An Overview of the Controller Area Network

José Rufino
ruf@digitais.ist.utl.pt
IST - UTL [*]

Abstract

The Controller Area Network (CAN) is a communication bus for message transaction in small-scale distributed environments. Introduced for automotive applications plays nowadays a relevant role in the control and automation arena.

This paper provides an overview of CAN by addressing the fundamental issues concerning its physical and data-link layers.

1 Introduction

The Controller Area Network (CAN) [1, 2] is a communication bus for message transaction in small-scale distributed environments. Although originally designed for automotive applications, CAN has rapidly gathered a growing attention in the control and automation arena, where field-buses[1] play an important role.

CAN is a broadcast bus with a multi-master architecture able to provide real-time and fault-tolerance features extremely relevant for control and automation. CAN is structured accordingly with the OSI layered model, in a collapsed three-layer architecture. This paper provides an overview of the physical and data-link layers. Several options exist for the application layer: CAL - the CiA CAN Application Layer; CANOpen; SDS (Smart Distributed System), DeviceNet and CAN Kingdom [3]. Use of specialised proprietary protocols is also possible.

The paper is organized as follows: the next section presents the fundamental properties of CAN physical layer; section 3 discuses the data-link layer, namely the CAN Medium Access Control method , frame formats, coding and timing; the error handling mechanisms of CAN, of fundamental relevancy for the provision of consistency in data transfers, are described in section 4; CAN error confinement mechanisms are addressed in section 5 while important aspects concerning consistency of data transfers are further discussed in section 6.

2 Physical Layer

The Physical Layer (PHY) is responsible for the transfer of bits between the different nodes in a given network; it defines how signals are transmitted and therefore deals with issues like timing, encoding and synchronization of the bit stream to be transfered.

[*]Instituto Superior Técnico - Universidade Técnica de Lisboa , Avenida Rovisco Pais - 1096 Lisboa Codex - Portugal. Tel: +351-1-8418397/99 - Fax: +351-1-8417499. NavIST CAN WWW Page - http://pandora.ist.utl.pt/CAN.

[1]Real-time instrumentation networks, mainly designed for sensing and actuating.

In the original Robert Bosch CAN Specification [1], no media was defined thus allowing different options for the transmission medium and for signal levels. These properties were latter settled within the scope of the ISO standard [2] where bus signalling characteristics are defined. The CiA DS 102-1 standard [4] complements those definitions with regard to physical medium and connector specifications. Nowadays most of existing CAN implementations use a differential two-wire bus line.

The network operates in a *quasi-stationary* mode: at each bit transmission it is given enough time for signal level stabilization before sampling be performed almost "simultaneously" at all network nodes. That means bus capacity is equal to one bit. Because of the aforementioned stability requirement the network maximum length depends of the data rate. Typical values are: 40m @ 1 Mbps; 1000m @ 50 kbps.

Bit signalling on the bus line can take two possible representations: *recessive*, which only appears on the bus when all the nodes send recessive bits; *dominant*, which needs to be send only by one node to stand on the bus. This means that a dominant bit sent by one node can overwrite recessive bits sent by other nodes. This feature will be exploited for bus arbitration, as explained ahead.

Each bit is transmitted using the NRZ[2] code: this code displays a low spectral density thus allowing a good utilization of transmission medium bandwidth.

On the other hand, the standard CAN Physical Layer specification [2] defines built-in fault tolerance means, that allows network operation in harsh conditions, although with a reduced signal-to-noise ratio. Standardized fault tolerance is based on single-wire operation and it is able to ensure continuity of network operation in the presence of:

– one-wire breakage;
– one-wire short-circuit either to power or ground;
– two-wire short-circuit.

The standard specification does not provide any means to tolerate the simultaneous breakage of both wires in the bus line.

3 Medium Access Control Layer

The data link layer CAN communication paradigm relies on the dissemination of the so-called "communication objects". The data to be transfered is encapsulated within network level information packets and an unique identifier is assigned to each one of these "communication objects". The uniqueness of communication object identifiers avoids the need of using addressing information since it can be used to establish an unambiguous association between the object identifier and the meaning of the data it contains (e.g. temperature, pressure and humidity readings). However, nothing prevents the use of CAN as a traditional message-passing network.

Distinct network level information packets are used to provide (*data frame*) and to request (*remote frame*) the dissemination of communication objects. Two different lengths can be used for frame identifiers:

– standard 11-bit identifiers (**CAN 2.0A**);
– extended 29-bit identifiers (**CAN 2.0B**).

[2] Non-Return-to-Zero.

On the other hand, the uniqueness of communication object identifiers is also used to arbitrate bus access requests from competing nodes. The Controller Area Network [2] is a *carrier sense multi-access with deterministic collision resolution* (CSMA/DCR) network: nodes delay transmissions if the serial bus-line is busy; when a bus idle condition is detected, any node may start transmitting; bus access conflicts are resolved through the bitwise comparison of communication object identifiers and work as follows:

– while transmitting a communication object identifier, each node monitors the serial bus-line;
– if the transmitted bit is recessive and a dominant bit is monitored, the node gives up from transmitting and starts to receive incoming data;
– the node transmitting the object with the lowest identifier goes through and gets the bus.

That means: arbitration is *non-destructive*, since transmission of the object with the lowest identifier undergoes without delay; bus access is prioritised, allowing transmission of more urgent data to takeover less urgent one. Automatic retransmission of a communication object is provided after a loss in an arbitration process.

3.1 Frame Formats

As mentioned above two different lengths are allowed for data/remote frame identifiers. In order to maintain compatibility between these two definitions distinct frame formats are used. Let us start our analysis by describing the structure of a *standard* CAN frame (i.e. version 2.0A [1, 2]) (Figure 1):

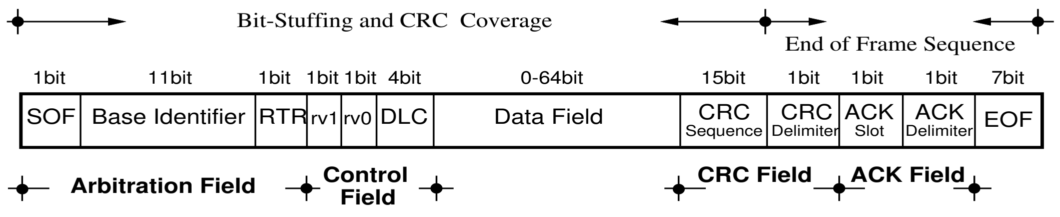

Figure 1: CAN 2.0A Data Frame Format

- **SOF** - *Start Of Frame Delimiter* - this field signals the beginning of the frame through the transmission of a single dominant bit.
- **Base ID** - the standard 11-bit sequence that, accordingly with CAN Specification 2.0A, uniquely identifies each communication object.
- **RTR** - *Remote Transmission Request* - this single bit field distinguishes data and remote frame types. In the former the RTR bit takes a dominant value; for remote frames this bit is set to recessive. Both the Base ID and the RTR field participate in the arbitration process. Therefore, accordingly with the previous definitions and for the same Base ID, data frames will take precedence over remote frames.
- **Control** - this field is six bit long and begins with two reserved bits that in CAN Specification 2.0A take always dominant values. It is followed by a four bit **Data Length Code** (DLC) that may assume any value in the interval [0,8] and indicates the number of bytes in the *Data Field*.

  In remote frames, there is no data field, but the Data Length Code should nevertheless take a consistent value for all the nodes requesting the transmission of a given communication object. Otherwise, in the presence of simultaneous remote transmission requests for the same object identifier, an unresolved collision will subsist in the network.

- **Data Field** - this field will hold the data to be transfered. Its size varies between zero and eight bytes.
- **CRC Field** - consisting of the following two elements:
  - **CRC Sequence** - a 15-bit cyclic redundancy code (CRC) used by receivers to check the integrity of incoming frames.
  - **CRC Delimiter** - a single bit always set to a recessive value.
- **Acknowledge Field** - used by receivers to acknowledge a correct data/remote frame transmission. It contains two distinct elements:
  - **ACK Slot** - a single bit element sent with a recessive value by the transmitter; its value is changed to dominant, by any receiver, upon reception of a frame without CRC errors.
  - **ACK Delimiter** - a single bit delimiter assuming always a recessive value.
- **EOF** - *End Of Frame Delimiter* - made from a fixed form sequence of seven recessive bits.

The size of CAN frame identifiers was extended in CAN Specification 2.0 Part B [1, 5] to a length of 29 bits in a way that compatibility with the previous definition was maintained. Let us now analyze the impact of such extension on data/remote frame structure (see Figure 2). The two formats are distinguished through a different utilization of the two bits that immediately follow the standard 11-bit base identifier. In the standard 2.0A format definition, the first one of these two bits corresponds to the RTR field. In the extended 2.0B format it is replaced by:

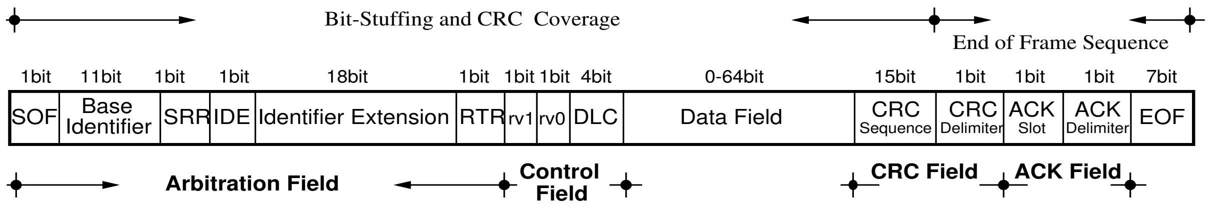

Figure 2: CAN 2.0B Data Frame Format

- **SRR** - *Substitute Remote Request* - a single bit always transmitted with a recessive value. Since the bit transmitted at this location in the standard CAN frame structure belongs to the arbitration field, that means CAN 2.0A traffic always prevails over any CAN 2.0B frame using the same 11-bit identifier.

The next bit to be transmitted belongs to the Control Field in the standard CAN frame structure; it is always sent with a dominant value. In the extended CAN frame structure the SRR bit is followed by:

- **IDE** - *Identifier Extension* - a single recessive bit is sent after the SRR bit in order to signal the presence of an identifier extension.
- **Extended ID** - a 18-bit sequence of additional identifier bits. It follows the IDE bit and precedes the RTR bit and the Control Field. In the extended CAN frame format these last two fields are fulfilled accordingly with the rules defined above for the standard CAN 2.0A frame.

3.2 Frame Coding

In order to avoid the transmission of long sequences of bits with identical polarity[3] outgoing data/remote frames are subject to a bit-stuffing coding methodology that prevents more than

[3] The absence of enough transitions in a bit stream will prevent an accurate synchronization of the clocks that control, at each receiver, the sampling of the bus line.

five consecutive bits of identical polarity to be transmitted, through automatic insertion of a complementary bit.

The bit-stuffing coding is performed from the start of frame delimiter until the end of the 15-bit CRC sequence. The CRC delimiter, the acknowledge field and the end of frame delimiter display a fixed form not subject to bit-stuffing encoding.

The use of bit-stuffing coding makes frame transmission durations dependent not only of its size but also of the its bit pattern. These issues are treated next.

3.3 Frame Timing

Several network parameters bound the time required to perform a given frame transmission, once the node gets access to the network after winning an arbitration process[4]. A fundamental parameter that influences the duration of a given frame transmission is:

◇ **Data Rate** - The nominal rate of data signalling, on the bus. It gives a meaning to t_{bit}, the nominal duration of a single bit.

Other parameters are the frame format specification (2.0A or 2.0B), the frame type (data or remote), the data field size[5]. For establishing a frame transmission time lower bound, one consider that no bits are stuffed in the outgoing stream[6]. Therefore:

$$t_{data}^{lb} = (l_{fix} + l_{data} + l_{efs}) \, . \, t_{bit} \tag{1}$$

◇ l_{fix} - represents the length (in bits) of fixed size fields subject to bit-stuffing. It includes the start of frame delimiter, the arbitration and control fields, as well as the CRC sequence. Its exact value depends on CAN frame format specification (2.0A or 2.0B).

◇ l_{data} - represents the length in bits of the data field. It varies between 0 and 64, in 8 bit increments.

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

On the other hand, for establishing an upper bound on frame transmission time, we consider that all the fields subject to bit-stuffing display a pattern that leads to the maximum insertion of stuffed bits. Therefore:

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

where $\lfloor \quad \rfloor$ represents the *floor* function[7]; l_{stuff} represents the stuff width, i.e. the maximum number of consecutive bits of equal value that can be found in an outgoing stream.

[4]Under controlled load conditions a bound can also be placed in the time required to get the access to the network [6, 7, 8].

[5]The data field size is always zero for a remote frame.

[6]Equations (1) and (2) do not account the minimum three bit bus idle period that mandatory precedes any data or remote frame transmission (intermission).

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

A more detailed analysis on data/remote frame timing variability, including a study on the probability of insertion of a given number of stuffed bits, can be found in [9]. The frame transmission time bounds, evaluated through application of equation (1) and (2) respectively to the shortest and to the longest frame sizes, are shown in Table 1. For completeness this table includes also error and overload frame timings. This network error handling packets will be described ahead.

Frame	Symbol	Duration (μs)			
		CAN 2.0A		**CAN 2.0B**	
		min.	*max.*	*min.*	*max.*
Data frame	t_{data}	44.0	132.0	64.0	157.0
Remote frame	t_{rdata}	44.0	52.0	64.0	77.0
Error frame	t_{error}	14.0	20.0	14.0	20.0
Overload frame	t_{oload}	14.0	20.0	14.0	20.0

Table 1: Duration of CAN frames (Data Rate = 1 Mbps)

4 Error Handling

An extensive set of error detection methodologies are used in CAN in order to enforce the reliability of communication object transfer. Furthermore, as a general rule, errors are signalled as soon as they are detected, a fundamental condition to obtain low error recovery latencies [10]. Error signalling is performed in CAN through the broadcast of special purpose data-link packets.

4.1 Error Signalling

Most of the abnormal network operating conditions are signalled, by the node detecting such condition, through the issuing of an error frame. An error frame begins with the transmission of an *error flag* and ends with an *error delimiter*. This last element always consists in a sequence of eight recessive bits. On the other hand, the error flag can either comprise a sequence of six consecutive dominant bits (*active error flag*) or a sequence of six consecutive recessive bits (*passive error flag*), depending on the node error status (see section 5). Error signalling is performed as follows:

- the node detecting the error and initiating the error signalling starts it by transmitting one *error flag*;
- if there are nodes that have not yet detected the error, the transmission of that error flag will, in most cases[8], lead to an error condition[9];
- therefore, all nodes respond to an error by starting their own transmissions of an *error flag*;
- at most, two *error flags* will flow in the network before the joint transmission, by all the nodes, of the corresponding *ending delimiter* (Figure 3).

Error frames are not subject either to CRC checking or to bit-stuffing coding. The corresponding best and worst transmission times are given by equations:

[8] Exception made when a *passive* error flag is issued by a receiver.

[9] Notice, for instance, that error flag pattern violates the bit-stuffing coding rule. Further details on error detection are provided in section 4.2.

$$t^{bc}_{error} = (l_{flag} + l_{del}) \cdot t_{bit} \tag{3}$$

$$t^{wc}_{error} = (2 \cdot l_{flag} + l_{del}) \cdot t_{bit} \tag{4}$$

where l_{flag} and l_{del} are, respectively, the lengths in bits of the *error flag* and of the *error delimiter*.

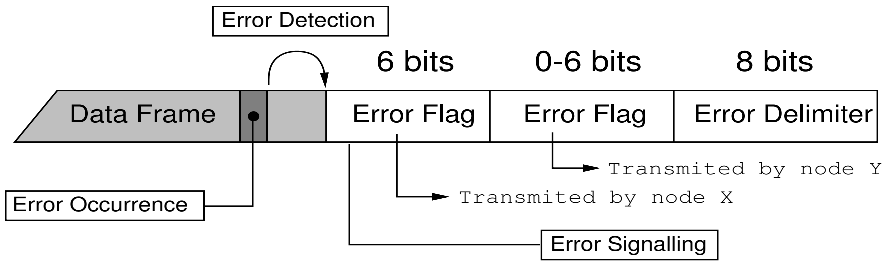

Figure 3: CAN Error Frame Format

4.2 Error Detection

A set of different mechanisms are provided in CAN for error detection. These mechanisms include:

- **Bit Errors** - each node listens the bus while transmitting and compares both streams on a bit-by-bit basis. When the transmitted bit level does not match the monitored one, error signalling is started at the next transmitted bit. Bit-error checking is not performed in frame fields where transmitted/monitored bit levels are allowed to differ, i.e. inside the *arbitration field*, at the *acknowledge slot* and while sending a *passive* error flag.
- **Stuff Errors** - a sequence of more than l_{stuff} consecutive bits with identical polarity is detected in a data or remote frame, within the corresponding bit-stuffing encoding range (see Figures 1 and 2). Transmission of an error flag is started immediately after detection of the bit-stuffing code violation.
- **CRC Errors** - a frame heard without errors should be acknowledged to the transmitting node, by superscribing with a dominant level, the recessive *acknowledge slot* issued by the transmitter. Conversely, when a CRC error is detected, the receiver takes no action at the *acknowledge slot* and starts error signalling only at the bit following the *acknowledge delimiter*, unless an error frame due to another error condition has already been started.
- **Form Errors** - a receiver detects a value violation in a fixed form frame field. This includes: the 1-bit *CRC delimiter*, the 1-bit *acknowledge delimiter* and the 7-bit *end-of-frame* delimiter, in data and remote frames; the full error and overload frames. The transmission of the corresponding error flag is started immediately.
- **Acknowledge Errors** - a transmitting node does not get any acknowledgment from the receivers. Recognition of the error is restricted to the transmitting node. Error signalling is started at the *acknowledge delimiter*.

Besides the aforementioned errors, there are another set of conditions that prevents normal network operation during the time required for its recovery. They concern violation of the three recessive bit period (intermission) that mandatory precedes any data/remote frame transmission and are known, in CAN terminology, as *overload* conditions. Like other errors, overload conditions are also signalled to other nodes through the broadcast of a special purpose data link packet, known in CAN terminology as *overload frame*.

Overload frames have the same structure than error frames (i.e. an *overload flag* followed by an *overload delimiter*) and are broadcasted through a similar method. The only difference regards the overload flag: it always consists of six consecutive dominant bits.

The need for overload signalling can be due to external reasons: detection of a dominant bit during intermission; a receiver detects a dominant bit at the last bit of the *end of frame* delimiter. Overload signalling can also be performed under "*request*", due to the need of an extra delay by the internal receiver circuitry.

A fundamental difference between errors and overload conditions is that the latest ones do not influence the operation of error confinement mechanisms (to be addressed in section 5).

A comprehensive study of CAN performance in the presence of errors, that includes all the above scenarios, was presented in [11]. The results of that study are summarized in Figure 4.

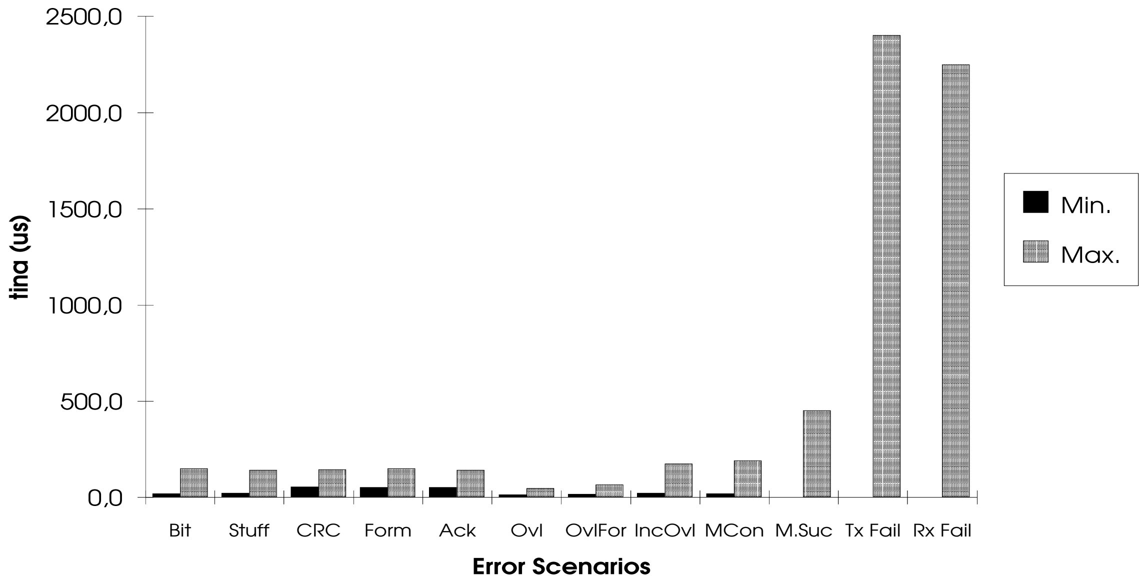

Figure 4: CAN Error Recovery Latency (Data Rate = 1 Mbps)

5 Error Confinement

The error confinement mechanisms provided by the CAN protocol aims to distinguish between temporary errors and permanent failures as well the shutdown of defective nodes. These mechanisms are based on two different error counters recording, at each node, transmit and receive errors. These counters have a non-proportional update method, with each error causing an increment larger than the decrement resulting from a successful data or remote frame transfer. Details on *Transmit* and *Receive-Error-Count* update procedures can be found in [1, 2].

The rules used in error counting have been defined in order that nodes closer to the error-locus will experience, with a very high probability, the highest error count increase. This way, disturbances due to a faulty node can be localised and their influence restricted, accordingly with the following classification:

Error-Active - the normal operating state. Such a node is able to transmit and receive frames and fully participates in error signalling.

Error-Passive - the node is still able to transmit and receive frames, but: after transmitting a data or remote frame the node requires an extra eight bit bus idle period following the *intermission*, before it can start a new transmission[10]; error signalling is performed by issuing a *passive error flag*, meaning it has only guarantees to succeed if the node was transmitting.

Bus-Off - a node in this state does not participate in any bus activity, being unable to send or receive frames.

A node enters the "error-active" state after power-on initialization with both error counters set to 0. It will become "error-passive" if any error counter exceeds 127. An "error-passive" node goes back to "error-active" when both counters are equal to or lower than 127. Conversely, an "error-passive" node goes "bus-off" after its transmitter error counter exceeds 255. A node in the "bus-off" state is allowed to become "error-active" after a reset (that sets to 0 both error counters) and after 128 occurrences of 11 consecutive recessive bits have been monitored on the bus line.

6 Data Transfer Consistency

The set of error detection mechanisms described in section 4.2 aims to ensure consistency of data transfers among "error-active" nodes. Even under the assumption that no transmitter failure occurs, it is required an extra condition to ensure that behaviour:

Frame Validity Rule - for a data/remote frame transfer be considered valid by a transmitter it is required that no error occurs until the end of the corresponding *end of frame* delimiter. On the other hand, a receiver considerer that transfer valid if there is no error until the last but one bit of the *end of frame* delimiter.

Should the transmitter and the receivers obey to the same validity rule, an error in the last bit of a frame seen only by a set of receivers but not by the transmitter, will lead to an inconsistent transfer. The above rule allows receivers to accept frames whose transfer is disturbed by such errors, thus ensuring consistency in the absence of transmitter failure.

However, because of the aforementioned rule, there is the possibility that duplicates of the same frame could be received, by the set of receivers detecting no error until the last but one bit of a given data/remote frame transfer. Furthermore, if after such an error, the transmitter fails before being able to successfully complete the frame transfer, that transfer will stay inconsistent. Although the occurrence of such events display a low probability value, the design of high layer protocols should take it into account.

7 Conclusion

In this paper we have presented the fundamental characteristics of CAN. This broadcast bus with a multi-master architecture aims the transaction of messages in small-scale distributed environments. Because of its real-time and fault-tolerance capabilities, CAN has gained a wide acceptance in a large number of application areas: automotive, astronomy, agriculture, biochemical, medical systems, robotics, building and industrial automation.

At the Instituto Superior Técnico, in the Technical University of Lisbon, we are using CAN as a fundamental block for building reliable real-time distributed control systems.

[10] An action known in CAN terminology as *Suspend Transmission*. If during this period another node starts transmitting, the suspended CAN node will become receiver of that frame.

References

[1] Robert Bosch GmbH. *CAN Specification Version 2.0*, September 1991.

[2] ISO. *ISO International Standard 11898 - Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication*, November 1993.

[3] K. Lennartsson. Fundamental parts in SDS, DeviceNet and CAN-Kingdom. In *Proceedings of the 2nd International CAN Conference*, London, England, October 1995. CiA.

[4] CiA - CAN in Automation. *CAN Physical Layer for Industrial Applications - CiA/DS102-1*, October 1992.

[5] ISO. *ISO International Standard 11898 - Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication - Amendment 1*, April 1995.

[6] K. Tindell and A. Burns. Guaranteeing message latencies on Controler Area Network. In *Proceedings of the 1st International CAN Conference*, Mainz, Germany, September 1994. CiA.

[7] K. Tindell, A. Burns, and A. Wellings. Calculating Controller Area Network (CAN) message response times. In *Proceedings of the IFAC Workshop on Distributed Computer Control Systems*, Toledo, Spain, September 1994. IFAC.

[8] K. Zuberi and K. Shin. Non-preemptive scheduling of messages on controller area networks for real-time control applications. In *Proceedings of the IEEE Real-Time Technology and Application Symposium*, pages 240–249, Chicago, Illinois, USA, May 1995. IEEE.

[9] L. Rauchaupt. Performance analysis of can based systems. In *Proceedings of the 1st International CAN Conference*, Mainz, Germany, September 1994. CiA.

[10] P. Veríssimo, J. Rufino, and L. Ming. How hard is hard real-time communication on field-buses? Technical Report NavIST, Instituto Superior Técnico, Portugal, December 1996. (submmited for publication).

[11] J. Rufino and P. Veríssimo. A study on the inaccessibility characteristics of the controller area network. In *Proceedings of the 2nd International CAN Conference*, London, England, October 1995. CiA.