-Safe Command

The Global Fail-Safe Command (GFC) is a critical feature of the CANopen Safety protocol, designed to ensure the

immediate and reliable transition of all connected devices to a predefined safe state in the event of a detected fault or emergency. The driving mechanism is integral to the safety communication layer of CANopen Safety. It provides a standardized method for sending a global command that forces all Safety-Related Logical Devices (SRLDs) within the network to enter a safe state. This mechanism is essential for preventing accidents and ensuring the safety of personnel and equipment in industrial environments.
Within the object dictionary, it is located specifically at index 1300_{h}

Attribute	Definition
Index	1300_{h}
Name	Global fail-safe command parameter
Object code	VAR
Data type	UNSIGNED8
Category	Optional

Figure 3.8: Object Definition

The respective entry is defined in the following way:

- **Sub-index 00_{h}**: defined by default
- **Access:** Read-write (RW), meaning it can be updated or polled by safety management systems to either set

or check the status of the GFC.

- **PDO Mapping:** No, indicating that this parameter does not participate in PDO mapping but operates independently.
- **Value Range:**
  - 00_h — GFC is not valid (inactive)
  - 01_h — GFC is valid (active)
  - 02_h to FF_h — Reserved for future use or specific implementation nuances
- **Default Value:** set to 00_h (indicating that by default, the GFC is inactive unless specifically triggered.)

Attribute	Definition
Sub-index	00_h
Access	rw
PDO mapping	No
Value range	00_h — GFC is not valid 01_h — GFC is valid 02_h to FF_h — reserved
Default value	00_h

Figure 3.9: Entry Definition

3.3.1 Structure and Functionality

Global Command Message: The GFC is transmitted as a high-priority CAN message with a unique identifier that distinguishes it from other types of messages. The content of this message is minimal, typically consisting of a command code that instructs all SRLDs to transition to the safe state.

Immediate Response: Upon receiving the GFC, all SRLDs are required to immediately cease their current operations and switch to their predefined safe states. This immediate response is crucial in preventing the propagation of errors and mitigating potential hazards.

Predefined Safe States: Each SRLD must be configured with a predefined safe state that it will enter upon receiving the GFC. These states are designed to ensure that the device and the overall system remain safe, even in the presence of faults. Examples of safe states include stopping all motion, disconnecting power, or activating emergency brakes.

3.3.2 Configuration of GFC

Configuring the GFC involves setting up the network and devices to recognize and appropriately respond to the global fail-safe command. This includes:

Device Configuration: Each SRLD must be programmed with a specific response to the GFC. This involves setting the appropriate parameters in the device's object dictionary to define the actions to be taken when the GFC is received.

Network Configuration: The CANopen network must be configured to prioritize the GFC message, ensuring it is transmitted and received with the highest urgency. This may involve setting the GFC message to have the highest possible priority in the CAN identifier scheme.

Validation and Testing: Once configured, the GFC mechanism must be thoroughly tested to ensure that all SRLDs correctly interpret and respond to the command. This testing includes simulating fault conditions and verifying that all devices transition to their safe states as expected.

3.3.3 Error Detection and Mitigation

The GFC mechanism plays a vital role in the error detection and mitigation strategies of CANopen Safety. By providing a quick and reliable method for transitioning devices to a safe state, the GFC helps to contain and manage errors that could otherwise lead to catastrophic failures.

Trigger Conditions: The GFC can be triggered by various conditions, such as the detection of critical errors, loss of communication, or manual activation by an operator. These trigger conditions must be carefully defined and monitored to ensure the GFC is activated only when necessary.

System Monitoring: Continuous monitoring of the network and devices is essential to detect conditions that may require the activation of the GFC. This monitoring includes checking for error states, communication failures, and other anomalies that could compromise system safety.

Automatic and Manual Activation: The GFC can be configured to activate automatically upon detecting specific fault conditions, or it can be manually triggered by an operator. This dual activation mechanism provides flexi-

bility and ensures that the GFC can be deployed quickly in a variety of scenarios.

3.3.4 Implementation in CANopen Networks

Implementing the GFC in a CANopen network involves integrating it with the existing communication framework while ensuring that all safety requirements are met. This includes:

Network Integration: The GFC message must be integrated into the network's communication schedule, ensuring it is given the highest priority and can be transmitted immediately when needed. This may involve configuring message filters and priority settings within the CANopen stack.

Device Integration: Each SRLD must be capable of recognizing and responding to the GFC. This requires updating the device firmware and configuring the object dictionary to support the GFC message and its associated actions.

Safety Protocols: Implementing comprehensive safety

protocols is essential to manage the activation and response to the GFC. These protocols define the steps to be taken when the GFC is triggered, including logging the event, notifying operators, and initiating system shutdown procedures.

3.3.5 Applications

The GFC is widely used in various safety-critical applications to ensure the safe operation of industrial systems. Examples include:

Manufacturing Automation: In automated manufacturing environments, the GFC is used to halt machinery and robotic systems in response to detected faults, preventing accidents and equipment damage.

Process Control: In process industries, such as chemical and petrochemical plants, the GFC ensures that critical processes are safely shut down in the event of an emergency, protecting personnel and the environment.

Automotive Safety Systems: In automotive applications, the GFC is employed to bring the vehicle to a safe state in the event of a critical system failure, enhancing passenger safety.

Medical Devices: In medical equipment, the GFC ensures that devices such as infusion pumps and surgical robots can be safely stopped to protect patients during critical failures.

4 | Results and Analysis

The following picture shows the setup used to implement some Can Open communication functional safety scenarios

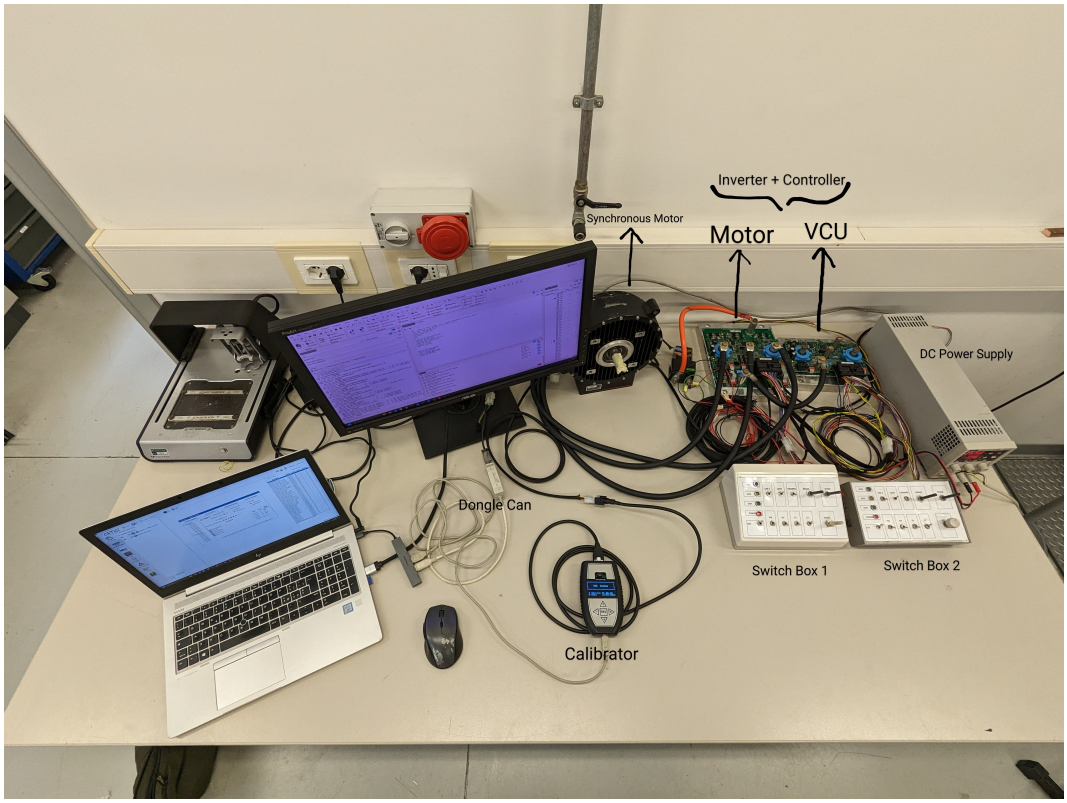

Figure 4.1: Setup

in particular one highlights

- **first inverter-controller device**: it is assigned to *NODE-ID 5*, and emulates the *VCU*; the Vehicle Control Unit is responsible for managing the communication and control of various vehicle systems, including the electric motor controller, ensuring safe and efficient operation. It has a crucial role in the overall functional safety of the process operation by coordinating inputs and outputs from various sensors and actuators, executing safety protocols, and handling communication over the CANopen network.
- **second inverter-controller device**: it is assigned to *NODE-ID 1*, and drives the *synchronous motor*, through FOC-based control algorithms. Such synchronous motor with superficial poles features permanent magnets mounted on the rotor's surface. This design allows for high efficiency, precise control, and a constant magnetic field, which helps in maintaining synchronous speed with the stator's rotating magnetic field.
- **calibrator**: it is designed for precise parameter setup, adjustment, and validation of motor controllers. It allows for the input and fine-tuning of parameters such

as speed and torque limits, in order to have the sought optimal performance. The device tests the controller's responses to various conditions, providing detailed diagnostic feedback and error codes to identify and resolve issues. Additionally, it verifies correct communication within the CANopen network, ensuring data integrity and protocol adherence. Mainly, it ensures the motor controller to operate safely and efficiently.

- **donge CAN**: it serves as an interface for connecting a computer or diagnostic tool to the CAN bus network. It facilitates communication by converting CAN signals to USB or another protocol compatible with the diagnostic tool. The dongle allows real-time monitoring, data logging, and analysis of the CAN network traffic. It also supports firmware updates and parameter adjustments for devices on the network. By ensuring proper communication, the dongle CAN device is then important for diagnosing, configuring, and maintaining the electric motor controller and its associated components.
- **switch box**: it is used to control the power supply and signal routing to the motor controller and other con-

nected components. It allows for the manual switching of power circuits, enabling or disabling power to the motor or other devices as needed. The switch box also provides protection by incorporating circuit breakers or fuses to prevent overloads and short circuits. Additionally, it can route control signals, facilitating the safe and efficient operation of the motor controller within the CANopen network.

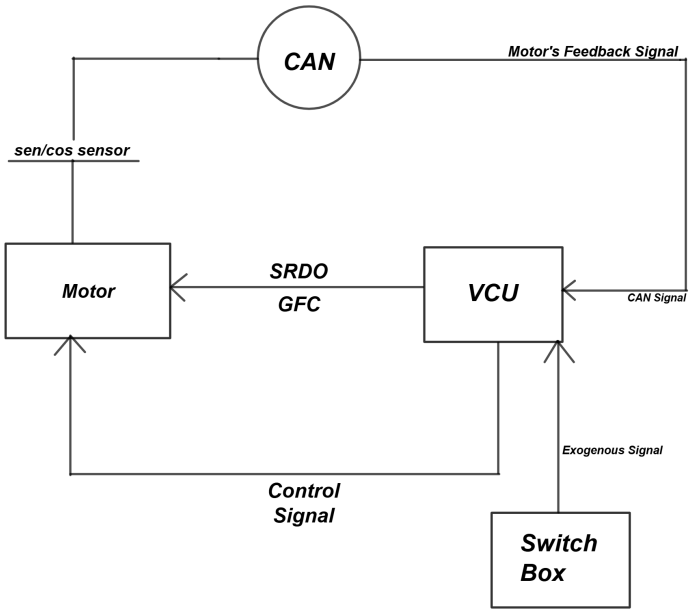

Figure 4.2: Setup scheme

For what concerns the **power supply**, the combination of a bench DC power supply and 2 batteries was used

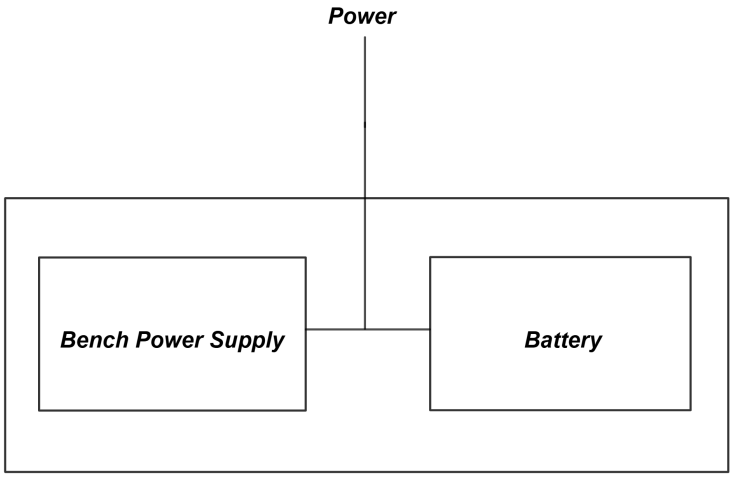

Figure 4.3: Power Supply scheme

this configuration allows the current generated by the motor, when working as generator, to not run into the power supply and possibly ruin it, but instead to flow entirely into the batteries so to charge them.

The **SRLD** device is in charge of maintaining the safety and integrity of the motor system. Its primary responsibility is to *monitor the system* for any fault conditions or unsafe states. When such conditions are detected, the SRLD will *open the line contactor* connected to the motor. This action effectively *disconnects the motor* from its power source, bringing it to a safe state and preventing potential damage or hazardous situations.

The SRLD continuously *assesses the operational parameters* of the motor and its controller, ensuring they remain within predefined safety limits. It also ensures that any commands to operate the motor are safe and do not compromise the system's integrity. By opening the line contactor, the SRLD can rapidly *shut down the motor* if necessary, thus protecting both the motor and its surroundings.

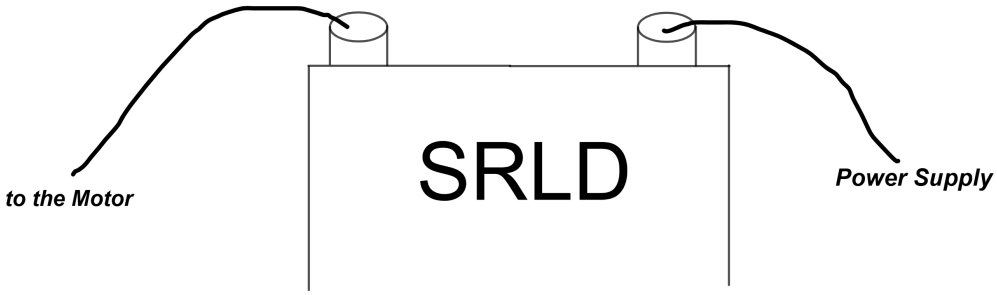

Figure 4.4: SRLD

Furthermore, the SRLD is integrated into the CANopen network, allowing it to *communicate with other safety devices and controllers* within the system. This communication capability guarantees coordinated responses to faults

and enhances the overall safety of the electric motor control system.

4.1 Scope

The following points identify the objective of this work

1. GFC propagation and management
    - **GFC transmit**: the signal is transmitted by the controller to the VCU, as soon as $\omega_{mot} \geq 905[rpm]$
    - **GFC receive**: the signal is transmitted by the VCU to the controller, as consequence of some hypothetical intern logic condition to the VCU
2. SRDO implementation
    - **SRVT timing**: implementation and testing of the scenario of SRVT timer expiration
    - **SCT timing**: same experiment, for SCT timer expiration scenario
    - **COB-ID 1 inconsistency**: implementation and testing the condition of bitwise inconsistency in the transmission of COB-ID 1
    - **COB-ID 2 inconsistency**: same implementa-

tion, but now in the condition of bitwise inconsistency in the transmission of COB-ID 2

- **COB-ID 1** & **COB-ID 2 inconsistency**: case where the two IDs are inconsistent with respect to each other

4.2 Experimental Implementation

Each one of these 7 scenarios has been realized and analyzed through the experimental setup described above.

4.2.1 GFC Transmit

In the first experiment, the GFC signal is transmitted from the controller to the VCU when the motor speed ω_{mot} reaches or exceeds 905 rpm, with the primary goal of bringing the motor to a safe state.
The collected data in the plot consists of two subplots: the top subplot displays motor speed and contactor state, while the bottom subplot shows motor current and fault code. In the top subplot, the motor speed gradually increases, closely following the speed setpoint. When the motor speed reaches the critical threshold of 905 rpm at around 20,000

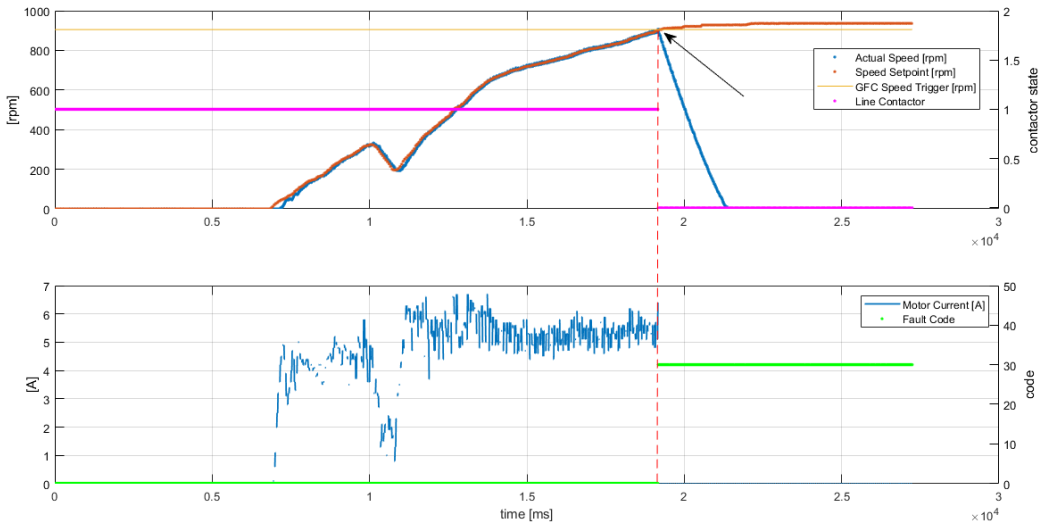

Figure 4.5: Experiment 1: GFC Transmit

ms, the GFC signal is triggered, prompting a change in the contactor state. This action is necessary to bring the motor to a safe state, preventing potential damage or hazardous conditions. As a result, the motor speed rapidly decreases, indicating that the motor has been successfully stopped. In the bottom subplot, the motor current initially shows fluctuations as the speed increases but stabilizes as the motor reaches a steady speed. Upon triggering the GFC signal at 905 rpm, the motor current drops significantly, corresponding with the motor stopping. At the same time, the fault code (**code 30**) becomes active, highlighting a fault or stop condition due to the GFC trigger. This experiment

effectively demonstrates the necessity and functionality of the GFC signal in ensuring the motor is brought to a safe state when operating conditions exceed safe limits.

4.2.2 GFC Receive

The GFC signal is now transmitted by the VCU to the controller due to an internal logic condition within the VCU. The purpose of this experiment is to demonstrate the motor's transition to a safe state as dictated by the VCU.

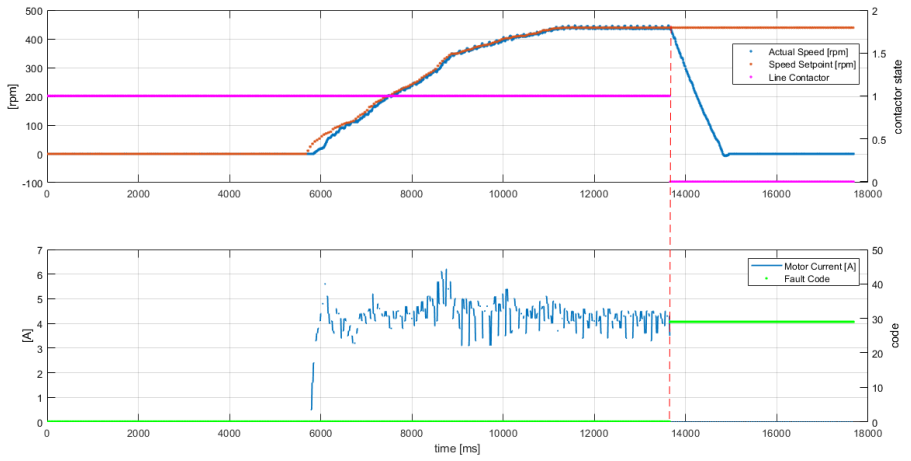

Figure 4.6: Experiment 2: GFC Receive

Initially, the motor speed follows the speed setpoint closely, increasing steadily. Around 20,000 ms, an internal logic condition in the VCU triggers the GFC signal, which is then transmitted to the controller. This transmission is

promped in order to bring the motor to a safe state, ensuring it does not operate beyond safe limits. At receiving the GFC signal, the contactor state changes, leading to a rapid decrease in motor speed. This indicates that the motor is being brought to a stop, thus showing the system's ability to ensure safety through coordinated communication between the VCU and the controller. Simultaneously, the motor current, which initially fluctuates and then stabilizes as the speed increases, drops significantly when the GFC signal is received. This drop in current corresponds with the motor stopping, ensuring that it reaches a safe state. The activation of the fault code (**code 29**) highlights the system's response to the VCU's internal logic condition, marking a fault or stop condition initiated by the GFC signal.

4.2.3 SRVT Timing

In this experiment the SRVT (Safety-Related Validation Time) timing is concerned, by implementing and testing the scenario of SRVT timer expiration. The SRVT is used to validate the temporal integrity of a single SRDO transmission. It ensures that the two consecutive frames making

up an SRDO are received within a predefined time window.

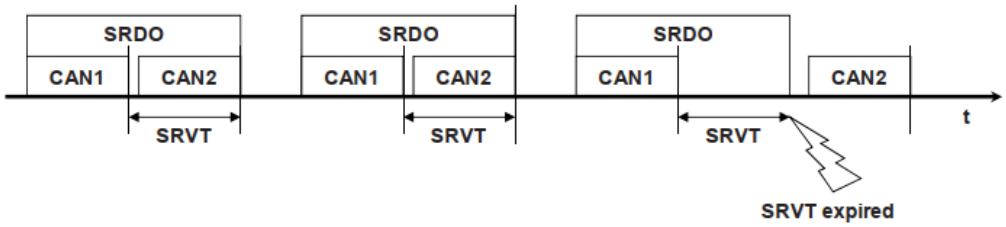

Figure 4.7: SRVT Timing

At the begin, the motor speed closely follows the speed setpoint, and the system operates normally. As the SRVT timer is running, it monitors the time between the reception of the two parts of each SRDO. When the timer expires, it indicates that the expected time window has been exceeded without receiving the complete SRDO (CAN2 frame has been deleted through the switch box). Since the SRDO is transmitted cyclically (not event-driven), indeed as the next cycle has occurred and the COB-ID 2 frame is not detected for a time $\geq t_{SRVT}$, a safety response is triggered.

After SRVT timer expiration, the line contactor changes state, resulting in a rapid decrease in motor speed. This action is carried out in order to bring the motor to a safe state, preventing any potential hazards due to the missed or

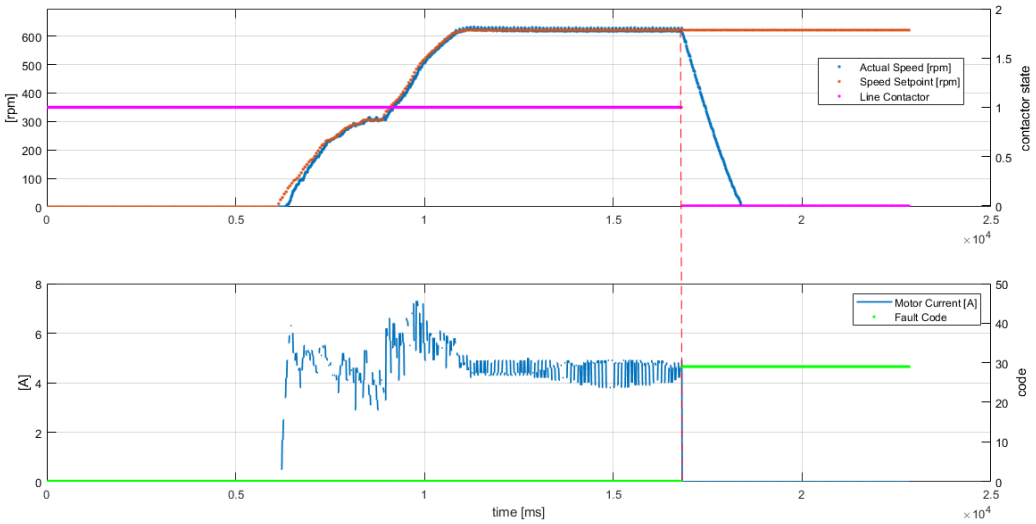

Figure 4.8: Experiment 3: SRVT Inconsistency

delayed safety-related data. Concurrently, the motor current drops significantly, and the fault code (**code 29**) becomes active, indicating a fault condition due to the SRVT timer expiration. This experiment shows the importance of timely and complete reception of safety-related data to maintain the motor's operational safety and integrity.

4.2.4 SCT Timing

Here we examine the SCT (Safety Cycle Time) timing by implementing and testing the scenario of SCT timer expiration. The SCT is used to monitor the cyclic transmission rate of SRDOs (Safety-Related Data Objects). If an ex-

pected SRDO is not received within the SCT interval, it is assumed that a fault has occurred, and the system takes safety actions.

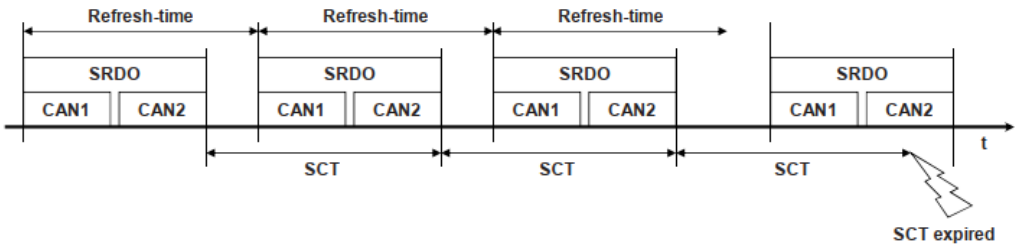

Figure 4.9: SCT Timing

As the SCT timer runs, it monitors the interval between SRDO transmissions. If the timer expires without receiving the expected SRDO, it indicates a fault condition. This is reflected in the plot when the SCT timer expires at around 20,000 ms.

In the moment of the SCT timer expiration (now the CAN1 frame was removed with the switch box), the line contactor state changes, resulting in a rapid decrease in motor speed. This action ensures the motor is brought to a safe state, preventing any potential hazards due to the missed safety-related data. Simultaneously, the motor current drops significantly, indicating that the motor has been stopped. The fault code becomes active, highlighting a fault condition due to the SCT timer expiration. This shows how the

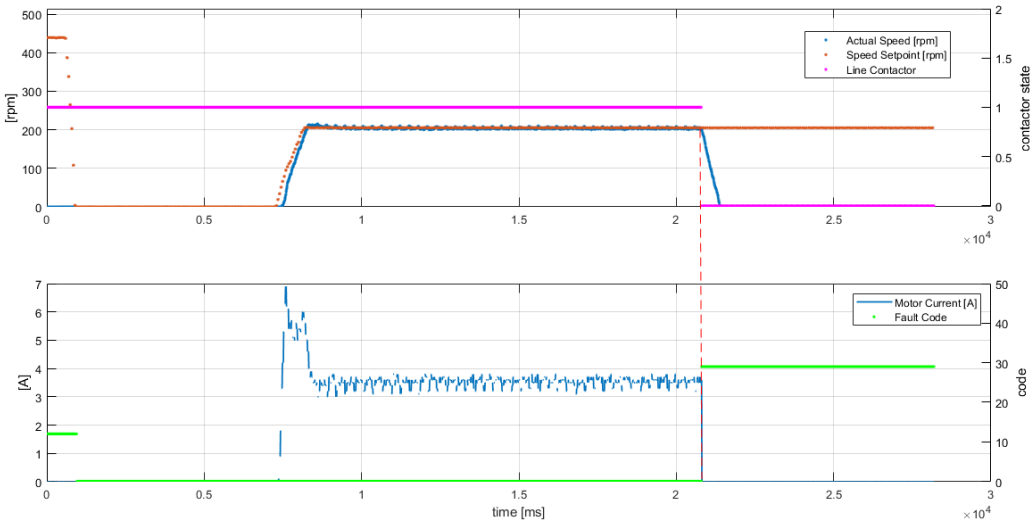

Figure 4.10: Experiment 4: SCT Inconsistency

SCT timer ensures that any delays or interruptions in data transmission trigger appropriate safety measures, bringing the motor to a safe state and preventing potential hazards.

4.2.5 COB-ID 1 Inconsistency

This experiment replicated the bitwise inconsistency in the transmission of COB-ID 1. Such consistency is crucial for ensuring data integrity and safety in the CANopen network, specifically within the SRDO framework. In the CANopen network, each SRDO consists of two CAN data frames: the first carries the actual safety-related data, and the second contains the bitwise inverted version of the data

in the first frame. This dual-frame structure allows for integrity checks during transmission. If the second frame does not exactly invert the first, an error is detected, indicating a potential safety issue.

7,6253	1	Rx	SRDO_COBID1	4	68 02 97 FD
7,6348	1	Rx	TxPDO1_Node1	8	68 02 00 00 00 01 30 00
7,6351	1	Rx	SRDO_COBID2	4	68 02 97 FD
7,6448	1	Rx	TxPDO1_Node1	8	6A 02 00 00 00 01 30 00
7,6548	1	Rx	TxPDO1_Node1	8	67 02 00 00 00 01 30 00
7,6648	1	Rx	TxPDO1_Node1	8	6B 02 00 00 00 01 2E 00
7,6749	1	Rx	TxPDO1_Node1	8	71 02 00 00 00 01 2E 00
7,6753	1	Rx	SRDO_COBID1	4	68 02 68 02
7,6760	1	Rx	GFC	1	00
7,6765	1	Rx	EMCY_1	8	00 80 10 1D 1C 00 00 00
7,6848	1	Rx	TxPDO1_Node1	8	6B 02 1D 1C 00 00 2E 00
7,6851	1	Rx	SRDO_COBID2	4	68 02 97 FD
7,6863	1	Rx	EMCY_1	8	00 80 10 1D 1E 00 00 00
7,6947	1	Rx	TxPDO1_Node1	8	63 02 1D 1E 00 00 00 00
7,7048	1	Rx	TxPDO1_Node1	8	5E 02 1D 1E 00 00 00 00
7,7104	1	Rx	NodeHeartbeat_1	1	05
7,7148	1	Rx	TxPDO1_Node1	8	58 02 1D 1E 00 00 00 00
7,7248	1	Rx	TxPDO1_Node1	8	55 02 1D 1E 00 00 00 00
7,7251	1	Rx	SRDO_COBID1	4	68 02 68 02
7,7348	1	Rx	SRDO_COBID2	4	68 02 97 FD

Figure 4.11: Experiment 5: Trace Log

The trace log shows that, at some point, a discrepancy in COB-ID 1 is detected, and consequently the second frame does not correctly invert the first frame. This inconsistency triggers a safety response. Note that such inconsistency has been generated through the switch box, which simulates the propagation of an electromagnetic impulse which modifies the bit wise informative content of the COB-ID 1 frame.

At the detection of the COB-ID 1 inconsistency, the line

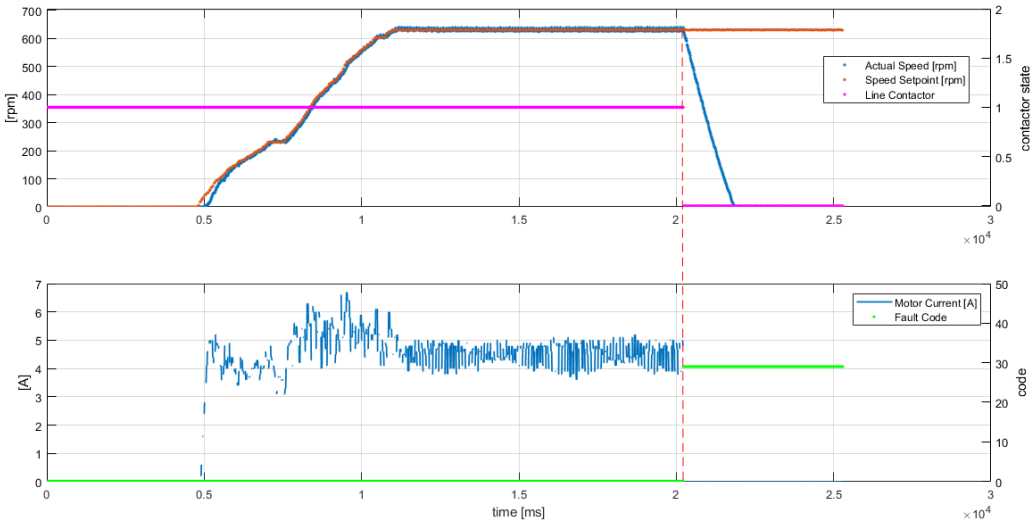

Figure 4.12: Experiment 5: COB-ID 1 Inconsistency

contactor state changes, causing a rapid decrease in motor speed to bring the motor to a safe state. Simultaneously, the motor current drops significantly, and the fault code becomes active, indicating a fault condition due to the detected inconsistency. This experiment highlights the importance of bitwise consistency in COB-ID transmissions for maintaining system safety, in view of the system's ability to detect errors in SRDO transmissions and take appropriate safety measures to prevent potential hazards, ensuring the motor transitions to a safe state promptly.

4.2.6 COB-ID 2 inconsistency

The sixth data collection replicates the bitwise inconsistency, but in the transmission of COB-ID 2.

15,0377	1	Rx	SRDO_COBID1	4	45 02 BA FD
15,0400	1	Rx	TxPDO1_Node1	8	43 02 00 00 00 01 2D 00
15,0477	1	Rx	SRDO_COBID2	4	45 02 BA FD
15,0500	1	Rx	TxPDO1_Node1	8	43 02 00 00 00 01 2D 00
15,0601	1	Rx	TxPDO1_Node1	8	43 02 00 00 00 01 30 00
15,0700	1	Rx	TxPDO1_Node1	8	41 02 00 00 00 01 30 00
15,0802	1	Rx	TxPDO1_Node1	8	4A 02 00 00 00 01 30 00
15,0877	1	Rx	SRDO_COBID1	4	45 02 BA FD
15,0900	1	Rx	TxPDO1_Node1	8	4C 02 00 00 00 01 2A 00
15,0976	1	Rx	SRDO_COBID2	4	45 02 45 02
15,0981	1	Rx	GFC	1	00
15,0986	1	Rx	EMCY_1	8	00 80 10 1D 1E 00 00 00
15,0999	1	Rx	TxPDO1_Node1	8	46 02 1D 1E 00 00 2A 00
15,1100	1	Rx	TxPDO1_Node1	8	3F 02 1D 1E 00 00 00 00
15,1200	1	Rx	TxPDO1_Node1	8	3F 02 1D 1E 00 00 00 00
15,1301	1	Rx	TxPDO1_Node1	8	36 02 1D 1E 00 00 00 00
15,1377	1	Rx	SRDO_COBID1	4	45 02 BA FD
15,1405	1	Rx	TxPDO1_Node1	8	33 02 1D 1E 00 00 00 00
15,1476	1	Rx	SRDO_COBID2	4	45 02 45 02

Figure 4.13: Experiment 6: Trace Log

The trace log reveals that at a specific moment, a discrepancy in COB-ID 2 is detected, leading to the second frame failing to correctly invert the first frame. This inconsistency initiates a safety response. Again, this inconsistency was introduced using the switch box, which simulates the effect of an electromagnetic impulse altering the bit wise informational content of the COB-ID 2 frame.

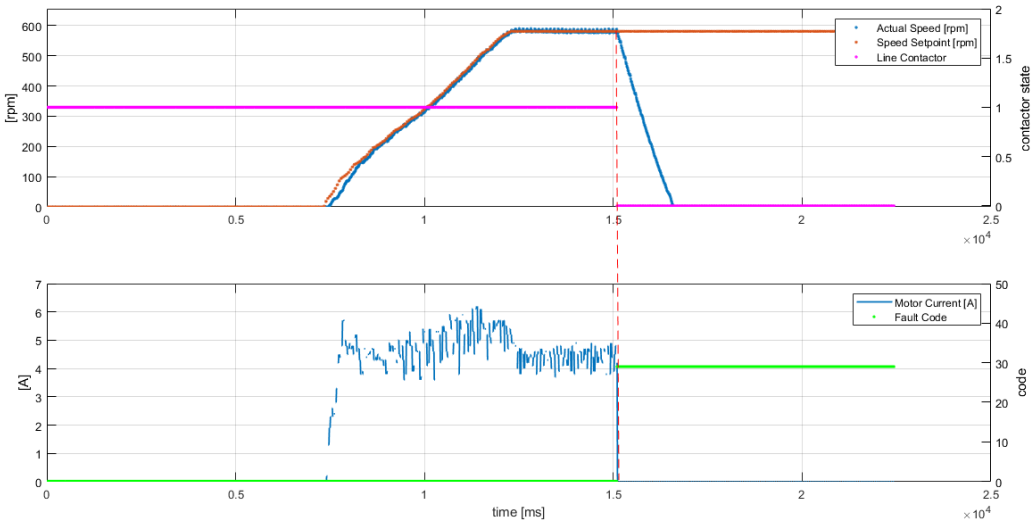

Figure 4.14: Experiment 6: COB-ID 2 Inconsistency

The trajectory of the quantities of interest follows the one described at the previous experiment.

4.2.7 COB-ID 1 & COB-ID 2 inconsistency

In this final experiment, the condition where both COB-ID 1 and COB-ID 2 are inconsistent with respect to each other is considered.

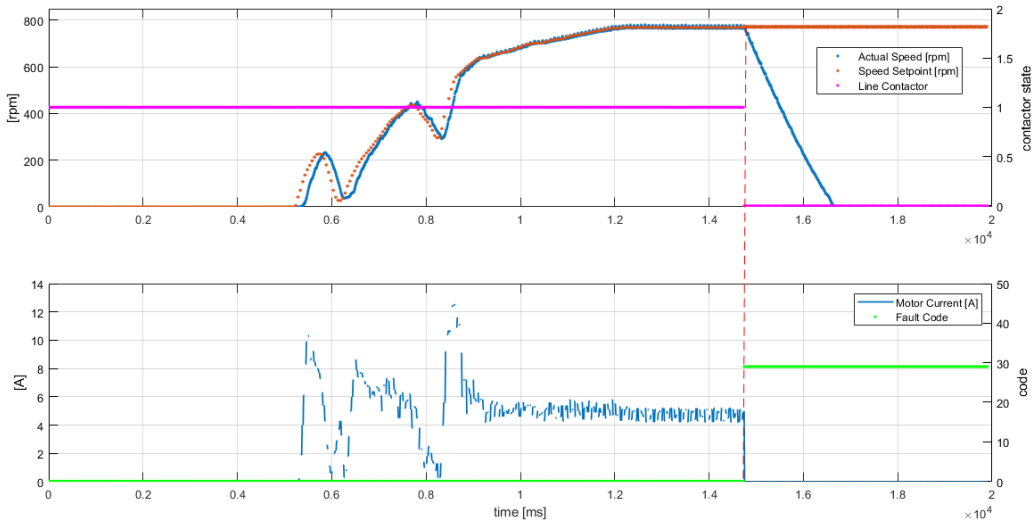

Figure 4.15: Experiment 7: COB-ID 1 COB-ID 2 Inconsistency

The trace log shows that both COB-ID 1 and COB-ID 2 become inconsistent with each other. The highlighted entries in the trace log indicate the moment when the inconsistency is detected, leading to a mismatch between the expected bit wise inversion of the two frames. This inconsistency triggers a safety response.

14,6798	1	Rx	SRDO_COBID1	4	01 03 FE FC
14,6839	1	Rx	TxPDO1_Node1	8	FD 02 00 00 00 01 2C 00
14,6894	1	Rx	SRDO_COBID2	4	01 03 FE FC
14,6934	1	Rx	TxPDO1_Node1	8	00 03 00 00 00 01 2C 00
14,7035	1	Rx	TxPDO1_Node1	8	FD 02 00 00 00 01 2C 00
14,7137	1	Rx	TxPDO1_Node1	8	FD 02 00 00 00 01 31 00
14,7235	1	Rx	TxPDO1_Node1	8	FF 02 00 00 00 01 31 00
14,7295	1	Rx	SRDO_COBID1	4	01 03 FE FC
14,7333	1	Rx	TxPDO1_Node1	8	FD 02 00 00 00 01 31 00
14,7395	1	Rx	SRDO_COBID2	4	06 03 F9 FC
14,7398	1	Rx	GFC	1	00
14,7408	1	Rx	EMCY_1	8	00 80 10 1D 1E 00 00 00
14,7435	1	Rx	TxPDO1_Node1	8	08 03 1D 1E 00 00 32 00
14,7533	1	Rx	TxPDO1_Node1	8	05 03 1D 1E 00 00 00 00
14,7634	1	Rx	TxPDO1_Node1	8	FA 02 1D 1E 00 00 00 00
14,7733	1	Rx	TxPDO1_Node1	8	F7 02 1D 1E 00 00 00 00
14,7795	1	Rx	SRDO_COBID1	4	01 03 FE FC
14,7833	1	Rx	TxPDO1_Node1	8	F0 02 1D 1E 00 00 00 00
14,7897	1	Rx	SRDO_COBID2	4	06 03 F9 FC

Figure 4.16: Experiment 7: Trace Log

This last experiment concurs to outline the necessity for maintaining consistent data contents, for the correct development of the process within the network; in particular the coherency between COB-ID 1 and COB-ID 2 in SRDO transmissions to ensure system safety. In so doing then the system can effectively detect errors and inconsistencies in these transmissions and take the necessary safety actions to prevent potential hazards: ensuring that, in this case, the motor is quickly brought to a safe state.

5 | Conclusion

This work has highlighted the critical role of functional safety mechanisms in CANopen communication for electric motor controllers. Through a series of experiments, we demonstrated the effectiveness of Safety-Related Data Objects (SRDOs), Global Fail-Safe Commands (GFC), and error detection protocols in maintaining system safety and integrity.

The experiments underscored the importance of bitwise consistency between COB-ID 1 and COB-ID 2 in SRDO transmissions. Any discrepancy between these data frames was promptly detected by the system, triggering safety responses to prevent potential hazards. This capability ensures that safety-critical data is transmitted reliably, safeguarding the operation of electric motor controllers in various applications.

Furthermore, the necessity of timely and complete reception of safety-related data was evident in the experiments

involving SRVT (Safety-Related Validation Time) and SCT (Safety Cycle Time) timers. These experiments showed that exceeding predefined time windows for data reception leads to immediate safety actions, such as stopping the motor to prevent unsafe conditions.

The role of the switch box in simulating faults and electromagnetic interference was crucial for testing the robustness of the CANopen safety protocols. The generated inconsistencies in COB-ID frames demonstrated the system's ability to detect and respond to data corruption, ensuring the motor transitions to a safe state.

In conclusion, the experiments validated the effectiveness of CANopen's functional safety mechanisms in real-world scenarios. Maintaining bitwise consistency in SRDO transmissions, ensuring timely data reception, and implementing robust error detection protocols are essential for achieving reliable and safe operation of electric motor controllers.

Bibliography

[1] CAN in Automation (CiA) e. V. (2011) *CANopen Application Layer and Communication Profile*, Version 4.2.0, February. `https://www.can-cia.org/`.

[2] BSI Standards Publication. (2019) *Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 1: General requirements*, BS EN 50325-1:2019, December 2019.

[3] BSI Standards Publication. (2001) *Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 2: DeviceNet and SDS*, BS EN 50325-2:2001.

[4] BSI Standards Publication. (2001) *Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 3: WorldFIP*, BS EN 50325-3:2001.

[5] BSI Standards Publication. (2002) *Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 4: CANopen*, BS EN 50325-4:2002.

[6] BSI Standards Publication. (2010) *Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 5: Functional safety communication based on EN 50325-4*, BS EN 50325-5:2010.