UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

CAN FD: a communication network for future avionic systems

João de Sousa Alves

Mestrado em Engenharia Informática

Arquitetura, Sistemas e Redes de Computadores

Versão Provisória

Dissertação orientada por:

Professor Doutor António Casimiro

2019

2019

Agradecimentos

Em homenagem ao Professor Doutor José Rufino, sem o qual a realização deste trabalho não teria sido possível.

Um agradecimento especial ao Professor Doutor António Casimiro que aceitou o desafio de me acompanhar e orientar rumo à conclusão desta dissertação.

Dedicado à Ingrid.

Resumo

A presente dissertação incide sobre o protocolo de tempo real apelidado Controller Area Network with Flexible Data Rate, comumente designado como CAN FD, que opera ao nível da camada de Ligação de Dados do modelo OSI. Este protocolo é uma versão revista e melhorada (essencialmente em aspetos que envolvem a sua eficiência) do protocolo Controller Area Network (CAN), lançado em 1986 por uma equipa de engenheiros da Robert Bosch GmbH.

O desenvolvimento do protocolo CAN FD tem por base dois grandes objetivos: Em primeiro lugar, aumentar a velocidade de transmissão de dados na rede (de 1 para 15 Mbps); em segundo, permitir o envio de mensagens com um campo de dados superior ao do seu antecessor (de 8 para 64 bytes). No entanto, para conseguir implementar estas melhorias foi necessário introduzir alterações significativas à estrutura e funcionamento do protocolo original. Estas alterações tiveram vários tipos de impacto no protocolo, desde o tempo de transmissão de mensagens à forma como identifica e lida com erros. Isto faz com que a vasta literatura científica que atualmente acompanha o CAN (e que contribuiu grandemente para a confiança que existe em torno deste protocolo) não tenha necessariamente aplicabilidade direta ao CAN FD. Sobre esta perspetiva, a questão central passa por entender o impacto não óbvio que as alterações efetuadas têm, e o caminho para o conseguir determinar envolve, numa primeira instância, a reprodução de alguns dos estudos efetuados sobre o CAN.

O Controller Area Network, enquanto rede de tempo real, conheceu diversas aplicações, desde a indústria automóvel, para a qual foi inicialmente desenhado, até à engenharia aeroespacial e ao controlo industrial. Se tivermos de escolher um fator comum a muitos dos sistemas onde usualmente o CAN é implementado, poderíamos escolher a sua criticidade. Se um dos sistemas de suporte ao voo de um avião falhar, este pode afetar negativamente o piloto ou outros sistemas, induzindo-os em erro ou limitando as suas capacidades. Da mesma forma, uma falha numa linha de montagem industrial pode ter consequências desastrosas para todos os envolvidos. Neste sentido, é importante saber como os sistemas detetam e tratam erros e deve-se estudar o seu comportamento na presença e ausência destes erros. Se o máximo delay é o pior cenário de transmissão na ausência de erros para um protocolo como o CAN, o máximo delay somado ao tempo de identificação e recuperação de um erro dá-nos o pior cenário de transmissão na presença de um erro. Em sistemas críticos é esta última abordagem que devemos utilizar e acautelar. O conceito de inacessibilidade representa o tempo que uma rede de transmissão de dados está inoperacional (a recuperar de um erro) e mede-se desde o início da transmissão de uma

mensagem, onde é identificado um erro, até ao retorno do protocolo ao normal funcionamento. No CAN, foi efetuado um estudo sobre os diversos tempos de inacessibilidade em função do tipo de erro que o protocolo poderia experienciar. No CAN FD, esse estudo está ainda por realizar e é esse o primeiro objetivo deste trabalho.

Como foi referido, as melhorias introduzidas pelo CAN FD obrigaram a certas readaptações no formato das tramas e em alguns mecanismos nativos do protocolo. Após um levantamento exaustivo destas modificações, conseguimos determinar qual o impacto direto que elas tiveram no tempo de transmissão de uma mensagem. Consequentemente, e tendo especial atenção a pequenas alterações introduzidas nos mecanismos de deteção e sinalização de erros do CAN FD, foi possível observar como este protocolo responde perante a existência de erros durante a transmissão de uma mensagem, derivando fórmulas matemáticas simples que permitem descrever o seu comportamento. Por sua vez, estas fórmulas matemáticas possibilitou-nos colocar um valor real sobre o tempo de inacessibilidade da rede quando esta é afetada por cada um dos tipos de erro definidos na especificação do protocolo. O resultado final deste estudo permitiu-nos fazer uma análise comparativa com os resultados de um outro, efetuado sobre o CAN original, uma vez que ambos partem dos mesmos pressupostos iniciais.

A simplicidade e "elegância" do CAN original sempre foram um fator de distinção relativamente a outros protocolos. Contudo, o processo de desenvolvimento do CAN FD trouxe consigo uma camada de complexidade adicional. Em alguns casos, as opções tomadas pela equipa de engenheiros responsável pelo desenho do CAN FD poderão ser vistas como polémicas ou discutíveis, muito devido à escassez de argumentos apresentados ou à ausência de estudos que as sustentem. Atentos a este facto, tomámos a decisão de mergulhar na cronologia de acontecimentos que levaram a cada uma das modificações observadas no CAN FD, com o intuito de nos pronunciarmos sobre se os meios utilizados realmente justificam os fins a que se propõe. A análise crítica e detalhada das modificações impostas pelo CAN FD é o segundo objetivo deste trabalho.

A pesquisa exaustiva das razões que antecederam algumas das alterações adotadas aquando do desenvolvimento do CAN FD levou-nos ao encontro de casos de escape em que a possibilidade de ocorrência de determinados erros mina toda a robustez anunciada do protocolo. Neste documento fica latente que a correção deste tipo de situações foi uma das prioridades da equipa por detrás do CAN FD, não tivesse a sua correção o impacto que teve no protocolo. Curiosamente, durante este processo de indagação acabámos por tropeçar num novo tipo de erro que afeta todas as versões do protocolo (CAN e CAN FD). As condições necessárias para a sua emergência e a forma como este afeta as transmissões de dados envolvem a conjugação de vários fatores, o que, hipoteticamente, diminui a probabilidade de observarmos a sua ocorrência num cenário de implementação real. Não obstante, existe validade teórica que suporta a sua existência e, como tal, deve ser identificado e comunicado. A explicação de como este erro ocorre e de quais as consequências que pode encerrar é o último tema abordado nesta dissertação.

Palavras-chave: Tempo real, Sistemas embebidos, CAN, CAN FD, Inacessibilidade, Robustez, Deteção de erros, Tratamento de erros, Recuperação de erros.

Abstract

The present dissertation focuses on the real-time protocol Controller Area Network with Flexible Data Rate, commonly referred to as CAN FD, which operates at the level of the Data Link layer of the OSI model. This protocol is a revised and improved version (essentially in terms of efficiency) of the Controller Area Network (CAN) protocol, launched in 1986 by a team of engineers at Robert Bosch GmbH.

The development of the CAN FD protocol is based on two main objectives: First, to increase the speed of data transmission in the network (from 1 to 15 Mbps); second, to allow messages to be sent with a data field higher than its predecessor (from 8 to 64 bytes). However, in order to implement these improvements it was necessary to introduce significant changes to the structure and operation of the original protocol. These changes had several types of impact on the protocol, from the time of message transmission to the way it identifies and handles errors. This makes the vast scientific literature that currently accompanies CAN (and which contributed greatly to the confidence that exists around this protocol) does not necessarily have direct applicability to the CAN FD. On this perspective, the central issue is to understand the non-obvious impact that the modifications have, and the way to determine it involves, in a first instance, the reproduction of some of the studies carried out on CAN.

The Controller Area Network, as a real-time network, has seen a variety of applications, from the automotive industry to which it was initially designed, to aerospace engineering and industrial control. If we have to choose a factor common to many of the systems where CAN is usually implemented, we could choose its criticality. If one of the flight support systems of an airplane fails, it may adversely affect the pilot or other systems, misleading them or limiting their capabilities. Likewise, a failure in an industrial assembly line can have disastrous consequences to everyone involved. In this sense, it is important to know how systems detect and treat errors and to study their behavior in the presence and absence of these errors. If the maximum delay is the worst transmission scenario in the absence of errors for a protocol like CAN, the maximum delay added to the time for identification and recovery of an error gives us the worse transmission scenario in the presence of an error. In critical systems is this last approach that we must use and caution. The concept of inaccessibility represents the time that a data transmission network is inoperative (to recover from an error) and is measured from the beginning of the transmission of a message, where an error is identified, until the return of the protocol to normal operation. In the CAN, a study was performed on the various inaccessibility times due to the type of error that the

protocol could experience. In CAN FD, this study is still to be carried out and is the first objective of this work.

As mentioned, the improvements introduced by CAN FD have forced some adaptations in the format of the frames and in some native mechanisms of the protocol. After an exhaustive survey of these changes, we were able to determine the direct impact they had on the transmission time of a message. Consequently, and with special attention to small changes introduced in the detection and signaling mechanisms of the CAN FD protocol, it was possible to observe how it responds to the existence of errors during the transmission of a message, deriving simple mathematical formulas that allow the description of its behavior. In turn, these mathematical formulas permitted to place a real value on the network inaccessibility time when it is affected by each one of the error types defined in the protocol specification. The final result of this study allowed us to make a comparative analysis with the results of another one, carried out on the original CAN, since both start from the same initial assumptions.

The simplicity and "elegance" of the original CAN protocol has always been a distinguishing factor compared to others. However, the CAN FD development process has brought with it a layer of additional complexity. In some cases, the options taken by the team of engineers responsible for the design of the CAN FD may be seen as controversial or debatable due to the lack of arguments presented or to the lack of studies that support them. With this in mind, we decided to investigate the events that led to each of the modifications observed in CAN FD, in order to decide whether the means really justify the purposes. The critical and detailed analysis of the modifications imposed by the CAN FD protocol is the second objective of this work.

The exhaustive research of the reasons that preceded some of the changes adopted during the development of the CAN FD led us to find escape cases in which the possibility of occurrence of certain errors undermines all the announced robustness of the protocol. In this document it becomes clear that the correction of this type of situations was one of the priorities of the team behind CAN FD. Curiously, during this examination process we ended up stumbling over a new type of error that affects all versions of the protocol (CAN and CAN FD). The conditions required for its emergence and the way it affects data transmissions involve the combination of several factors, which, hypothetically, decreases the probability of observing its occurrence in a real implementation scenario. Nevertheless, there is theoretical validity that supports its existence and, as such, it must be identified and communicated. The explanation of how this error occurs and of what consequences it may contain is the last topic addressed in this work.

Keywords: Real-time, Embedded systems, CAN, CAN FD, Inaccessibility, Robustness, Error detection, Error handling, Error recovery.

Index

List of Figures

List of Tables

1. Introduction

1.1 Context

The present thesis presents, studies and evaluates the real-time network protocol – Controller Area Network with Flexible Data-Rate [2], commonly known as CAN FD, by critically analyzing some of its structural characteristics and its behavior under normal and abnormal operation conditions, focusing on determinant real-time networks properties like reliability and robustness, and others like performance. This work is motivated by the CAN FD release and attempts to shed light on yet uncovered aspects of this new protocol.

CAN FD was developed to improve some of the features in the original Controller Area Network [1] – a protocol developed in the year of 1986 by Bosch engineers and first implemented in 1988. Classical CAN was specifically designed to be used in the automotive industry. Its main function was to enable reliable real-time communication between car components. Later, it was found to be useful in many other domains, such as industrial control and aerospace engineering, due to its strong reliability and robustness properties, low implementation costs and relatively small implementation complexity. All this upsides contributed to a wide dissemination, which gave notoriety to the CAN protocol during many years. However, the same characteristics that seemed reasonable in 1980's – data rates up to 1 Mbps and data payload up to 8 bytes - are hardly sufficient to face the current market needs, where information volume and system performance are increasingly determining factors. Even though performance is not what defines a real-time system (but timeliness and predictability are), neglecting it may call into question its overall acceptance. For this reason, Bosch put together a team in 2011 with the task of creating a new protocol that enables the transmission of larger data payload messages, at a higher rate, while maintaining the advantages that made CAN a successful network protocol in the first place. In 2012, the specification of the new Controller Area Network with Flexible Data rate was released to the public [2]. This publication uncovered a protocol which can be 15 times faster than CAN, supports an 8 times larger payload while, allegedly, maintaining the same reliability and robustness. This work addresses, in part, the mechanisms utilized by CAN FD designers to guarantee the above properties, in order to test their efficiency and applicability.

1.2 Motivation

The Controller Area Network protocol has been the subject of a profound study over time. Since its foundation, various aspects of the protocol have been addressed and evaluated in a way that it has gained the overall acceptance and appreciation of the scientific community. All this work contributed for a vast understanding and knowledge of CAN network. Nowadays, with the development and release of CAN FD and considering the profound changes in fundamental operation characteristics to which the protocol was subject, the same issues that arose over the original protocol become once more relevant and must be addressed, in order to determine if the existent scientific research is valid and applicable to the new protocol or, on the contrary, must be revised.

In addition to this discovery process, there are pertinent questions that are worth answering, concerning the transformation process that culminated in the deployment of CAN FD and later in the ISO CAN FD:

- Is there a pertinent reason behind the changes made during the passage from CAN to CAN FD and do they all serve the intended purpose?
- Has the overall network performance improved substantially? What are the costs of this enhancement?
- Does CAN FD deliver the same reliability and robustness properties of its predecessor?

1.3 Objectives

A CAN FD network is exposed to a series of events that can lead to a malfunction in its normal operation. The origin of such errors may range from electromagnetic interference (EMI) to a failure in network components. These errors can have a permanent effect, effectively disabling all network communication (i.e. damaged media bus), or a transient one. In the second case, the network is only unable to operate during the time frame in which the error occurs, is treated and gets recovered from. This type of incident is common and can happen several times during network operation, depending on internal factors like the chosen physical media or external ones, such as the environment in which it is deployed. Even though some networks can operate under frequent and long communication glitches, in CAN FD real-time behavior is expected. This often means high predictability and timeliness and gets increasingly important if we aim at hard real-time implementations. As such, the periods in which the network is inoperative must be analytically studied and measured in a way that they can be accounted for when designing this type of systems. This type of study was done over the original CAN protocol. However, the new features introduced by CAN FD make the revision of this topic mandatory.

Additionally, a major concern of the Bosch CAN FD development team was to endow the new protocol with at least the same error detection capabilities than its predecessor, since the

enhancements made to the new version forced the redefinition of some of its core mechanisms and the alteration of the frame structure itself. A specific study on these changes and on their impact over the mechanisms that guarantee the frame's integrity is also under the scope of this work.

Given the above, the objectives of the work presented in this thesis are defined as follows:

- Determine the changes made to the CAN protocol, with special emphasis in the error detection and error handling mechanisms;
- Understand and modulate the behavior of CAN FD in the absence and presence of errors;
- Derive the CAN FD transmission times in the absence and presence of errors and analyze the differences with respect to CAN;
- Review and critically assess the groundwork that support each of the changes made to the CAN protocol.

1.4 Contribution

This thesis contributes to a better understanding of the CAN FD protocol in two distinct ways. First and foremost, the frame transmission times of CAN and CAN FD, in the absence of errors, are derived and calculated. This study, which focuses purely on efficiency, provides the bases for a more profound examination of the protocol behavior in the presence of errors, which is also part of this work. A subset of equations, one for each error type, supports the calculation of the time taken by a node in a CAN FD network to recover from an erroneous transmission and proceed with network operation. In sum, the first part of this thesis concentrates on the frame transmission time and on the error detection and recover mechanisms, comparing and evaluating the results against the same aspects of the previous protocol.

Secondly, the curtain that covers the reasons behind every transformation (made upon the passage from CAN to CAN FD and to its standardization) is raised and the fundaments that support them revealed. Here, a balancing exercise is made considering the benefits that these changes bring and the problems they carry. During this analysis we also encountered some inconsistencies in the work developed by Bosch engineers and the ISO task force that are here reported. Furthermore, the result of this second line of work is the discovery of a new fault type, which affects all versions of the protocol and was not, to this day, accounted for in related work.

1.5 Structure

The first part of this work presents CAN state of the art. Chapter 3 provides an overview over CAN FD structure and operation (determining its transmissions times) and identifies the differences between the two protocols. In chapter 4 we extend a study made over the original

CAN protocol to CAN FD, which considers network behavior in the presence of errors and, analytically, derive equations that measured the time periods in which the network is unable to operate (inaccessibility periods). In chapter 5 we make a detailed analysis of the changes to which the protocol was submitted to, giving a straightforward answer to how and why they were made and objectively identifying advantages, disadvantages and how they could be made differently. Finally, chapter 6 points out some challenges to be addressed as further work.

2. State of the Art: the CAN Protocol

The evaluation of CAN FD operation, in a more general sense, and of its new functionalities in relation to CAN is one of the purposes of this work. To accomplish this objective, it is essential to accurately describe the original protocol so that we can fully understand it before extending our analysis to the new version.

The Controller Area Network is a multicast, multi-master serial bus, with a deterministic collision resolution scheme. The transmission medium is composed by a twisted pair of wires. A node is a station that sends and receives messages through the bus. A message is composed by data and sent as a sequence of bits. Each bit is sequentially transmitted by setting the bus level (voltage on the wires) to one of two possible values: Dominant (logical 0) and Recessive (logical 1), which is also the state of an idle bus. Dominant values always overwrite recessive ones. This means that if a node sends a dominant bit to the bus and, at the same time, another node sends a recessive one, only the dominant state will be visible on the network. This characteristic is exploited during prioritization of messages – *arbitration* in CAN – to allow a non-destructive collision resolving scheme.

Bosch published several CAN specifications, being the last one CAN 2.0, which came out in 1991 [1]. It is composed by two parts: A and B, which respectively define a base frame format and an extended one, latter created to expand the upper limit of active nodes in a network at a certain point. For the sake of simplicity, the analysis of the CAN protocol will be divided into the following sections:

- Message transmission and codification;
- Message prioritization;
- Message validation;
- Fault confinement and node state.

2.1 Message transmission and codification

Application data is encapsulated in a CAN frame. We refer to CAN frames as messages. CAN frames respect a well-defined format, with several frame fields that allow receiving nodes to interpret the encapsulated application data and check the integrity of the received frame. A data frame can be transmitted in different formats. The latest specification of the CAN protocol [1], defines two distinct frame formats:

- CAN base format;
- CAN extended format.

The structural difference between the two consists of an augment in the arbitration field, from the base format to the extended one. Functionally, there is an important distinction, because it determines an augment in the available frame identifiers which, in turn, are uniquely used to identify every data frame in the network.

In addition to the transmission of application data, a node may also need to send CAN frames, for instance to signal an error condition such as the reception of a corrupted frame. Therefore, the following 4 types of CAN frames are defined:

- Data frames – to transfer data between nodes;
- Remote frames – to request the transmission of data by other nodes;
- Overload frames – to delay the transmission of a frame;
- Error frames – to signal the detection of an error.

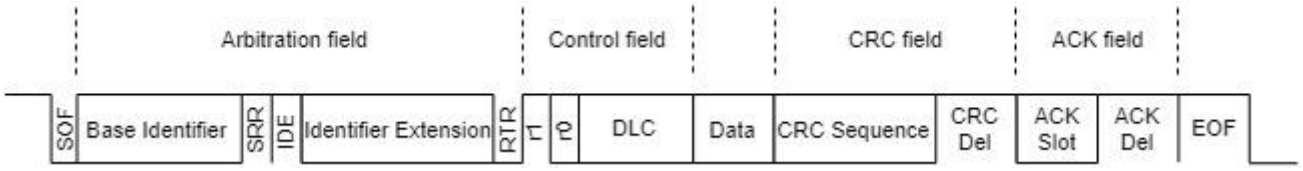

Figure 2.1 – CAN Data Frame (Extended Format)

A data frame is the message that carries the actual information. It encompasses the following fields in sequence:

Start of Frame (1 bit) – It is the first bit to be transmitted and marks the start of a new frame.

Arbitration Field (12 bits in base format and 32 in extended format) – It is composed by the Identifier, which is unique for every data frame and determines its priority in the network, and by the **RTR** (remote transmission request) bit, which differentiates a data frame from a remote frame. If the **RTR** bit is transmitted dominant, the receiver will interpret it as a data frame. Otherwise, it will be considered a remote frame. Furthermore, the extended frame format comprises a longer Identifier field (29 bits instead of 11) and two additional bits – The **SRR** (substitute remote request) and the **IDE** (identifier extension) bits – which, respectively, substitute the **RTR** bit in the base format and indicate the format in which the frame is being transmitted. The values are dominant for standard format and recessive for the extended one.

Control Field (6 bits) – The control field contains the **DLC** (data length code) and two independent bits. The **DLC** is 4 bits long and indicates the total length of the frame's data field. The meaning of the two other bits differs from one format to the other. In the extended format both bits are reserved for further expansion and, as such, not used by the protocol. However, in the base format one bit is a reserved bit while the other one is the **IDE** bit. Notice that this is the same bit than the one presented above (in the arbitration field) for the extended format.

Data Field (up to 64 bits) – It is the actual data to be transferred. That is, the information that the application wants to transmit.

CRC Field (16 bits) – The CRC (cyclic redundancy check) field contains a 15 bit CRC sequence, used to check frame integrity, and a 1 bit CRC delimiter. The CRC sequence is the result of multiple XOR operations between a predetermined polynomial and the SOF, arbitration, control and data fields. Receivers also calculate the sequence themselves to check if the frame is error free. As to the CRC delimiter, it is always transmitted with recessive value.

ACK Field (2 bits) – The acknowledgement field is made of an ACK slot and an ACK delimiter. A receiving node signals a successful message reception by means of a dominant bit during the ACK slot. Otherwise, it sends a recessive bit to the bus. During this time, the transmitter listens to the transmission channel. Alike in the CRC field, the ACK delimiter ends the ACK field and is transmitted with recessive value.

End of Frame **Sequence** (7 bits) – Made of seven consecutive recessive bits that end every data frame.

The remaining three frames types are characterized as following:

- Structurally, a remote frame is almost identical to a data frame. The difference between the two is that a remote frame does not transport any information (there is no data field).
- Overload frames are composed of two fields: an overload flag with 6 dominant bits and an overload delimiter with 8 recessive bits. They are transmitted between data and remote frames to extend the intermission field.
- An error frame also consists of two fields: an error flag with 6 bits of the same level (all recessive or all dominant) and an error delimiter with 8 recessive bits. It is triggered every time a node detects an error condition in the network. Once an error frame is issued by a node it eventually becomes visible to all other nodes in the network. Consequently, they abort their current transmission and emit themselves other error frames. This frame type serves the purpose of keeping the network consistent by forcing all nodes to accept or discard a message.

Messages in CAN are coded by the non-return-to-zero binary mechanism. This means that the bus is always in one of two possible states: recessive (when transmitting a ‘1’ bit) or dominant (when transmitting a ‘0’ bit). Given that, nodes are only able to synchronize themselves with others when they see a change in the bus level. If a transmitter sends a continuous sequence of bits of the same value, there is no opportunity for synchronization. If this sequence is long enough, clocks in two different nodes may drift from one another, causing distinct readings of the same bit stream (missing or doubling a bit). To overcome this problem and guarantee synchronization within the network, additional measures are considered in the CAN protocol.

The bit stuffing technique is one of those measures. In data frames and until the end of the CRC sequence, after every five consecutive bits of identical value an additional one, of opposite polarity, is inserted into the bit stream (stuff bit). From the receiver side, the bit following those five bits is interpreted as a stuff bit and consequently discarded. This is a way for the protocol to provide sufficient synchronization points between transmitter and receivers, while adding a small overhead to the network.

In a CAN network there is a 1Mbps data rate limit, introduced into the protocol by design. This bound applies to every frame type and can take a lower value depending on the length of the network (larger networks imply a lower data rate).

2.2 Message prioritization

At the beginning of every frame there is an *identifier* field, which determines its priority on the network. All nodes with messages waiting for transmission start a transmission by sequentially sending the first bits of their frames into the bus. At this point two different scenarios are possible:

1. A node sends a dominant bit and reads a dominant bit from the bus (as it will always do in the absence of errors). In this case it carries on with the transmission and sends the next bit.
2. A node sends a recessive bit and reads a dominant value from the bus. This means that another node is attempting to send a message with a higher priority and consequently it aborts.

By the end of this process, the node sending a frame with the lowest valued identifiers, wins arbitration, as it becomes the only transmitter, and simply carries on with the message diffusion. All other nodes turn into receivers, which are synchronized with the transmitter. Furthermore, messages that have lost the arbitration process are scheduled to be tentatively retransmitted after the full reception of the current message in the bus. Nondestructive arbitration is one of CAN's fundamental characteristics for it allows the protocol to resolve collisions without wasting network bandwidth.

Two frames can have the same identifier if, and only if, one is a data frame and another is a remote frame. In such cases the **RTR** bit, which is part of the arbitration field, will force the nodes sending the remote frame to abort its transmission, since it goes with recessive value in remote frames and with dominant value in data ones. This guarantees that data frames have precedence over remote ones.

2.3 Message validation

Usually, CAN networks operate in environments with a high degree of electromagnetic interference which can cause the corruption of bits in the bit stream, by altering the bus level.

These kinds of errors have a determining effect on message integrity and, thereby, must be controlled. Moreover, faulty components may also undermine network operation by invalidating correct messages or transmitting erroneous ones. To impede such events from causing a malfunction or even disabling the network, a set of message validation mechanisms were introduced in CAN.

Different error detection mechanisms are put in place depending on whether a node is sending or receiving a message. These errors are then categorized according to its origin, enabling a better overall comprehension of network behavior in the presence of faults.

Error detection

When a transmitting node is sending a frame through the bus, it monitors its state after every transmitted bit. Should a transmitting node detect a deviation between the sent and observed value (e.g. sends a recessive bit and reads a dominant one from the bus) it will interpret it as a **bit error**. There are, however, two exceptions to this rule:

1. During the arbitration phase, when nodes compete for bus access. In this case, if a node sends a recessive value and reads a dominant one it aborts transmission.
2. During the ACK slot, where receivers have to tell the transmitter if the message was correctly received (by means of a dominant bit) or not (by a recessive bit).

Additionally, a recessive level during the ACK slot means that no receiving node correctly perceived the message, meaning that an error has occurred. Such cases are classified by transmitters as **ACK errors**.

From the receiver's perspective, there are other conditions that indicate the existence of an error: Certain frame fields respect a pre-determined fixed format and are always transmitted with the same value. As so, a violation of this rule represents an incoherence in the network and is seen by receivers has a **format error**. Furthermore, the CRC sequence allows receivers to check for frame integrity. Should the result of this mechanism deviate from the expected value, the receiving node will comprehend it as a **CRC error**. At last, a node is in the presence of a **stuff error** whenever there is a violation to the bit stuffing rule, that is, it reads from the bit stream more than five consecutive bits of equal value (e.g. 6 dominant bits).

Error signaling

Every time a node detects an erroneous message it has to inform the rest of the network by means of an error frame. In most cases this signalization begins at the bit that immediately follows the detection. However, in the case of CRC errors, it only starts after the ACK delimiter.

Furthermore, error frames were projected to violate bit stuffing rules. Thus, every node listening to an error frame transmitted into the bus will also start the transmission of one of their

own. This is a way of guaranteeing that error frames are eventually perceived by all nodes in the network.

2.4 Fault confinement and node state

The behavior of a CAN node varies according to its internal state which, to some extent, is determined by the actions it has been taking on the network. As such, there are three possible node states:

- **Error Active:** The node takes part in bus communication. When it detects an error it issues an error active flag.
- **Error Passive:** The node takes part in bus communication but, after a transmission, it waits before initiating a new one (thus giving precedence to error active nodes).When the node detects an error issues an error passive flag.
- **Bus-off:** The node is impeded to take part in bus communication.

State assignment is an easy way of limiting the action of a faulty node in the system. Node internal counters keep track of their own number of identified errors. An upper bound is defined (during network configuration) for the maximum number of identified errors before a node categorizes itself as error passive or bus-off. The logic in this mechanism is that if a node is identifying an abnormal number of errors, maybe their source relies in itself and, as so, diminishes its intervention in the network.

No node in the network has global knowledge about the network state because node status is local. As such, every node controls and acts merely upon itself. This is somewhat impeditive in terms of network membership but, on the other hand, helps to keep the protocol simpler.

3. CAN FD Protocol Description and Behavior

For better understanding, CAN FD can be seen as an upgrade to CAN instead of an entirely new protocol, mainly because its architecture and most of its "*modus operandi*" remained untouched or suffered minor changes. Also, CAN FD only defines part of the protocol operation, more specifically what concerns the transmission of actual data (data frames). The rest of its internal processes are supported by the original CAN protocol. As such, the CAN FD protocol, defined in [2], can be explained by addressing the following themes, which are detailed in section 3.1:

- Message transmission and codification;
- Message prioritization;
- Message validation;
- Fault confinement and node state.

Summarily, CAN FD protocol as two main features which differentiate it from CAN. On one hand, it enables the transmission of messages with a payload up to 64 bytes instead of the usual 8 bytes in CAN. On the other, it introduces a secondary data rate at which part of the message can be transmitted up to 15 times CAN upper bound (1 Mbps). These changes have a direct impact in frame transmission times as we show latter in this chapter (section 3.2).

3.1 CAN FD protocol

3.1.1 Message transmission and codification

Unlike CAN, which specified 4 frame types, each with a different purpose, CAN FD only resorts to one specific frame to communicate: data frame. The changes made to the original protocol, which are in the scope of this work, refer specifically to this frame type.

Therefore, message transmission in CAN FD is based only in data frames. This means that the protocol is intended to improve the transmission of user level information in a CAN network, instead of replacing the entire network infrastructure. Nevertheless, there are a lot of structural changes to data frames, which must and will be addressed next.

As in CAN, CAN FD defines two distinct data frame formats.

- CAN FD base format;
- CAN FD extended format.

The two formats are distinguished in the same way as in the CAN protocol. That is, the extended frame format comprehends some additional bits in the starting fields of a frame (arbitration and control fields) in order to enable an augment in the number of active nodes on a CAN FD network.

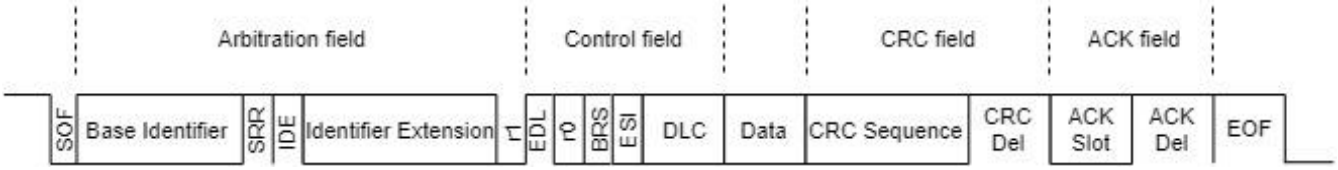

Figure 3.1 – CAN FD Data Frame (Extended Format)

A CAN FD data frame is of longer length and higher complexity. First, it not only has a wider payload, but a larger CRC field as well (to compensate for the frame growth). Second, it incorporates some additional control bits that allow for new functionalities. The field names are the same as in CAN:

Start of Frame (1 bit) – It is the first bit to be transmitted and marks the beginning of a new frame, as in the original CAN.

Arbitration Field (12 bits in base format and 32 in extended format) – It encompasses the Identifier and the **r1** bit (former **RTR** in CAN), which is always transmitted with dominant value since there are no remote frames in CAN FD format. As in CAN, the extended frame format comprises a longer Identifier field (29 bits instead of 11) and two additional bits – The **SRR** (substitute remote request) and the **IDE** (identifier extension) bits – which, respectively, substitute the **r1** bit in the base format and indicate the format in which the frame is being transmitted. Again, the values are dominant for the base format and recessive for the extended one.

Control Field (9 bits in base format and 8 in extended format) – The control field contains part of the changes accomplished by CAN FD. It also comprises the **DLC** (data length code) but, instead of two independent bits, it has five in the base format and four in the extended one. These bits are: The **IDE** (only in the base format, since in extended it passes to the arbitration field); The **EDL** (extended data length) which is always transmitted with recessive value to distinguish from CAN frames; The **r0** that represents a reserved bit for further expansion; The **BRS** bit (bit rate switch) is used to determine if the current bit rate should be switched to a secondary bit rate; The **ESI** bit (error state indicator) that has the mission of informing the receiving node about the transmitter internal state (see 2.2.4 - fault confinement and node state).

Data Field (up to 512 bits) – As in CAN, it is where is located the actual information to be transmitted.

CRC Field (18 or 21 bits) – In CAN FD the CRC field contains a CRC sequence of variable length, which is used to check frame integrity, and a 1 bit CRC delimiter. Just like in the former protocol, the CRC sequence is the result of multiple XOR operations between a predetermined polynomial and the SOF, arbitration, control and data fields. The main difference introduced by CAN FD is that the payload size determines the polynomial to be used in the CRC sequence calculation (bigger payloads means a larger polynomial). Receivers calculate the CRC sequence upon frame reception to check if it is error free. The CRC delimiter is always transmitted with recessive value.

ACK Field (2 bits) – The acknowledgement field is characterized in the same way as in CAN. It is made of an ACK slot and an ACK delimiter. A receiving node signals a successful message reception by means of a dominant bit during the ACK slot. Otherwise, it sends a recessive bit to the bus. During this time, the transmitter listens to the transmission channel. Alike in the CRC field, the ACK delimiter ends the ACK field and is transmitted with recessive value.

End of Frame Sequence (7 bits) – Made of seven consecutive recessive bits that end every CAN and CAN FD data frame.

Data frames in CAN FD experience the same codification than in CAN. That is, every message is coded by the non-return-to-zero mechanism and a stuffing technique is applied to ensure the existence of sufficient points for synchronization, during message transmission. Nevertheless, CAN FD introduces a slight change in the stuffing mechanism. In CAN the bit stuffing technique was applied in every data frame and until the end of the CRC sequence. However, in the new protocol, the bit stuffing method is different within the CRC sequence. Here, the stuff bits are inserted at pre-determined positions, starting with a stuff bit at the beginning of the CRC sequence and subsequent stuff bits after every sequence of four "normal" bits. These fixed stuff bits shall have the opposite polarity of the immediately preceding bit. The number of fixed stuff bits in CAN FD is equal to the maximum number of stuff bits that would result from applying the traditional CAN method to the CRC sequence.

CAN FD nodes make use of a dual data rate during frame transmission. After the BRS bit, and if its value is recessive, all nodes switch to a secondary bit rate which is maintained until the end of the CRC field. At this point, transmitter and receivers return to the primary bit rate to conclude the frame's transmission. These two phases are known as *arbitration-phase* and *data-phase*, respectively. In the *arbitration-phase* CAN limits of 1 Mbps have to be respected. However, within the *data-phase* the data rate can take values up to 15 Mbps.

3.1.2 Message prioritization

CAN FD inherits the deterministic, non-destructive arbitration scheme implemented by the CAN protocol (already described in Chapter 2). Nonetheless, one should keep in mind that a network infrastructure supported by both CAN and CAN FD protocols is not only possible but

necessary (since CAN FD only defines part of the protocol operation). As a result, there can be messages with two distinct formats transiting the network. If CAN and CAN FD frames, with equal identifiers, are transmitted to the network at the same time the CAN frame always takes precedence. This is guaranteed by the EDL bit, which is always sent as recessive, in place of the dominants r0 or r1 bits (depending on the base or extended format) in CAN.

3.1.3 Message validation

CAN FD networks are susceptible to the same events that disrupt the normal functioning of a CAN network, such as electromagnetic interference or erroneous components. Therefore, maintaining the same error detection capabilities was a concern for protocol developers, which decided to keep the original mechanisms while introducing some minor corrections, to adapt them to the structural changes carried out by CAN FD.

Error detection

In CAN, every violation to the stuffing rule caused a stuff error. In CAN FD, however, there are two different types of stuff bits – dynamic stuff bits, which follow the traditional pattern of the stuffing mechanism, and fixed stuff bits, which are part of the CRC field codification. This differentiation resulted in a dual error categorization:

- A **stuff error** is triggered every time a **dynamic stuff bit** infringes the stuff rule (6 bits of the same level are read from the transmission media).
- A **format error** is detected if the value of a **fixed stuff bit** is the same as the immediately preceding bit.

The other adjustment made to the error detection mechanisms is a direct consequence of the dual bit rate introduced by CAN FD. As said before, at the CRC delimiter, every node in a network returns to the lower bit rate, enabling receivers to communicate the ACK bit to transmitters. At this point, it is possible to have synchronization failures between nodes in the system, originated by the high transmission rates evidenced during the frame data-phase. As a result, the ACK bit may arrive to transmitters at different points in time. To overcome this problem, CAN FD transmitters tolerate a double bit during the ACK slot, which means that even if a receiver is delayed or advanced one bit time (the ACK arrives during the ACK delimiter, for example) its acknowledgement will be taken into account.

Error signaling

Since CAN FD does not define error frames, every node must resort to the original CAN protocol specification to signal the detected errors. Normally, this is done by simply applying CAN mechanisms. However, if the node that wishes to signal the error is changing information

at a higher data rate than CAN bounds (during the data-phase) it must return to the low data rate before it can issue an error frame. If the node that detects the error is the transmitter, all other nodes will sample multiple times each of its error frame bits (since their bit time is smaller) and eventually detect an error themselves (normal protocol operation). On the other hand, if the node that identifies the error is a receiver, its error frame bits can get consecutively overwritten by the transmitter until one of them gets sampled and fulfills an error condition on the rest of the nodes in the network.

3.1.4 Fault confinement and node state

CAN FD inherits from the original protocol its fault confinement techniques and node states. There is however, a difference that can positively impact the overall knowledge of network condition. Every data frame carries a flag named Error State Indicator (ESI) inside the control field. This bit is intended to provide knowledge to the rest of the network about the current internal state of the transmitting node (error active or error passive).

3.1.5 ISO CAN FD protocol

Upon standardization, the ISO team decided to introduce some additional safeguards into the protocol, to solve a specific corner case which decreased the overall coverage of the CRC field, thus improving its failure detection capability. This adjustment, which was materialized in ISO 11898-1:2015, made the ISO CAN FD non compatible with the original CAN FD, designed and published by Bosch.

The ISO task force announced a modification in the data frames format. They included an additional field, immediately before the CRC sequence, called stuff bit counter, which holds the total number of dynamic stuff bits inserted into a frame. The receiving node can then compare the total number of read stuff bits with this value and signal an error if any divergence is detected. Any error detect during the reading of the stuff bit counter will be interpreted as a CRC error and signaled accordingly (after the ACK delimiter).

3.2 Frame Duration in the Absence of Errors

To achieve the objective of transmitting data faster, CAN FD as to make use of a dual data rate. CAN FD switches between these data rates at pre-determined bits of a frame. In order to facilitate differentiation between the data rates, two distinct names are assigned:

- *Arbitration-phase* - Uses a “normal” data rate (CAN bounds - 1 Mbps);

- *Data-phase* - Can use a “higher” data rate (CAN FD bounds - 15 Mbps).

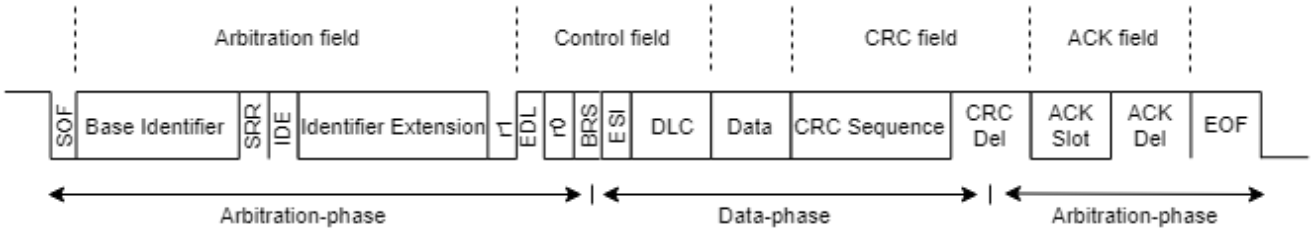

Figure 3.2 – CAN FD Data Frame (Extended Format) with transmission phases

The lower data rate allows a bit to reach the entire network before another one is forwarded, giving all sender nodes the opportunity to listen to others who may also be transmitting. This characteristic is critical during prioritization and validation of messages, and it is used at the beginning and end of the frame. On the contrary, during the data-phase there is only one sender and all other nodes are synchronized with it, making possible the data rate increase to higher values. This new feature, however, makes mandatory the review of how the duration of the frames is calculated, because now different bits inside the same frame have distinct propagation times. Moreover, the frame length, which must also be accounted for, is not the same in CAN and CAN FD.

3.2.1 CAN - Data and remote frames

The main parameters that define the normalized duration of a frame are the frame format specification (base or extended) and the size of the payload field. With exception of the end-of-frame sequence, all the fields in a CAN frame are subject to dynamic bit-stuffing coding. To establish a lower bound (*lb*) for the duration of a data frame, we assume no bits are stuffed in the outgoing stream:

$$t_{data}^{lb} = \left(l_{fix} + l_{dlc} + l_{data} + l_{efs}\right) . \, t_{bit} \qquad (1)$$

where the meaning of the different length parameters in equation (1) is:

- l_{fix} is the length (in bits) of the fixed size fields subject to dynamic bitstuffing. It includes the dominant one-bit start-of-frame (SOF) delimiter, the frame identifier and control bits, as well as the CRC sequence. The exact value depends on the CAN frame format specification (base or extended). However, it does not include l_{dlc}, the 4-bit DLC field;
- l_{data} is the length (in bits) of the payload field. It varies between 0 and 64, in 8 bit increments, being also subject to dynamic bit-stuffing;

- l_{efs} is the length (in bits) of the fixed form sequence, not subject to bit stuffing, that ends every data or remote frame. It includes the CRC delimiter, the 2-bit acknowledgement field and the 7-bit end-of-frame delimiter.

To establish an upper bound (*ub*) for the duration of a data frame, we assume that all the fields subject to dynamic bit-stuffing exhibit a pattern that leads to the maximum insertion of stuffed bits. Therefore:

$$t_{data}^{ub} = \left(l_{fix} + l_{dlc} + l_{data} + \left(1 + \left\lfloor \frac{l_{fix} - l_{stuff} + l_{dlc} + l_{data}}{l_{stuff} - 1} \right\rfloor + l_{efs}\right).t_{bit} \qquad (2)$$

where $\lfloor\ \rfloor$ represents the floor function[1] ; l_{stuff} represents the bit-stuffing width, i.e. the maximum number of consecutive bits of identical value that can be found in the outgoing stream, stuffed bits included. In the worst case, the first (recessive) stuffed bit is inserted in the outgoing stream immediately after the transmission of l_{stuff} initial dominant bits, starting with the SOF delimiter. The minimum and maximum durations of a data frame can be derived by setting l_{data} to zero in equation (1) and l_{data} to the maximum value (64 bits) in equation (2), respectively. To obtain the minimum and maximum durations of a remote frame, l_{data} must be set to zero in both equations.

3.2.2 CAN FD - Data frames

The changes introduced by the CAN FD protocol in the format of data frames, lead to the following updates to equations (1) and (2):

$$t_{FDdata}^{lb} = \left(l_{fix} - 1\right).\ t_{bit} + \left(1 + l_{dlc} + l_{data} + l_{fdcrc}\right).\ \frac{t_{bit}}{\sigma} + l_{efs}.\ t_{bit} \qquad (3)$$

$$t_{FDdata}^{ub} = \left(l_{fdix} + \left\lfloor \frac{(l_{fdix} - 1) - l_{stuff}}{l_{stuff} - 1} \right\rfloor\right).t_{bit} + \left(1 + l_{dlc} + l_{data} + l_{fdcrc} + \left\lfloor \frac{(1 + l_{dlc} + l_{data})}{l_{stuff} - 1} \right\rfloor\right).\frac{t_{bit}}{\sigma} + l_{efs}.t_{bit} \qquad (4)$$

where $\sigma = \frac{baud\_high}{baud\_normal}$, is defined as the ratio between the higher and the normal data rates. The meaning of the new length parameters is as follows:

[1] The floor function $\lfloor x \rfloor$ is defined as the greatest integer not greater than x.

- l_{fdix} is the length (in bits) of the fixed size fields subject to dynamic bit-stuffing transmitted at the lower data rate, which in CAN FD excludes the 4-bit DLC field and the full CRC sequence;
- l_{fdcrc} is the length (in bits) of the CRC sequence, which varies according to the size of the payload: 17 bit for payloads up to 16 bytes, 21 bit for payloads longer than 16 bytes. This parameter also includes the statically inserted fixed stuffed bits (FSB) and the stuff counter in the ISO version.

The mandatory values taken by a relevant set of control bits, such as SRR, IDE, r1, and EDL, in the CAN FD frame format (Figure 3.2), are not considered in the definition of equation (4), which thus slightly overestimates the maximum number of bits that may be dynamically stuffed in the outgoing stream. Table 3.1 summarizes the fundamental parameters required for the assessment of data frames durations and their periods of inaccessibility.

3.2.3 CAN and CAN FD evaluation

To determine the frames transmission time, it was taken into account a maximum and minimum size data, remote, error and overload frames, which represent the best and worst case scenarios. Then, it was adopted a network transmission rate of 1 Mbps for CAN frames and of 1 Mbps in arbitration-phase and 8 Mbps in data-phase, for CAN FD frames. The bit time was measured as the time a node takes to send a bit into the bus. Knowing that the transmission time is inversely proportional to the data rate, the following values were determined: t_{bit} = 1 micro-second in CAN and CAN FD arbitration-phase and t_{bit} = 0,125 micro-seconds in CAN FD data-phase. The results are summarized in Table 3.2. Figure 3.3 shows the transmission time of a data frame in all three versions of the protocol.

Parameters	Frame field length (bit)							
CAN Parameters	l_{bid}	l_{eid}	l_{ctl}	l_{crc}	l_{fix}	l_{data}		
CAN Base format (2.0A)	11	-	4	15	30	≤ 64		
CAN Extended format (2.0B)	11	18	6	25	50	≤ 64		
CAN FD Parameters	l_{bid}	l_{eid}	l_{ctl}	l_{fdix}	l_{data}	l_{crc}	l_{fsb}	l_{fdcrc}
CAN FD Base format	11	-	7	18	≤ 128	17	5	22
	11	-	7	18	≥ 128	21	6	27
CAN FD Extended format	11	18	8	37	≤ 128	17	5	22
	11	18	8	37	≥ 128	21	6	27
ISO CAN FD (both formats)	=	=	=	=	=	+5	=	=

l_{bid} and l_{eid} are the base and extended identifier field lengths, respectively;

l_{ctl} – number of control bits, including the 1-bit SOF but excluding the 4-bit DLC field;

l_{crc} – CRC field length;

l_{fsb} – number of static stuffed bits, in CAN FD.

Table 3.1 – CAN, CAN FD and ISO CAN FD frame field length

Data Rate: 1 Mbps / 8 Mbps						
Protocol	Frame	Symbol	Duration base (µs)		Duration extended (µs)	
			best	worst	best	worst
ISO CAN FD	Data frame	$t_{ISOFDdata}$	31.0	112.9	50.0	136.9
CAN FD	Data frame	t_{FDdata}	30.4	112.3	49.9	136.3
CAN	Data frame	t_{data}	44.0	132.0	64.0	157.0
CAN	Remote frame	t_{rdata}	44.0	52.0	64.0	77.0
CAN	Error frame	t_{error}	14.0	20.0	14.0	20.0
CAN	Overload frame	t_{oload}	14.0	20.0	14.0	20.0

Table 3.2 – CAN, CAN FD and ISO CAN FD frame duration

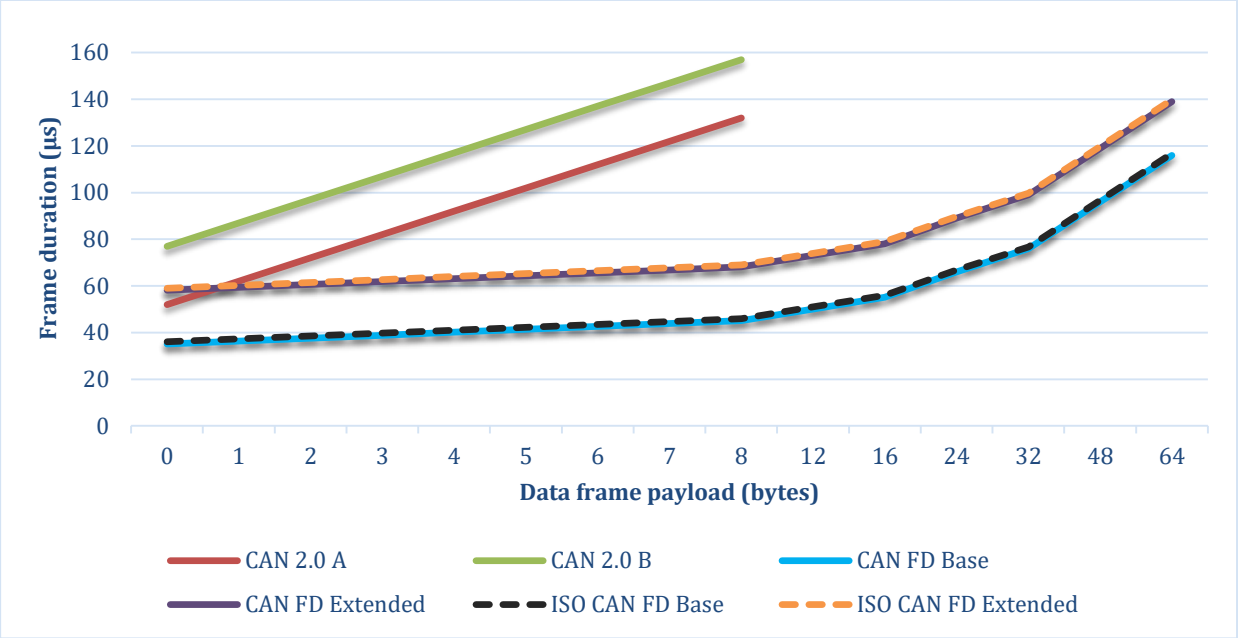

Figure 3.3 – CAN, CAN FD and ISO CAN FD frame duration

Figure 3.3 demonstrates that for the same data frame format, the transmission time is always smaller in CAN FD, even in a zero payload frame. When the payload is 8 bytes long (maximum size in CAN) the transmission time practically reduces to half in CAN FD. The improvement is still existent when comparing a CAN - 8 byte payload - data frame with a CAN FD - 64 byte payload - data frame (several times the size of a complete CAN data frame). Both statements are also true for ISO CAN FD, even taking into account its minor increase in terms of frame size (5 more bits to be transferred during data-phase). These results have a direct impact on the inaccessibility scenarios derived in the next chapter.

4. CAN FD Inaccessibility Characteristics

Real-time networks must guarantee response within specified deadlines. For this purpose, controlling time becomes crucial. On the other hand, data throughput is increasingly a determining factor in current computational systems. However, in networks of this nature, the search for efficiency must not bypass properties like timeliness, reliability or availability.

In a real environment, and in addition to network delay, there are other types of events, like interferences and failed components, which may prevent normal communication during a certain time period generating what is commonly known as partitions and possibly leading to inconsistent states of distributed nodes. To this time period, in which the network is unable to operate, we call inaccessibility period. These intervals must also be accounted for and added to normal transmission delay when considering worst case transmission scenarios in the presence of errors.

In this chapter we aim to provide a comparative analysis of CAN and CAN FD, by carefully analyzing the protocols in terms of: transmission time in the presence of errors, error signaling and error handling. To accomplish these objectives the transmission time of a maximum and minimum size data frame is used, as derived in Chapter 3. Then, based on our understanding of the protocol behavior upon the occurrence of errors, we present, in section 4.1, mathematical equations to determine the time required for the protocol to return to normal operation (this time period represents an inaccessibility period). For a matter of completeness, both releases of the CAN FD protocol (non-ISO and ISO versions) are addressed in this chapter. Furthermore, section 4.2 provides a comparison regarding the same inaccessibility aspects of the original CAN protocol [4] and in section 4.3 we present our conclusions on the subject in question.

4.1 Inaccessibility Scenarios

CAN and CAN FD fault models obeys to a set of rules, in terms of error categorization, that are determined in both protocols specifications, to appropriately manage error detection and signaling mechanisms. In turn, these rules are established based on: the current role of the node which identified the error; the bit at which it occurred. Thus, the most logical approach to follow is to define each error type as a premise, and from there reproduce, observe and measure the events that succeed until the protocol returns to a normal operation state.

The spectrum of the evaluation is derived from the error cases referred to in the protocol specification. The best and worst case scenarios will be referred to as bc and wc superscripts, respectively.

4.1.1 Bit Errors

Corruption of a single bit within a bit-stream can be detected early on by the transmitter error detection mechanism. This technique requires bus verification upon every successful bit transmission. If the monitored bit value differs from the one sent, it will be considered an error. There are, however, two exceptions to this rule:

1. The transmitter sends a recessive bit and samples a dominant one inside the arbitration field.
2. The transmitter sends a recessive bit and samples a dominant one during the acknowledgment slot (ACK).

In case a bit error is detected the node signals it by starting the transmission of an error frame at the next bit slot. Should this error take place in CAN FD data-phase, the node has to return to the arbitration-phase data rate before emitting the error indicator.

The best case scenario for this network inaccessibility period is given by the sum of the earliest possible detection - at the first bit of a data frame, with the minimum time elapsed when signaling the error and with the fixed Interframe Spacing time (IFS). The corresponding formulas for CAN and CAN FD are given by (5) and (6).

$$t_{ina\leftarrow berr}^{bc} = t_{bit} + t_{error}^{bc} + t_{IFS} \tag{5}$$

$$t_{FDina\leftarrow berr}^{bc} = t_{bit} + t_{error}^{bc} + t_{IFS} \tag{6}$$

Unlike the preceding, the worst case scenario comprises a late detection allied with the utmost error signaling time. This late detection can be no longer than *end-of-frame* delimiter in a maximum size data frame.

$$t_{ina\leftarrow berr}^{wc} = t_{data}^{wc} + t_{error}^{wc} + t_{IFS} \tag{7}$$

$$t_{FDina\leftarrow berr}^{wc} = t_{data}^{wc} + t_{error}^{wc} + t_{IFS} \tag{8}$$

The inaccessibility formulas for this error type are consistent with those in CAN. Even so, the corresponding times will eventually be different since t_{data} values are not the same in CAN, CAN FD and ISO CAN FD, as shown in Table 3.1.

4.1.2 Bit Stuffing Errors

In CAN FD, as well as in CAN, additional bits are incorporated into data frames (until the end of the CRC sequence) by the bit stuffing method, to provide enough synchronization points between nodes. After transmission, these dynamic stuff bits are discarded by receivers in order to

correctly reconstruct the original message. During this process these bits are also analyzed according to the bit stuffing rules - no more than 5 bits of identical polarity can be consecutively transmitted - to detect deviations from the expected behavior. Under normal operation, the deleted bit should be of opposite polarity of the preceding ones. If not, the node will interpret it as an error and start the transmission of an error frame.

The first position at which there can be a dynamic stuff bit (the sixth bit of a frame) represents the best error detection case for this error type. Thus, the respective inaccessibility formula goes as following, where l_{stuff} represents the bit stuffing width (5 bits).

$$t_{ina\leftarrow stuff}^{bc} = \left(l_{stuff} + 1\right).t_{bit} + t_{error}^{bc} + t_{IFS} \quad (9)$$

$$t_{FDina\leftarrow stuff}^{bc} = \left(l_{stuff} + 1\right).t_{bit} + t_{error}^{bc} + t_{IFS} \quad (10)$$

The worst case scenario, however, is calculated differently for CAN and CAN FD. CAN bit stuffing mechanism only ends at the CRC delimiter, while in the new protocol the stuff bits inside the CRC field are inserted at fixed positions and thus, do not follow the same pattern and the same error categorization scheme. This way, the traditional technique of stuff bits insertion ends at the last bit of the data field. As such, the point at which the error is detected, in the worst case scenario, is given by the following expression with t_{CRC} being the CRC field duration and t_{EFS} the duration of the fields that follow it, until the end of the frame.

$$t_{ina\leftarrow stuff}^{wc} = t_{data}^{wc} - t_{EFS} + t_{error}^{wc} + t_{IFS} \quad (11)$$

$$t_{FDina\leftarrow stuff}^{wc} = t_{data}^{wc} - t_{CRC} - t_{EFS} + t_{error}^{wc} + t_{IFS} \quad (12)$$

4.1.3 CRC Errors

In the CAN protocol, the CRC field is intended to check frame correctness and is made of a 15 bit CRC sequence plus a 1 bit CRC delimiter. While this is sufficient to guarantee a reasonable coverage in CAN 2.0 version A, the same does not apply to version B (its extension), due to the inclusion of more bits into data and remote frames. In order to correct this flaw, CAN FD increased the size of the CRC field sufficiently enough to restore its original error detection capability. The CRC mechanism and the changes described above can be found with more detail in the protocol specification [2] and in [6].

Furthermore, in ISO CAN FD, additional safeguards were included to enforce frame's integrity. As so, the new data frame CRC field is made of a stuff count, a 17 or 21 bit CRC sequence - according to the frame's length, a 1 bit CRC delimiter and Fixed Stuff Bits. The stuff

count enables a more efficient control by exposing masked stuff bits (imperceptible stuff bits due to an inversion in one of the bits of the originating sequence) and false positives. An error detected in the stuff count, as well as in the CRC sequence itself, is always signaled as a CRC error.

Should a CRC error be detected, it will only be signaled after the ACK Delimiter. Thus, the worst and best case scenarios only differ in error signaling time and size of the frame, and are the same for both protocols. The changes introduced by the CAN FD protocol are only visible at the level of inaccessibility times.

$$t_{ina\leftarrow CRC} = t_{data} - t_{EOF} + t_{error} + t_{IFS} \quad (13)$$

$$t_{FDina\leftarrow CRC} = t_{data} - t_{EOF} + t_{error} + t_{IFS} \quad (14)$$

4.1.4 Form Errors

CAN FD data frame fields must follow a predefined format. In turn, some of these fields have reserved bits that are always transmitted with the same value to the bus. They are:

- the CRC delimiter;
- the Acknowledgement delimiter;
- the End of Frame field (sequence of seven consecutive recessive bits).

The protocol defines that each one of the previous mentioned bits must be transmitted with a recessive level and not be changed by any other node in the network. Therefore, any violation to this rule will result in a form error.

Additionally, and despite fixed stuff bits lay within the CRC field, the CAN FD specification [2] defines that an error in their polarity must also be interpreted and signaled as a form error. Thus, the best case scenario is obtained by considering the presence of an error in the first Fixed Stuff Bit (at the beginning of the CRC field) of a minimum size data frame.

$$t^{bc}_{ina\leftarrow form} = t^{bc}_{data} - (t_{EFS} - t_{bit}) + t^{bc}_{error} + t_{IFS} \quad (15)$$

$$t^{bc}_{FDina\leftarrow form} = t^{bc}_{data} - (t_{CRC} + t_{EFS} - t_{bit}) + t^{bc}_{error} + t_{IFS} \quad (16)$$

In opposition, an error can occur upon the reception of the last but one bit of the End of Frame field in a maximum size data frame (the detection of a dominant bit during the last bit of End of Frame is not considered an error by receivers). Therefore, the inaccessibility equation goes as following:

$$t_{ina\leftarrow form}^{wc} = t_{data}^{wc} - t_{bit} + t_{error}^{wc} + t_{IFS} \quad (17)$$

$$t_{FDina\leftarrow form}^{wc} = t_{data}^{wc} - t_{bit} + t_{error}^{wc} + t_{IFS} \quad (18)$$

There are notorious differences between CAN and CAN FD best case inaccessibility formulas, for data frames, manifested by an earlier detection in the second one, due to the interpretation of a Fixed Stuff Bit inversion as a form error. Form errors in the remaining frame types (remote, overload and error frames) are not addressed since they are not applicable to CAN FD.

4.1.5 Acknowledge Errors

The CAN FD acknowledgment mechanism is the same as in CAN - after a successful message reception, a CAN FD node signals it to the sender by means of a dominant bit during the AKC bit. In turn, at this point, the sender expects the bus to be in a dominant level and, if not, starts the transmission of an error frame at the next bit slot. Since dominant bits overwrite recessive ones, the sender only detects an error if all receivers agree on the message incorrectness.

The corresponding network inaccessibility equations, for the best and worst case scenarios, only differ in terms of frame's length and error signaling time, because ACK errors always take place in the same bit.

$$t_{ina\leftarrow ack} = t_{data} - (t_{EFS} - 2.t_{bit}) + t_{error} \quad (19)$$

$$t_{FDina\leftarrow ack} = t_{data} - (t_{EFS} - 2.t_{bit}) + t_{error} \quad (20)$$

It is possible that the sender detects a falsified dominant bit during the ACK slot due to interferences on the bus. In such a scheme, the transmitter has no way of detecting that the message was, in fact, correctly received by the other nodes. Message integrity is guaranteed by receivers, who will signal the error that led to the transmission of a recessive bit during the ACK slot in the first place (as a CRC error).

4.1.6 Consecutive and Successive Errors

Until this point we have been considering the presence of errors during the transmission of a frame in an isolated manner. However, this assumption is not enough to represent a real environment, where faulty nodes, erroneous transmission mediums or electromagnetic interferences cause errors to often come in bursts, possibly affecting not one, but several consecutive frame transmissions.

Multiple Consecutive Errors:

In order to correctly address this issue we begin by assuming a premise **P**, which states that: *during a known bounded time interval **T**, there cannot be more than **n** transmissions affected by errors.* Additionally, we also assume that during **T,** the network does not suffer from more than one fault (the error burst has the same origin). This fault model implies that the network always recovers to a normal operation state after the transmission of no more than **n** erroneous frames, providing an upper bound for the correspondent inaccessibility time.

CAN FD, as well as CAN, signals errors by starting the transmission of an error frame. This frame includes an error flag, consisting of 6 dominant bits, which forces all other nodes to transmit their own error frame, as it violates the stuffing rule. When a recessive level is detected, it means that all nodes have finished the transmission of the error flag and that the 8 bit error delimiter is now beginning. One should keep in mind that during this process the network is not safe from additional errors being detected, which leads us to the scenarios considered below.

In a situation where a network malfunction causes an error burst that affects **consecutive** frames, the best case inaccessibility time is derived by considering the occurrence of such errors during the first bit of each frame.

$$t^{bc}_{ina\leftarrow cerr} = 2.t_{bit} + t^{bc}_{error} + t_{IFS} \quad (21)$$

$$t^{bc}_{FDina\leftarrow cerr} = 2.t_{bit} + t^{bc}_{error} + t_{IFS} \quad (22)$$

Since the first bit of a CAN FD frame is placed inside the arbitration-phase, it has the same data rate bounds than a CAN frame. This means that not only the inaccessibility equations (21) and (22) are the same, but also the actual inaccessibility time derived for both protocols. In the worst case scenario, the **P** assumption serves as a limit for the maximum number of consecutive erroneous frames transmitted in the network. As such, the longest inaccessibility time has to consider the detection of an error at the last bit of a maximum size data frame and **n-1** consecutive error frames sent afterwards.

$$t^{wc}_{ina\leftarrow cerr} = t^{wc}_{data} + n.t^{wc}_{error} + t_{IFS} \quad (23)$$

$$t^{wc}_{FDina\leftarrow cerr} = t^{wc}_{data} + (n-1).t^{wc}_{error} + t_{IFS} \quad (24)$$

Multiple Successive Errors:

Let's now consider a scenario where a faulty component only affects data frames. Such behavior can be observed in a node that exhibits a failure pattern only while transmitting. In such a scheme, the error signaling would always be successful, since we are considering a single fault model. Thus, the corresponding best and worst inaccessibility duration is given by the following expression:

$$t_{ina\leftarrow serr}^{wc} = n.(t_{data}^{wc} + t_{error}^{wc} + t_{IFS}) \quad (25)$$

$$t_{FDina\leftarrow serr}^{wc} = n.(t_{data}^{wc} + t_{error}^{wc} + t_{IFS}) \quad (26)$$

Practical Approach:

The error confinement mechanisms, in both CAN and CAN FD protocol prevent a faulty node from having a permanent and continuous effect on the network. These safeguards guarantee that if a node exhibits a deviant behavior, and this behavior exceeds a pre-defined bound (in terms of the number of errors it has detected), its action on the network is repressed. This is achieved by means of a dual counter present inside every node that holds the number of produced and identified errors. These counters sum up, whenever an error is accounted for, at a higher rate than they decrease, when a correct transmission terminates, to keep track of the errors distribution in time. The error counters and their defined threshold determine the current node state: error active; error passive; bus-off. The description of these states can be found in chapter 2 and 3.

The practical effect of this mechanism is that **P** can be seen as more than a theoretical premise for our fault model, as it happens to have a direct application in CAN and CAN FD by means of the error counters threshold. In practice, this means the **n** variable can be replaced by an expression that encloses the error-active counter threshold and the real counter growth rate, to calculate the actual inaccessibility time of a real implementation. To demonstrate this revelation, we can represent equations (25) and (26) in an alternative way (27), where ⌈ ⌉ represents the ceiling[2] function, err_{a_thd} symbolizes the error active count threshold and Δ_{rterr} the rate at which the counter increases for each identified error.

$$t_{FDina\leftarrow serr}^{wc} = \left\lceil \frac{err_{a\_thd}}{\Delta_{rterr}} \right\rceil . (t_{data}^{wc} + t_{error}^{wc} + t_{IFS}) \quad (27)$$

The original paper on CAN inaccessibility characteristics [4] encompasses several scenarios concerning overload errors. The previous study is still valid in CAN FD, simply because the new

[2] The ceiling function $\lceil x \rceil$ is defined as the smallest integer not smaller than x.

protocol only defines data frames and the over mentioned analyzes is limited to the intermission period between them. This means that the changes introduced by CAN FD have no implication on the inaccessibility expressions or in the inaccessibility times derived for the overload error cases. This way, its inclusion in the present thesis becomes redundant and will not be made.

4.2 Evaluation

For demonstration of our study and further comparison with CAN protocol, we determined the worst case inaccessibility times of a CAN FD network in the presence of the several error types addressed in this chapter. Let us assume a network configuration with maximum data rate bounds of 1 Mbps for CAN and CAN FD arbitration-phase and 8 Mbps for the CAN FD data-phase (ISO and non-ISO versions), since current transceivers do not support higher throughputs. Note that the inaccessibility equations for each one of the error types make use of the normalized frame transmission times calculated in section 3.2.

Error scenarios	Normalized worst case inaccessibility duration					
	T_{ina} (μs)		T_{fdina} (μs)		$T_{ISOfdina}$ (μs)	
	CAN Base (2.0A)	CAN Extended (2.0B)	CAN FD Base	CAN FD Extended	ISO CAN FD Base	ISO CAN FD Extended
Bit errors	155,0	180,0	135,3	159,3	135,9	159,9
Bit-stuffing errors	145,0	170,0	121,9	145,9	121,9	145,9
CRC errors	148,0	173,0	128,3	152,3	128,9	152,9
ACK errors	147,0	172,0	127,3	151,3	127,9	151,9
Form errors	154,0	179,0	134,3	158,3	134,9	158,9

Table 4.1 – CAN, CAN FD and ISO CAN FD inaccessibility duration

The results of the experiment are summarized in Table 4.1, where we can observe the actual times of inaccessibility for each version of the protocol. Figures 4.1 and 4.2 show the same results in a graphical perspective.

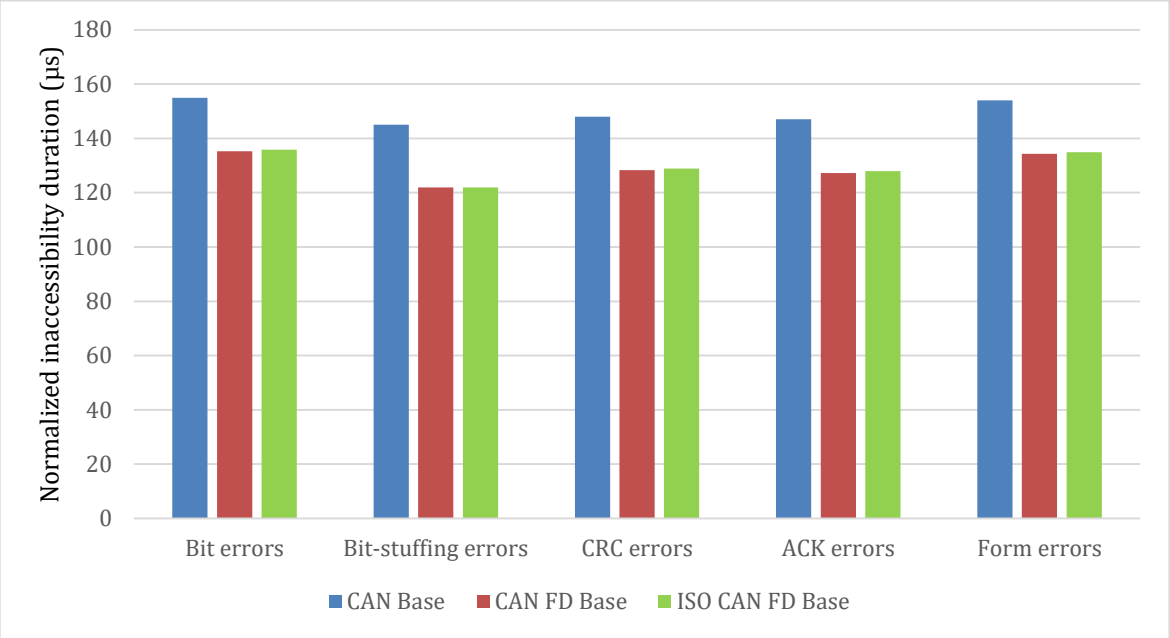

Figure 4.1 – CAN, CAN FD and ISO CAN FD base format inaccessibility duration

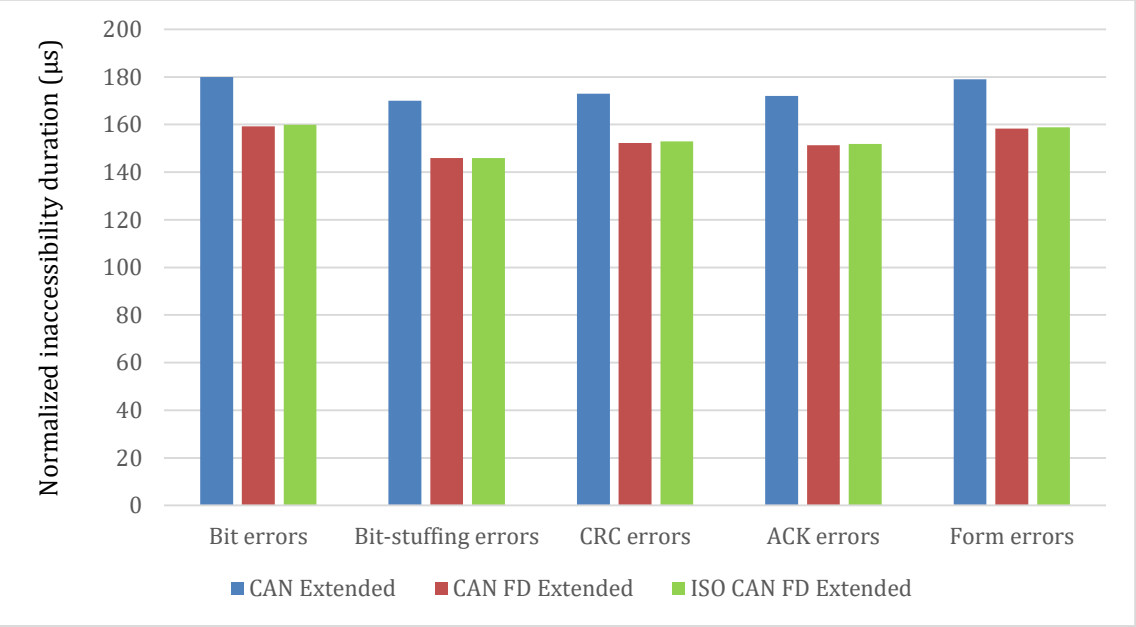

Figure 4.2 – CAN, CAN FD and ISO CAN FD extended format inaccessibility duration

Analyzing results

The time taken by the protocol to recover from partitions (since the beginning of the erroneous frame transmission until the end of the last error frame), in the worst case scenarios,

suffers a decrease in the various error types relative to CAN. This is mostly due to the CAN FD secondary bit rate that speeds up frames transmission. The results are surprising if we take into account that the worst case scenarios in CAN FD comprise a 64 bytes long data field, instead of the usual 8 bytes in CAN. The Bit-Stuffing scenario, where the inaccessibility time suffers the biggest improvement, takes advantage of the new CAN FD frame format - the traditional bit stuffing mechanism ends at the **last bit of the data field**, instead of at the **last bit of the CRC field**, meaning that, in the worst case scenario, this type of error will, with certainty, take place earlier in the frame.

We can then conclude that when studying worst case scenarios, in a network with this configuration (8 Mbps), the data rate increase does overcompensate for the CAN FD data frame's growth, leading to smaller inaccessibility times than in CAN. Furthermore, the tendency is to increase this gap as data rates become higher, since the CAN FD protocol allows for a data rate up to 15 Mbps.

Data-rate configuration plays a leading role when it comes to network inaccessibility duration. Bigger frames do not necessarily mean larger inaccessibility times. In this case, the data-phase transfer rate is inversely proportional to the inaccessibility duration of the worst case scenarios in a CAN FD network. As new transceivers with higher transfer capabilities emerge, we are expected to see the times of inaccessibility diminish.

4.3 Conclusions

CAN FD brought remarkable improvements to the original network protocol in terms of throughput, by augmenting the transmission data rate and payload size of data frames. Section 3.2 demonstrates that a CAN FD frame takes roughly half the time to be transmitted than a CAN frame with identical payload. Additionally, messages up to 64 payload bytes can be transmitted at once with no additional delay, in place of the usual 8 bytes in CAN. From an efficiency point of view, CAN FD meets the intended objectives, as it bridges the existent gap between market needs and the protocol characteristics, hypothetically increasing its range of uses and applications. However, real-time networks are all about timeliness and the occurrence of errors during normal operation can impact the network timing properties. From this perspective, it is of crucial importance to verify if the former error detection, signalization and confinement mechanisms are as effective in the new protocol as they were in CAN.

In section 3.3 we show that not all the error categories needed to have their inaccessibility equations reformulated due to the structural changes carried out by CAN FD: The bit-stuffing error, in a worst case approach, now occurs at an earlier moment in the frame; similarly, the form error best case inaccessibility formula now considers the occurrence of an error at the first fixed stuff bit of a frame (first bit of the CRC field) instead of at the CRC delimiter. Nevertheless, the dual data rate introduced by CAN FD, allied with the frame length increase, determines that equal

equations can produce distinct values for the two protocols, because the actual variables may hold distinct values. Our evaluation, described in section 4.2, demonstrates that this divergence translates into an overall improvement of the inaccessibility times in CAN FD. This means that the new protocol, upon the occurrence of errors, returns faster to normal operation than CAN, even though we are considering an 8 times larger payload. Once more, the higher data rate plays a determining role in this achievement.

ISO CAN FD distinguishes itself from the non-ISO version for having an additional set of bits placed at the beginning of the CRC field and intended to increase the coverage of the protocol error detection mechanisms. All other aspects concerning frame structure and message transmission remain unchanged. This new safeguard means an increase of five bits in data frames length and the correspondent value in the transmission time (which is dependent on the data rate). Consequently, the scenarios that consider the occurrence of errors at the end, or after, the CRC field, will also experience a slight augment in their inaccessibility duration relative to the non-ISO version. Still, their transmission times, both in the presence and in absence of errors, are so similar in all considered cases that we can conclude that the changes introduced by ISO CAN FD are almost irrelevant to this particular study.

In a real-time network predictability is the key. Thus, only considering worst case transmission delays in the absence of errors might not be enough to guarantee the protocol correct behavior and can represent a serious hazard in time-critical applications. In order to ensure reliable communication in a CAN FD network, the inaccessibility formulas presented in this paper as well as the ones in [4], should be derived and added to those times. This study is relevant as it provides grounds for a more informed and secured approach to current and upcoming CAN FD implementations, in order to develop more robust and reliable applications.

During the completion of this work we came across a number of questions regarding the changes made to the original CAN protocol. On one hand, the error categorization of the fixed stuff bits did not seem entirely consistent. On the other, the benefits of some of the new features appeared not to fully compensate the complexity increase they carried. When we took a closer look at these changes, we noticed that not all of them were sufficiently substantiated. To address this concerns we started by decomposing the problem into different modification layers to, from there, critically analyze the relevance of the issues that lead to them, as well as the way they were carried out. The result of this study is presented in chapter 5.

5. CAN FD Development Analysis

In this chapter we revisit the ISO and non-ISO versions of CAN FD with the goal of extensively analyzing, one by one, each of the several modifications made in relation to CAN. This evaluation is intended to uncover not only the structural and functional impact they have on the protocol, but mostly their reasonability.

In section 5.1 we address the modifications made to CAN upon the development of the non-ISO CAN FD version. We start by analyzing the newly bits introduced in the control field. Then, we explain the need for the inclusion of a dual CRC polynomial in CAN FD and assess its impact. Next, in section 5.1.3, the concept of dynamic stuff bits (a creation of the CAN FD development team) is introduced, along with some observations regarding its influence in the protocol operation. Section 5.1.4 reveals an incoherence in the CAN FD protocol specification, namely in the categorization of the errors affecting stuff bits. Finally, section 5.1.5 shows a specific fault time that reduces the coverage of the CAN error detection mechanisms and demonstrates how it is fixed in CAN FD.

The second part of this chapter focuses on the standardization of CAN FD. This process resulted in the inclusion of an additional field in the frame, which yields the total number of inserted stuff bits. In section 5.2 we will see why the ISO team found important to include this piece of information in the frame. We start with a brief explanation of the motivation behind this decision, which rests on the discovery of a specific fault type that, in turn, depends on the validity of two premises. As such, our first goal is to legitimize these premises (section 5.2.1). Next, in section 5.2.2 we focus our attention on the fault type itself, which can manifest in two possible ways. Here, we identify an incorrect analysis made by the ISO task force, comprising one of these two possible manifestations. Section 5.2.3 describes the adopted solution to this problem that, as said before, involves adding an additional field in the frame. In section 5.2.4 we provide an overview on this subject and discuss the decisions taken during this development process. We close section 5.2 by disclosing a new fault type, which affects all versions of the protocol (CAN and CAN FD alike) and remained concealed until now.

5.1 From CAN to non-ISO CAN FD (Bosch)

There are two major differences from classical CAN to the non-ISO version of the CAN FD protocol: Transmission rate and payload size. However, the changes don’t exhaust here. First, some of the control bits serve now different purposes to meet the needs of CAN FD. Second, the CRC calculation is now done based on one of two available polynomials and both of them include stuff bits. Third, the concepts of dynamic and fixed stuffed bits were introduced in data frames and encompass different usages.

5.1.1 Control Bits

Some CAN control bits have been adjusted in CAN FD, in order to guarantee an appropriate control over all data frames. The r1 and BRS bits ensure that receivers detect the version of the protocol in use and are informed about the need to switch to a secondary data rate ahead in the frame. The presence of these bits derives from the structural modifications carried out by the new protocol. Conversely, the ESI bit introduces a new feature to the protocol itself. It is an indicator that explicitly informs all other nodes in the network about the transmitting node state - a dominant value indicates an active error state, and a recessive value a passive error state. This near non-intrusive observability enables the early detection of faulty nodes in the network, a feature that might be exploited by higher level services addressing issues like [7] [8]:

- Membership;
- Transmission delays and response times;
- Fault tolerant communication.

This information is not especially useful at the data-link level, because a node does not take any immediate action depending on the knowledge of the transmitting node state. On the contrary, message processing is carried on no matter the value of the ESI bit. This information is sent to the layers above, so that it is used by protocols operating at a higher abstraction, dealing with issues such as the ones enumerated above.

5.1.2 Dual CRC polynomial

CAN FD aims at guaranteeing, at least, the same error detection capabilities exhibited by the original CAN protocol. The CRC sequence, present in both versions of the protocol, allows a node to check the integrity of a received message with a certain level of confidence. This effectiveness is given by the Hamming Distance[3] (HD) of the CRC generator polynomial. The CAN CRC 15-bit polynomial features a Hamming Distance of 6. However, the increase in frame size of the CAN FD protocol requires the use of higher order polynomials to maintain the same error detection capability.

In the CAN FD protocol, a maximum size data frame needs a 21-bit polynomial to assure a HD of 6. This means that the polynomial is large enough to maintain the desired error detection coverage for all frames. Nevertheless, the protocol development team decided to create two different size polynomials to be used in distinct situations. The CRC sequence of a CAN FD frame with a data field up to 16 bytes is calculated based on a 17-bit polynomial. In opposition, every frame with a data field bigger than 16 bytes uses the 21-bit polynomial. The obvious benefit of this mechanism is that, in some transmissions, the CRC sequence will be 5 bits shorter (4 CRC

[3] The Hamming distance between two strings of equal length is defined as the number of bits we must change to turn one into the other. In practice, a Hamming distance of 6 means that the CRC always detects up to five randomly distributed bit failures.

bits and 1 stuff bit). On the other hand, now every node needs to calculate three CRC sequences concurrently, with the 15-bit (for CAN frames), 17-bit and 21-bit polynomials [2].

The direct consequence of this measure is that the three polynomials have to be kept in memory for current use by every node in the network (one for frames in the CAN format and two for CAN FD frames), which implies additional memory allocation for this specific task. Additionally, there are extra computation costs that should also be considered. However, we could not find evidence that the 5 bit decrease in a fraction of the transmitted frames compensates for the augment in complexity, memory allocation and executed tasks.

5.1.3 Dynamic and Fixed Stuff Bits

Every data and remote frame in the CAN protocol follows a codification pattern known as bit stuffing mechanism, from the start of frame bit to the end of the CRC field, to ensure that there are enough synchronization points during a frame's transmission. This was performed in a uniform way in CAN, but the same does not apply to CAN FD.

In the CAN FD protocol, the rule for inserting stuff bits varies along the frame. From the start of frame bit to the end of the data field it is used the traditional method of bit stuffing, which originates the now called dynamic stuff bits (DSB). However, from the first bit of the CRC field until the CRC delimiter, stuff bits are inserted at fixed positions, with the opposite polarity of the immediately preceding bit. This means that there will be a stuff bit after every sequence of five bits, even if they are not of the same level. Following a methodical approach, we can discuss its negative and positive aspects:

Upsides – With the introduction of fixed stuff bits (FSB), CAN FD guarantees that receivers correctly withdraw every stuff bit within the CRC field of a frame. FSB also serve as an improved frame format checking mechanism as they are fixed and is required for them to have the opposite polarity of the immediate preceding bit.

Downsides – The worst case stuff bit insertion is considered by default, inside the CRC field. This implies that a frame can have up to 6 stuff bits that otherwise could not have been introduced. Considering that the CRC field is placed within the frame's data-phase, the data rate can vary from low values to 15 Mbps, which translates into a variable bit time[4]. If we take a pessimist approach and consider an equal data rate of 1 Mbps for the arbitration and data phases, 5 or 6 microseconds, depending on the CRC polynomial that is used (the longer polynomial needs one more fixed stuff bit), will be unnecessarily added to the frame transmission time, in CAN FD, due to the introduction of the fixed stuff bits. At higher data rates, this overhead tends to diminish. Nevertheless, in relative terms, these 5 or 6 additional bits can represent almost 10% of the overall bits in frame, since it can range from 64 (minimum) to 734 (maximum) bits.

[4] Bit time = 1 / data rate. At 1 Mbps the bit time is 1 micro second.

There is one reason that strongly motivates the creation of two distinct stuff bit codification mechanisms. This issue will be properly addressed when another change introduced by CAN FD is reviewed: the inclusion of DSB into the CRC calculation (section 5.1.5), where we will show that the traditional method of bit stuffing could not have been applied to the CRC field.

5.1.4 Fixed Stuff Bits error categorization

Dynamic stuff bits correspond to the traditional CAN stuff bits and, as such, exhibit the same error identification, categorization and signalization mechanisms. However, fixed stuff bits behave in a slightly different way, which lead to distinct error handling schemes.

When a FSB has the same level as the immediately previous bit, an error condition occurs. In such cases, it is important to determine how to categorize it, since it will impact the way it is signaled. From our perspective, the occurrence of an error in these conditions can be catalogued in three possible ways:

1. As a stuff error – following the idea that every change to the expected polarity of stuff bits should be included in the same group.
2. As a CRC error – based on the principle that each error within the CRC field should follow the same categorization, giving other nodes the opportunity to make themselves heard about the integrity of the received message (ACK slot).
3. As a form error – if they are perceived as frame control bits, in a perspective that they help to determine if the frame was correctly produced and if there is not lack of synchronization between sender and receivers.

From a strictly operational point of view, the only scenario that entails a delayed error signalization is option number 2, because stuff and format errors must be made aware to the network immediately after its detection, both in CAN and CAN FD.

For a matter of standardization the CAN FD specification defines how each error type should be handled. However, it is doubtful regarding the categorization of fixed stuff bits errors. In the form-error category, in section 6.1 - ERROR DETECTION, where all error types are described, it is said that:

> "*When the value of a fixed STUFF-BIT in the CAN FD format CRC SEQUENCE is equal to its preceding bit, this shall also be detected as a FORM-ERROR.*".

However, in section 5 – CODING, the following statement emerges:

"A Receiver shall discard the fixed STUFF-BITS from the bit stream for the CRC check, it shall detect a STUFF-ERROR if the fixed STUFF-BIT has the same value as its preceding bit."

It is readily apparent that the two statements are naturally conflicting, since the same error cannot be handled in two different ways. As such, the CAN FD specification needs to be revisited in order to clarify this definition.

5.1.5 Dynamic Stuff Bits into CRC

There is a known weakness in the CRC check of the CAN protocol, which was reported and published in [10]. It involves the occurrence of, at least, two bit errors that by generating or eliminating stuff conditions end up reducing the Hamming distance to 2. This happens because a frame with two bit flips can appear to the CRC as having more than 5 erroneous bits, undermining the effectiveness of the entire mechanism. A simple example of this problem can constructed as follows:

- In the frame data field, one of the bits of a 5 bit sequence that originates a stuff condition gets inverted.
- Since the stuff condition is no longer there, receivers don't detect the original stuff bit and interpret it as a normal bit.
- Now, every bit that follows the masked stuff bit is shifted one position to the right - to the CRC, there will likely be more than 2 different bits between the two sequences. At this point, the frame also has one additional data bit.
- Another bit, ahead in the frame data field, also gets inverted causing a falsified stuff condition.
- The "created" stuff condition, ahead in the frame, will readjust the sequence to its normal size. Therefore, the frame format control mechanisms don't detect the frame extra length.

In the work developed by Eushian Tran [10], a CAN network was put to the test by observing its behavior under diverse error rates. The results of the experimental simulations reveal the scenarios which originate the over mentioned phenomenon and provide the basis to address this subject in a probabilistic manner.

The experimental study alone shows that, in a given CAN network, the error detection failure rate of a message with 2 arbitrary flipped bits is $1{,}25 \times 10^{-7}$, contradicting what is stated in CAN specification of 100% error detection up to 5 randomly corrupted bits (due to the corner case explained above). It is also notable that as the number of corrupted bits in a transmitted message increases, the error detection mechanisms become more effective. This is so because, in such cases, there is an increased likelihood that one of the flipped bits originates stuff or form errors. Additionally, another simulation was made to observe the impact of burst errors in the network.

In this particular case the results are in line with what is defined in the protocol specification (full detection of burst errors up to 15 bits).

Two distinct approaches to overcome this weakness are proposed in the paper: One by software development, comprising the inclusion of an additional safeguard into the message. Another by a modification of the hardware functionalities, encompassing a change in the CRC computation scheme. Although both solutions can accomplish the desired goal with a satisfactory level of effectiveness, the last one allows for a more elegant low-level approach and diminishes the urge to add additional software layers every time a problem is encountered, which will make systems over complex.

Problem solution and the origin of Fixed Stuff Bits

Aware of this deviant behavior, the CAN FD development team seized the opportunity to resolve the flaw by mere protocol design, computing the CRC after the message is stuffed instead of before. This change guarantees that one bit flip does not impact the other bits of the bit sequence fed to the CRC, even if the flipped bit compromises a stuff condition.

This conceptual decision also explains the need for the fixed stuff bits. In CAN FD, the traditional bit stuffing codification needs to be finished before the CRC sequence is computed, so that the last can include all the dynamic stuff bits that will be inserted in the frame. As so, the CRC field codification must only be made later on: by repeating the custom bit stuffing mechanism or by creating an alternative scheme (in this case, the fixed stuff bits). This is the main reason why the CRC bit stuffing codification needed to be revised in CAN FD.

5.2 Protocol Standardization – ISO CAN FD

The CAN FD protocol, designed and made public by Bosch, went through a process of standardization by the ISO task force, which detected a flaw that affected the level of reliability of the protocol. This issue and the correspondent solution were reported in [11]. The uncovered defect is a consequence of the decision of including the dynamic stuff bits into the CRC calculation. Therefore, we can conclude right away that the measure taken to remove a very particular corner case in CAN originated another one in CAN FD.

All in all, the presence of the dynamic stuff bits in the CRC calculation made it vulnerable to a specific fault type. In this case, there is a possibility that a single bit error passes undetected through all error detection mechanisms, including the CRC. This odd behavior is originated by a bad synchronization, which causes the shortening or lengthening of the bit sequence (by adding or removing one bit), altering the number of bits fed to the CRC and undermining its result. The fault scenarios described in the paper are the following:

- **Case 1**: If the fault happens inside a stuff condition, the total number of data bits will be correct because the stuff bit compensates for the incorrect reading (the error is not

identified by the frame form check). However, the number of bits fed to the CRC will not be the same for transmitter and receiver.

- **Case 2**: If the fault happens outside a stuff condition, the number of data bits will be incorrect by one bit and the number of bits fed to the CRC will not be the same for transmitter and receiver.

The frame's inability to detect such fault relies on two assumptions:

1. The frame delimiters, namely CRC and ACK delimiters, may not detect a 1 bit deviation between receiving and transmitter nodes, if this deviation takes place within a data-phase with a bit time considerably smaller than that of arbitration-phase.
2. If the receiving node computes the CRC from a sequence with less or more bits than the transmitter sequence, the result of the CRC check is not reliable and must be regarded as corrupted.

Even though these premises undoubtedly lead to the conclusion reached in [11] - that the flaw exists in CAN FD protocol - they are partially taken for granted by the authors during the entire exposition of the concerned fault type. To overcome this gap, an analysis of both assumptions was made and is presented next.

5.2.1 Missing evidencies

1 – Frame format check and loss of synchrony

The central point of this issue is that, in a CAN FD network, the two data rates can be so disparate that a one-bit synchronization fault occurred during the data-phase may pass undetected in the arbitration-phase. In [11] is mentioned that a ratio of 4, or larger, between the two data rates, is sufficient to trigger this phenomenon. As such, a factor of 4 will be used to illustrate the following scenario.

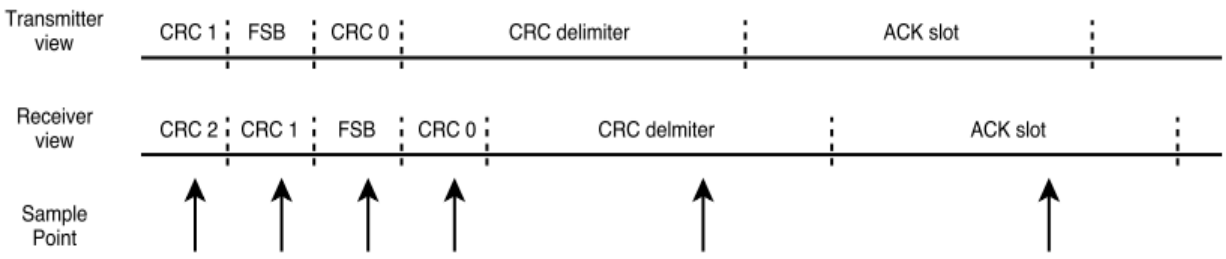

Figure 5.1 – Undetectable synchronization fault due to the dual data rate

The simple setup showed in figure 5.1 demonstrates how a 1 bit misalignment between two nodes can pass undetected by the CAN FD frame format checks. Let's assume that a synchronization fault does not lead to stuff or format errors until the end of the CRC sequence. In the example, the receiver's view is one bit behind the transmitter, due to a bad synchronization

that happened previously in the frame, within the data-phase, and caused one of the fault types described in [11] – shortening of the bit sequence. In this case, the difference between the two data rates causes the CRC delimiter bit to be sampled twice by the receiver, covering the lag between the two nodes. Since the bit has a recessive level, neither the receiver nor the transmitter will detect an error. From this point until the end of the frame, all the bus samples will be synchronized with the transmitters signal.

The larger the difference between the two data rates, the greater the probability of the network behaving as described here. The assumption made by the authors is a possible scenario and, therefore, has to be considered valid. Note that in CAN, this deviation would always be detected in one of two ways:

- By receivers, if the CRC delimiter was sampled during a dominant ACK bit.
- By the transmitter, if no receiver emitted a positive acknowledgement.

2 – Corrupted CRC

The second assumption is not so easy to validate. It relies on the inability of the CRC to guarantee the integrity of a message when it is applied to a different number of bits on transmitter and receiver side. The paper clearly states that, in such cases, an erroneous message can eventually lead to a positive CRC result, because the Hamming Distance only applies to strings of equal length. However, it lacks a more thorough exposition of the CRC behavior under these conditions or a reference to studies over this specific subject.

Starting from scratch, the HD is a metric that can only be used between strings of equal size. It represents the minimum number of required substitutions to turn one string into the other. When we are not in the presence of strings of equal length, the term Hamming distance should not be used because it has no practicality. As such, in our case, it is simply incorrect to state that the CRC has a HD = 5, when the transmitted and received bit sequences have different sizes. However, this does not necessarily mean that the CRC will not detect all bit flips up to 5 randomly distributed faults (typical CAN coverage). It simply means that we cannot use this metric to characterize its behavior.

One way of proving the validity of the assumption it's by finding at least one case where a shortening or lengthening error leads to a positive CRC result in the receiver's side. This would mean that the CRC check is unable to provide the publicized coverage upon the presence of this specific fault type. This task implies that the following steps are repeated until a positive CRC result is achieved:

1. Creation of a CAN frame;
2. Calculation of its CRC;
3. Induction of a shortening or lengthening error into the bit sequence;
4. CRC check.

Obviously, the effort of manually performing these operations is overwhelming. As so, to overcome this difficulty a Java program that performs the above sequence of steps was built. The results of the simulation show that there are several cases where the CRC sequence does not provide the traditional coverage when the two bit sequences have different lengths. Since our motivation is to prove the veracity of the Corrupted CRC statement and not to profoundly study the behavior of this error detection mechanism in such context, only one example is presented.

Let's consider the following CAN Base data frame[5]:

- Arbitration field: 0000000000000
- Control field: 000100
- Data field: 00110111011100010100010011110010
- Polynomial: 1100010110011001
- CRC sequence: 001100010010001

Note that the CRC sequence presented above is the result of multiple XOR operations between Arbitration + Control + Data fields and the Polynomial. Now let's assume that there is a shortening fault in the 13th bit of the frame bit sequence. This incident results in the following bit sequence being presented to the receiver:

- Arbitration field: 0000000000000
- Control field: 000100
- Data field: 00110111011100101000010011110010
- CRC sequence: 001100010010001

If the same polynomial is applied to this erroneous bit sequence, the result of the CRC check will be positive:

- Polynomial: 1100010110011001
- CRC sequence: 000000000000000

Realistically speaking, the behavior of the CAN protocol would not be exactly as described here, because the receiver expects one more data bit and, consequently, starts to interpret the CRC one bit later than supposed. Such event would eventually change the result of the CRC check. On the other hand, if we transpose the problem to CAN FD (case 1) the over mentioned situation would be masked by a stuff condition and wouldn't take place. Nevertheless, it is clearly demonstrated here that a lengthened or shortened bit sequence can undoubtedly originate a failure in this error detection mechanism which proves the author's statement.

[5] Bit stuffing is left out of the example since it has no influence over CAN CRC calculation.

Contrary to the Hamming distance, the Levenshtein distance[6] is an appropriate metric to use between strings of different size, because it also considers insertion and deletion operations, in addition to the HD substitutions. However, to our knowledge, no prior work that addresses the effectiveness of the CRC polynomials over distinct size strings has been made.

5.2.2 Article overview

Given that both assumptions are legitimized, it is pertinent to raise two topics for discussion: one about a fault scenario described in the article; and another about the completeness of the overall study.

1 – Incorrect fault scenario

According to the paper [11], there are two possible ways for a bad synchronization to generate an undetected error in non-ISO CAN FD. One of them involves the presence of a stuff condition and the other its absence. Although the first case did not raise any doubts about its rightfulness, the same cannot be said about the second, which reads as follows (page 6 of [11]):

> *"Case 2: If fault type B does not occur at a stuff condition, this does change the number of bits fed to the CRC algorithm in the receiving node by adding or removing a bit. The CRC calculation is corrupted. Additionally, the receiving node's view is shifted by one or more bits. If the ratio of data phase to arbitration phase bit-rate is large (e.g. 4 or larger), it is possible that the frame format checking at the end of the frame (CRC delimiter and later) does not detect this shift, because the shift is very small compared to the duration of an arbitration phase bit time."*

The underlined part of this quote conceals a mistake in the author's analyses, because the underlying principle of the statement is wrong: Contrary to case 1, if fault type B (shortening or lengthening of the bit sequence) does not occur at a stuff condition, it does **NOT** change the number of bits fed to the CRC algorithm in the receiving node and, consequently, the CRC calculation is **NOT** corrupted, but shifted instead.

[6] The Levenshtein distance between two strings is defined as the number of bits we must change to turn one into the other, considering insertion, elimination and substitution operations.

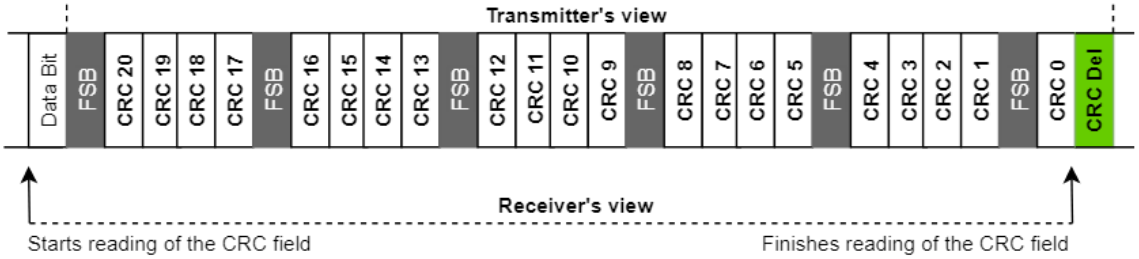

Figure 5.2 – Misaligned receiver view over the CAN FD 21-bit CRC field

As exemplified in figure 5.2, when the specified fault takes place outside stuff conditions, it shortens or lengthens the entire sequence in one bit (the example shows a shortening fault). However, the receiver is not aware of this displacement and starts to interpret the CRC field one bit sooner or one bit later than its first actual bit. Similarly, it finishes the CRC reading one bit earlier or one bit later than the actual CRC delimiter. This means that the CRC check, on the receiver's side, is made over the same number of bits than in the transmitter's one, invalidating the error scenario accounted in the paper of a corrupted CRC.

As for the possibility of the error derived from this faulty setup going unnoticed, it would have to meet the following conditions:

- Get a positive CRC result – the XOR result of the *shifted* bit sequence with the CRC polynomial would have to match the *shifted* CRC sequence;
- All the *shifted* FSB would need to have the opposite polarity of the immediate preceding *shifted* bit ;
- There could not be a dominant state on the bus during the reading of the *shifted* CRC and ACK delimiters.

Although there isn't any analytical or experimental study that reveals the probability that such event takes place, it seems to be of extraordinary difficulty for an erroneous frame to meet all the necessary criteria to pass unnoticed through all these verifications.

2 – Failure rate and use cases impact

The analyzed article explains with detail the possible error scenarios and correctly exposes a flaw in the error detection mechanisms of the non-ISO version of the CAN FD protocol. However, it lacks some insight about what is their actual failure rate. In the prior analyzed work [10], Eushian Tran tried to assess the real impact of a specific fault type in a CAN network by means of an experimental approach. Its results lead to the conclusion that there was an effect on the long-term operation of the network that should not be despised. Furthermore, he predicted that in a real situation (a car fleet) it was very likely that the fault would happen more than what

should be tolerable. Following the same rationale, it is advisable to submit a CAN FD network to a similar experimental simulation that allows a probabilistic assessment of the addressed error cases, in order to understand what its true impact on real usage scenarios is. Moreover, this type of study would be a decisive factor in the decision to correct the problem or not.

On the other hand, the authors take refuge in an undoubtedly relevant fact: The CAN FD protocol must have, at least, the same error detection capabilities (read reliability) of its predecessor (100% probability of detecting 1 bit errors) in order to impose itself as an improved version of classic CAN. This commitment requires correcting the error regardless of its likelihood. Nevertheless, the missing analyses should be taken into account as further work for those who want to endow the published study with additional information.

5.2.3 Adopted solution – Yet another modification

The CAN FD standardization process presents itself as the perfect opportunity to correct the flaw that affected its reliability. As such, the ISO CAN FD version of the protocol includes the necessary modifications to address this concern.

Stuff counter

To settle this issue, a counter of the number of transmitted dynamic stuff bits was introduced in the frame. The idea is to provide to the receiver the overall frame length – given by the frame type, DLC and stuff counter – allowing it to detect any deviation from the expected value. The stuff counter holds the grey coded value of the number of dynamic stuff bits in the frame as well as a parity bit. This combination enables a node to detect up to seven lengthening or shortening errors that coincide with stuff conditions. The counter is placed at the beginning of the CRC field for a matter of engineering – it can only be generated after the traditional method of bit stuffing is finished and also needs to be safeguarded by the CRC sequence. The changes made to the CAN FD frame format are shown in figure 5.3.

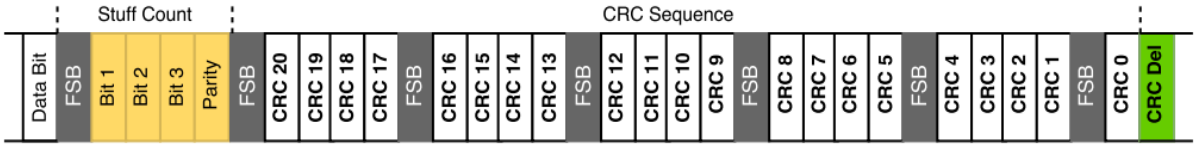

Figure 5.3 – ISO CAN FD 21-bit CRC field

The stuff counter has the downside of augmenting the protocol complexity and reducing its efficiency in three distinct ways:

1. Introduces 5 additional bits into the frame's length;
2. Creates the need for an internal counter to keep track of the identified stuff bits;
3. Performs an additional task of comparing the counted value with the transmitted.

Additionally, the stuff counter does not solve the error case that contemplates the occurrence of a shortening or lengthening fault outside a stuff condition, which was one of the two error scenarios presented in the article (despite, in this case, the protocol does not behave as assumed by the author, as explained in section 4.2.2). This fact is clearly stated in [11]:

> *"It is sufficient to transmit the stuff bit count modulo 8. With this it is possible to detect up to seven lengthening or shortening errors, if these coincide with stuff conditions. This is adequate as we consider a Hamming distance of 6 for CAN FD."*

Yet, the introduction of this additional safeguard reduces the overall probability that the error passes undetected through all the error detection mechanisms.

5.2.4 Related concerns

From a chronological perspective, the changes carried out to ensure a satisfactory level of reliability of the protocol, since the beginning of its development, can be synthetized as follows:

- A two bit error that reduces the CRC Hamming Distance to 2 is discovered in the classical CAN protocol. Two solutions are proposed at this point – one by software and another by hardware modification;
- The CAN FD development team, aware of this flaw, decides to welcome the proposed hardware solution;
- Because of the latest modification, the new protocol is now exposed to a single bit error that can go undetected through all error detection mechanisms;
- The solution to this issue involves the presence of an additional safeguard in the frame, which is included in the ISO CAN FD protocol version.

The key point of this scenario is that the solution to the first problem ended up involving two distinct changes to the protocol, as it was concluded that the initially proposed hardware approach had a side effect that had not been foreseen. Two questions must be raised at this point:

1. Since it became necessary to enter the stuff counter into the frame, could the CAN FD protocol return to the traditional method of bit stuffing, removing the fixed stuff bits? That is, does the counter prevent the occurrence of the initial fault type?
2. Is it possible to eradicate both faults with just one modification, or better, resolve the first one in a way that does not lead to the second correction?

The answer to the first question is no. The stuff counter would not be able to detect the fault type predicted in CAN protocol, simply because its occurrence did not change either the number

of data bits or stuff bits in the frame. As such, the count would give positive feedback even in the presence of the error.

The second question is very relevant because it casts doubt on whether the choice for the hardware solution, in the original problem, was the right one. To give a proper response, it's necessary to understand if the alternative suggested by Eushian Tran [10] (software approach), or other alike, would lead to a more simple, efficient and equally effective solution.

In [10], three different software solutions were submitted to tests in a controlled simulation to evaluate their behavior. They all contemplated the inclusion of an additional check, which occupied 8 or 16 bits of the data field, depending on the level of reliability to be achieved. The first was another CRC field; the second a simpler checksum; and the third a Longitudinal Redundancy Code (LRC) . From the performed test results, the approach that seemed to better detect the induced fault type was the checksum, most likely due to its different mathematical basis. In addition, it is less expensive to compute a checksum than a second CRC.

One of the concerns about the software solution in CAN protocol, was that it would have to make use of the ending part of the data field, limiting the payload size. However, during the development of CAN FD, it could have been created an appropriate space between the data and CRC fields for this purpose. This modification would have a similar impact in the frame's length than the hardware solution – fixed stuff bits and stuff counter.

The other concern is that this approach does not entirely eliminate the error, but instead reduces its probability of passing undetected. Eushian Tran simulation results show that an 8-bit checksum augments the detection of the erroneous messages that pass through the CRC in two orders of magnitude. A 16-bit checksum, in its turn, detects every corrupted message. Bridging these findings with the current protocol state, it is worth remembering that the changes made during its standardization don't guarantee 100% detection of single bit faults, because the stuff counter only fixes one of the two recently uncovered error scenarios.

In sum, both alterations lead to the inclusion of additional software verification and they both increase protocol complexity and frame size. One modifies the hardware functionality - namely the bit stuffing process during the CRC field - and neither of them fully meets the intended goal of removing the fault case from the spectrum of possibilities. Based on this information it is not clear which of the two solutions is most profitable, since both have very similar negative and positive aspects.

Figures 5.4 and 5.5 show the evolution of the CAN protocol, from a chronological perspective, emphasizing the discovered faults types and their solutions. The figures address the hardware and software approaches as proposed in [10].

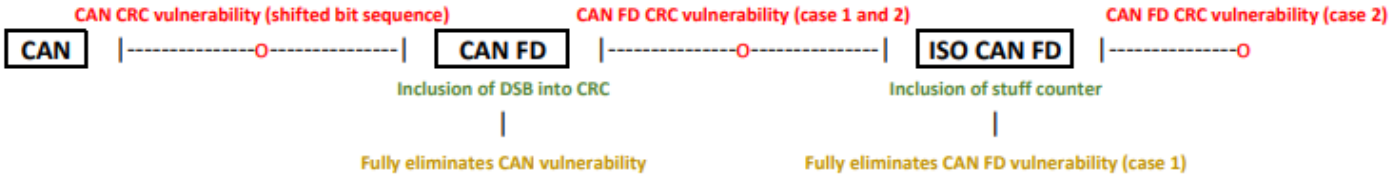

Figure 5.4 – Hardware approach to CAN CRC vulnerability

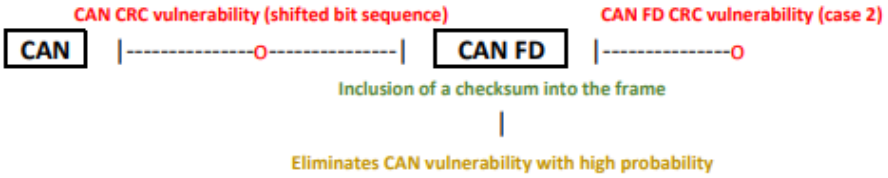

Figure 5.5 – Software approach to CAN CRC vulnerability

5.2.5 New fault type – Combination of the previous cases

During the evaluation of the recent changes made in the CAN and CAN FD protocols, there is an additional type of failure that appears to have passed unnoticed by the ISO task force, which affects all versions of the protocol.

The failure comprises a combination of the two previously examined fault types that affected CAN and non-ISO CAN FD – A two bit error that slips through the CRC check and a shortening or lengthening of the bit sequence – and is not covered by the inclusion of the DSB into the CRC calculation nor by the stuff counter, reducing once more the Hamming Distance to 2.

This particular case is originated when a bad synchronization leads to the shortening of the bit sequence, which is later compensated by a lengthening error (the inverse situation is also possible). It can happen inside or outside stuff conditions. In both cases, the bits located between these two faults will be shifted by one position, meaning that there can be more than 5 altered bits fed to the CRC, thereby jeopardizing its effectiveness (HD = 6). In sum, the same issue that compromised CAN reliability (a shifted bit sequence) and was resolved with the inclusion of the DSB into the CRC calculation can still take place if the bit flips are replaced by shortening and lengthening faults instead. Note that the detailed description of how a shortening or lengthening of the bit sequence occurs was already presented in [11] and will be omitted from Figure 5.6.

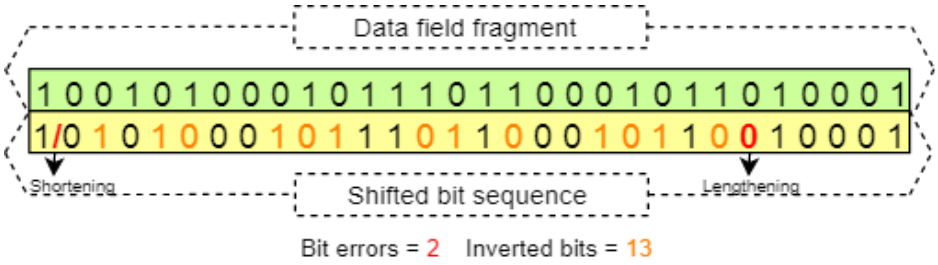

Figure 5.6 – Shifted bit sequence caused by shortening and lengthening faults

There are some characteristics of this fault type that make it imperceptible not only to the older but also the newer error detection mechanisms:

- The inclusion of the DSB into the CRC calculation fails to detect this errant behavior because, in this case, the receiving node is actually reading one bit more or one bit less than the transmitted stream, rather than confusing a data bit with a stuff bit or the opposite.
- In its turn, the stuff counter is completely ineffective because the combination of the two faults doesn't change the total number of stuff bits in the frame.

Additionally, the fault location inside a frame can determine whether the error passes or not undetected. Faults inside the data field are easier to conceal than those in arbitration or control fields. For example, if a lengthening fault occurs during the DLC, outside a stuff condition, it may lead the receiver to incorrectly perceive the data field length, probably causing a frame format violation ahead in the bit sequence. Also, the inverted bits in the shifted bit sequence will be the ones that comprise a change in the current bus level – where the bus passes from a dominant to a recessive state, or the opposite.

The uncovered weakness in the error detection mechanisms of both protocols depends on the occurrence of shortening and lengthening faults. As such, it is mandatory to understand in what circumstances they might take place and what kind of environment increases their likelihood. In [11] it is referred that:

> "*The shortening and/or lengthening may happen several times per frame. Inside one frame either shortening or lengthening is likely to happen. Which one is more likely is determined by the relation between the transmitter's and the receiver's clock rates.*"

Furthermore, how CAN and CAN FD error detection mechanisms react to the presence of this specific fault type should be a subject of study, in order to get an accurate indicator of their actual failure rate (again, remembering the work developed in [10]). For now, the absence of this information represents a tough obstacle when measuring the adherence of this perfectly valid fault

type, in a theoretical plan, to the reality of network operation across the wide and diverse spectrum of potential implementations.

One can argue that CAN also didn't guarantee the detection of every two bit errors in a frame. Still, the changes made to protocol, namely the inclusion of the DSB into the CRC and the stuff counter, come to allegedly return the Hamming Distance to 6. However, the reality is that not only the stuff counter doesn't cover all possible scenarios for single bit errors (only for one of the two identified cases) but there is also an additional unpredicted weakness for a particular two bit error setup.

5.3 Conclusions

CAN FD has undoubtedly brought improvements in terms of throughput and payload size. However, the set of alterations imposed by these improvements entail, in some cases, negative aspects. Throughout this chapter, a series of questions have been raised concerning the changes made to CAN and CAN FD protocols structure and operation. Some of them have been answered while others are still awaiting response and can be seen as a task for further work.

The recently introduced ESI bit can be of great use in higher level protocols that address issues like network membership and fault tolerant communication. On the other hand, the dual CRC polynomial has a slight perverse effect. Although its main purpose is to improve efficiency, it is doubtful whether or not it has the opposite impact in the overall network functioning.

The alternative way of stuff codification, which originated the concept of fixed stuff bits, is a consequence of the decision of including traditional stuff bits into the CRC computation – a measure taken to impede a corner case, which affected CAN's reliability, from happening in CAN FD. This solution proved not to comply with the intended goal of leveraging the error detection mechanisms coverage. Instead, it opened the door for another very particular fault type (shortening or lengthening of the bit sequence). In this chapter it is showed that although the referred article exposes a perfectly plausible fault case, one of the two presented error scenarios is slightly incorrect. In addition, the paper lacks an experimental study that sought to find out the actual failure rate of the error detection mechanisms when exposed to this fault type – crucial information when studying dependable real-time network deployments. Furthermore, the encountered solution, implemented in ISO version of the CAN FD protocol, only partially resolves the over mentioned issue.

An exercise is done in order to test in an alternative route that could also culminate in the removal of CRC flaw in CAN protocol, namely by including an additional software verification into the frame. However, it is difficult to come with a definite response to this matter, as there are positive and negative aspects in the two sides and both lead to similar results.

Finally, a new fault type that can pass undetected through all frame checks is identified and exposed. It encompasses the occurrence of 2 bad synchronizations that generate a shortening and a lengthening in the frame, shifting all the bits between them without changing the total frame length. In such case, the shifted bit sequence can have more than 5 different bits from the original one and lead to a positive CRC result. Its detection was made based on the combination of the two fault types studied in this chapter, which affected CAN and non-ISO CAN FD. Although its adherence to a real implementation setup remains to be demonstrated, it is a subject that justifies a more profound investigation.

6. Further Work

The inaccessibility study presented in chapter 4 is a first step towards a deeper understanding of the CAN FD protocol. Yet, there is still much ground to cover if we aim at achieving the same level of confidence we had in CAN protocol, which came as a result of all the academic investigation around it. To accomplish such a goal, all the relevant work made over CAN must be revised, rebuilding the studies that do not have a direct application in CAN FD, with special attention to the ones that can be most affected by the recently introduced changes addressed in chapter 5, like [13] and [14].

Although both CAN FD protocol versions have already been commercialized, it is still worth to completely understand the underlying issues that motivated some of the changes that were carried out. In addition, the recently uncovered fault type needs to be thoroughly analyzed in order to determine its real impact in network operation. All these matters can be addressed in a single study that involves the realization of an experimental simulation over the likelihood of specific fault types passing undetected, that is, the concrete failure rate of CAN FD error detection mechanisms under specific conditions.

We leave three suggestions that, in our opinion, should be in the scope of future work in this field:

1. What's the actual failure rate of the CAN FD error detection mechanisms in the presence of the error scenario identified in [11] (case 1), before the ISO protocol modification that comprised the inclusion of the stuff counter into the frame? Does it justify the encountered solution?

2. What's the prospect of the error scenario present in [11] (case 2) taking place and passing undetected, knowing that it has been reformulated in section 5.2.2 and is not settled by the introduction of the stuff counter into the frame?

3. Is the new fault type identified in this work a remote and unreachable situation or, on the contrary, it has significant impact in the long term functioning of a CAN FD network and must therefore be addressed?

7. References

[1] Robert Bosch GmbH. “CAN Specification Version 2.0.” September 1991.

[2] Robert Bosch GmbH. “CAN with Flexible Data-Rate Specification Version 1.0.” April 2012.

[3] CAN in Automation (CiA). *CAN FD - The basic idea.* https://www.can-cia.org/can-knowledge/can/can-fd/.

[4] J. Rufino, P. Veríssimo. “A Study on the Inaccessibility Characteristics of the Controller Area Network.” *2º International CAN Conference*. London, UK, October 1995.

[5] J. Rufino. “An Overview of the Controller Area Network.” *Proceedings of the CiA Forum CAN for Newcomers.* Braga, Portugal: CiA, January 1997.

[6] A. Mutter, F. Hartwich. “Advantages of CAN FD Error detection mechanisms.” *CAN in Automation iCC,* 2015.

[7] J. Kaiser, M. Mock. “Implementing the real-time publisher/subscriber model on the controller area network (CAN).” *2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing.* Saint-Malo, France, May 1999.

[8] J. Rufino, P. Verissimo, G. Arroz, C. Almeida, L. Rodrigues. “Fault-tolerant broadcasts in CAN.” *Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing.* Munich, Germany, June 1998.

[9] H. Zeltwanger. “CAN FD: Improved residual error-rate.” *CAN Newsletter 4*, 2014.

[10] E. Tran. *Multi-Bit Error Vulnerabilities in the Controller Area Network Protocol.* Thesis, Pittsburgh: Dept. of Electrical and Computer Engineering, Carnegie Mellon University, 1999.

[11] A. Mutter. “CAN FD and the CRC issue.” *CAN Newsletter 1*, 2015.