A CAN hub with improved error detection and isolation

Manuel Barranco[1], Julián Proenza[1], Guillermo Rodríguez-Navas[1] and Luis Almeida[2]

1DMI - Universitat de les Illes Baleares, Spain,

2LSE-IEETA/DET – Universidade de Aveiro, Portugal

Distributed embedded systems that require real-time performance need a network capable of deterministic access delay. CAN is one such network that became widespread in recent years due to its electrical robustness, low price, and priority-based access control. However, its use in safety-critical applications has been controversial mainly due to dependability limitations that arise from its bus topology, e.g. the existence of many possible points of failure. In this paper we propose and present a star topology that exhibits improved fault diagnosis and isolation mechanisms with respect to other commercially available hubs. Our hub[1] is fully compatible with existing CAN controllers but requires double links. The paper presents a prototype implementation configurable with 4 to 16 ports, describes its architecture and presents some performance results.[2]

1. Introduction

The Controller Area Network (CAN) protocol is a fieldbus that fulfills the communication requirements of many distributed embedded systems. In particular, CAN provides high reliability and good real-time performance with very low cost. Due to this, the CAN protocol is nowadays used in a wide range of applications, such as factory automation or in-vehicle communication.

Nevertheless, since CAN relies on a bus topology, the structure of a CAN network presents multiple components that are connected to each other without proper error containment. Therefore, a bus topology presents multiple components such that a single fault in any of them may make impossible the communication with more than one node. In particular, note that a bus topology presents multiples single points of failure. In this way, it is enough that one of them fails to lead to an entire network failure. For the objective of our work, we define *severe points of failure* (which include single points of failure) as those ones whose failure permanently affects the communication capabilities of two or more nodes, i.e. whose failure causes a *severe communication failure*. We have designed and implemented a new star topology (CANcentrate) whose central element, a hub, is provided with the proper error-containment mechanisms for making impossible the occurrence of *severe communication failures*. With our star, we reduce all the multiple severe points of failure present in a bus to one single point of failure: the hub. Even so, we consider our star topology a good solution since it is easier to deal with one single point of failure than with many of them.

In the following section we discuss the properties of existing solutions to improve the dependability properties of CAN, focusing on the advantages of a simplex star topology with respect to simplex and replicated bus topologies. Moreover, existing work on star topologies for CAN is also presented. Section 3 presents the architecture of CANcentrate. A prototype implementation of CANcentrate is described in Section 4. Section 5 presents future work and Section 6 summarizes the paper.

2. Bus versus star topology

In order to better understand how our star topology prevents the existence of multiple

[1] The contents of this article have been the subject of a patent filing submitted on the 16th of September of 2004.

[2] This work was partially supported by the European Commission through the Network of Excellence ARTIST2 (IST-004527).

components in the network such that a single fault in any of them may cause a severe communication failure, it is necessary to specify the different kinds of faults that may happen in the components of a CAN network. These kinds of faults are three. On the one hand, *stuck-at-dominant* faults and *stuck-at-recessive* faults that occur whenever a given element (a node or a medium) issues a constant dominant bit or a constant recessive bit respectively. Notice, however, that only stuck-at-dominant faults may cause a severe failure. On the other hand, *bit-flipping* faults occur whenever a given element has an uncontrolled behavior in the value domain and constantly sends arbitrarily erroneous sequences of bits. See [1] for a thorough discussion about these faults.

Some of these faults can be confined in bus-based CAN systems, up to a certain extent, using techniques that are already known. These techniques rely on the use of *replicated transmission media* (like the one proposed in [2]) as well as on the use of *bus guardians* [3]. However, even if these solutions are used together in the same system, they still allow multiple components to cause severe failures of the communication system. This is because common mode failures may affect both replicated media, as well as any bus guardian and its node. Thus, alternative solutions have been researched, namely those based on a star topology.

In a star topology, each node is connected to a central element, the *hub*, by its own *link*. The hub is a natural element to enforce the necessary error containment by isolating the appropriate hub's port when detecting a faulty link or a faulty node. Furthermore, the independence between the hub and the nodes is completely ensured. However, even though the star topology provides a good basis to improve the dependability of the communication system, the adoption of such a topology is not enough. Additional mechanisms should be included in the hub in order to diagnose and isolate faulty components that may cause a severe communication failure.

Some star topologies for CAN can be found in the literature, namely passive star topologies [4] [5], active star topologies [4] [5] [6], as well as the StarCAN protocol [7]. However, any of these available star topologies for CAN either do not address severe communication failures [4][5][7] or deal only with stuck-at-dominant faults [6]. Thus, almost all these star topologies behave as a bus with enhanced resilience to spatial proximity faults. Moreover, they present additional drawbacks. First, some of them impose strong bit rate and cabling length limitations (mainly passive stars and the stars presented in [6]). Second, StarCAN is not compatible with Commercial Off-The-Shelf components (COTS) and does not preserve the dependability properties of the CAN protocol. Third, the active star topologies presented in [6] even though addressing stuck-at-dominant faults, need a significant amount of time for diagnosing them.

In contrast, we have designed and built a new star topology, called CANcentrate, that overcomes these drawbacks and which focuses on achieving the goal specified above, i.e. to detect and isolate faulty components that may cause a severe communication failure.

3. Design of CANcentrate

In the CANcentrate architecture, each node is connected through a dedicated link to a different port of a central hub. In this way, a node together with its dedicated link constitute an error-containment region. From the hub perspective, a permanent fault within a given error-containment region manifests as a permanently faulty port. Therefore, the hub can prevent a severe failure by diagnosing and isolating any permanently faulty port. It is worth noting that the hub only diagnoses and isolates the kind of faults that may provoke a severe failure (stuck-at-dominant faults and bit-flipping faults). In addition, it is also able to detect stuck-at-recessive faults and includes a reintegration policy for re-enabling each port (previously isolated) after a period of inactivity is observed in this port.

Additionally, the preservation of all the CAN specifications is a requirement that

was imposed on the design of the hub. The preservation of the CAN specifications allows keeping all the CAN properties related to dependability, as well, as the use of COTS components.

The hub is divided into three modules, namely the *Input/Output Module*, the *Coupler Module*, and the *Fault-Treatment Module*. The structure and interconnections of these modules are depicted in Figure 1.

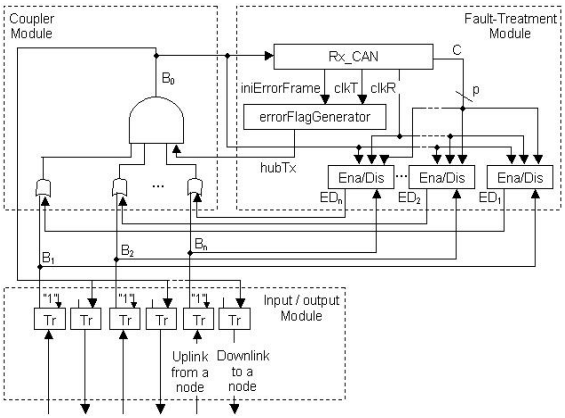

Figure 1. Internal structure of the hub

In order to preserve the CAN specifications, the hub replaces the wired-AND functionality of the CAN bus by an AND gate included in the so-called Coupler Module. This AND gate couples all the nodes contributions, $B_{1..n}$, and then the resultant coupled signal, B_0, is broadcast to all nodes.

In what concerns the fault-diagnosis mechanisms and the fault-isolation mechanisms, they are basically included in the Fault-Treatment Module. On the one hand, these mechanisms require the identification of the contributions from every node as well as to be able to independently isolate each contribution. On the other hand, in order to correctly understand (and thus to evaluate) the contribution of each node, the fault-diagnosis mechanisms also need the hub to be synchronized with the CAN nodes at both bit level and frame level.

The identification of each node contribution as well as the capability to independently isolate ports is achieved by using two independent links for each node. One uplink that carries the signal from the node to the hub, and one downlink that carries the coupled signal from the hub to the node. As depicted in Figure 1, the Input/Output Module includes two transceivers for each node that translate the physical signal into a logical form and vice versa.

Regarding the other requirement needed by the fault-treatment mechanisms of the hub, namely the synchronization at both bit level and frame level, the Fault-Treatment Module basically includes the Rx_CAN Module and the Error Flag Generator Module (errorFlagGenerator in the Figure 1). The Rx_CAN Module uses the basic reception mechanisms of a CAN controller [8] for allowing the hub to keep the synchronization at both levels. It generates the reception and the transmission clocks (clkR and clkT in Figure 1 respectively), as well as a set of signals, *C*, that describes the bit that is currently being observed at the coupled signal (e.g. whether the bit is a stuff bit or not; the location of the bit inside the frame, etc). Furthermore, in order to keep the synchronization in spite of errors that are detectable in the coupled signal by means of the error detection mechanisms of CAN, the Rx_CAN Module detects and forces the globalization of the same errors that would be detected by a CAN receiver. The error globalization is performed by the Error Flag Generator by means of an active error flag through the dedicated signal *hubTx*.

The *fault diagnosis* and *fault passivation* are carried out by the *Enabling/Disabling* units (*Ena/Dis* in Figure 1). On the one hand, each one of these units has a dedicated *event counter* and an associated *manager module* for each type of fault that must be detected (see Figure 2): the *Dominant Bit Counter* (DBC) and the *DBC Manager Module* for stuck-at-dominant faults; the *Non-Acknowledge Counter* (NACKC) and the *NACKC Manager Module* for the stuck-at-recessive; and the *Bit-Flipping Counter* (BFC) and the *BFC Manager Module* for the bit-flipping faults. Each management module basically analyzes the coupled signal, B_0, its corresponding port contribution, B_i, and the state signals, *C* from Rx_CAN, in order to decide how to increase or decrease its corresponding event counter.

On the other hand, each Enabling/Disabling Unit has a *Threshold Control Module* that is aimed at declaring the port as permanently faulty when appropriate as well as to isolate its contribution. The Threshold Control Module takes into account the value registered by each of its corresponding event counters and is programmed with a specific threshold for each of them: the *Dominant Bit Threshold* (DBT), the *Non-Acknowledge Threshold* (NACKT) and the *Bit-Flipping Threshold* (BFT). Whenever any of the event counters exceeds its corresponding threshold, the Threshold Control Module removes the contribution of this port from the system by issuing a logical '1' to the corresponding *Enabling/Disabling* signal, $ED_{1..n}$, which is connected to the OR gate (included in the Coupler Module) that corresponds to the faulty port. In general, this isolation mechanism based on the use of OR gates is similar to the one proposed in [2] to manage, locally in each node, the media redundancy in a replicated bus topology.

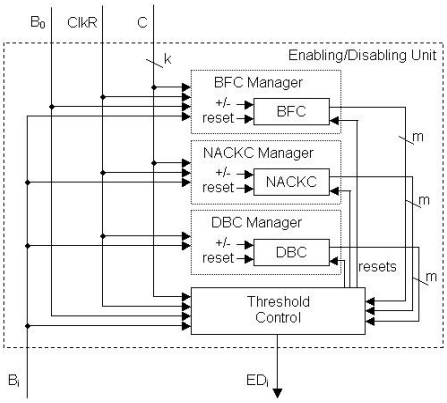

Figure 2. Internals of Enabling/Disabling Unit

Additionally, in order to increase the tolerance to transient errors, the Threshold Control Module may use a specific reintegration policy to re-enable the port contribution and to allow the operation of all managers again, after a given period of *inactivity* is observed at the port. As depicted in Figure 3, a port is in the *idle* state after the initialization of the hub or when the port is diagnosed as being stuck-at-recessive. When a port in the *idle* state sends any dominant bit during the arbitration, in the ACK slot or issues an active error flag, it enters the *active* state. The contribution of a port in the *idle* or in the *active* state is enabled. Note that a port diagnosed as being stuck-at-recessive enters into the *idle* state and its contribution is not isolated. This is because it does not generate errors that propagate to other ports. In contrast, whenever the port is declared as permanently being stuck-at-dominant or bit-flipping, the port enters the *disabled* state and its manager modules are reset. This actually implies that the port is isolated, as well as that the event counters are also reset. However, if during the *disabled* state, 127 bus free occurrences are detected, the port enters again the *idle* state and its contribution is re-enabled.

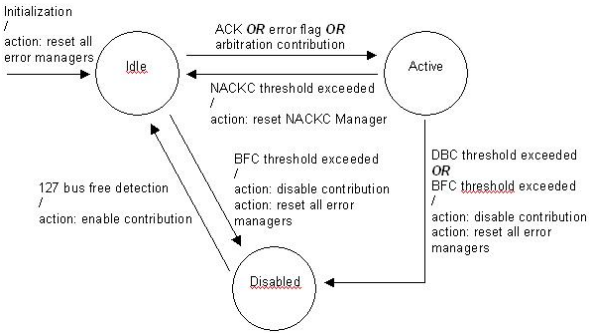

Figure 3. Reintegration policy

4. CANcentrate prototype implementation

This section is aimed at describing the basics of the first prototype of CANcentrate. The experimental platform that has been set up in order to test this prototype is also discussed. Finally the main results of these tests are presented.

4.1. Description of the prototype

The prototype is divided into several parts. Each of them corresponds to a given part or parts of the CANcentrate architecture. When building our prototype, we differentiated the following parts: the Coupler and the Fault-Treatment modules (referred hereafter as the *internal part* of the hub), the Input/Output Module, the links, and the CAN nodes. Next, a general description of the characteristics of each part of the implementation is given.

The *internal part* of the hub has been implemented using the VHSIC Hardware Description Language (VHDL) and the *Xilinx Virtex XCV300-PQ240* Field

Programmable Gate Array (FPGA), which is placed in the Xilinx prototype board *PQ240-100 Prototype Platform* (*HW-AFX-PQ240-100* version).

A dedicated board has been used for implementing the Input/Output Module. This board mainly contains four pairs of Philips PCA82C250 high-speed CAN transceivers and four RJ45 plugs (one plug for each pair of transceivers), so that up to four CAN nodes can be connected to the hub at the same time. The pin CANL (LOW level CAN voltage input/output) and the pin CANH (HIGH level CAN voltage input/output) of each transceiver are then connected to the appropriate pins of the corresponding RJ45 plug. The interconnection between the Input/Output Module and the *internal part* of the hub is made by means of a flat cable, which connects the specific reception and transmission pins of the CAN transceivers with the corresponding pins of the Xilinx prototype board.

One UTP (Unshielded Twisted Pair) Category 5/5e/6 Ethernet cable is used for implementing each link, which is constituted by an uplink and a downlink (as explained in Section 3). Both the uplink and the downlink use two-wire differential lines. The uplink uses the Transmit pair while the downlink uses the Receive pair of the Ethernet cable.

The CAN nodes have been implemented using *CANivete* boards, which are a previous development of the Universidade de Aveiro (UA) for standard CAN applications and implements a typical CAN node based on the Philips 82C592 micro-controller. A small modification was carried out to adapt the standard CAN interface of the CANivete to the double links of CANcentrate, adding a Philips PCA82C250 high-speed CAN transceiver and a RJ45 plug.

4.2. Experimental platform

The prototype of CANcentrate was tested to check its correct operation under error-free conditions and in the presence of faults, as well as to measure its performance. To perform these tests, an experimental platform was built. Specifically, the issues that were taken into account when devising this platform are the configuration of the application that is executed at the CAN nodes, the configuration of the network and the fault-injection mechanisms. Additionally, two requirements are imposed on this experimental platform: to achieve the maximum network utilization with a given bit rate, and to force an arbitration at the beginning of the transmission of each frame.

The first issue concerning the experimental platform is the configuration of the application that the CAN nodes execute. All CAN nodes run the same program, but with different sets of CAN identifiers. The program is constantly trying to send data frames with different identifiers and different data lengths, in order to test different frames. In addition, for fulfilling the requirement of achieving the maximum network utilization with a given bit rate, the program follows two basic rules: it must trigger a new transmission whenever it successfully transmits a frame and it must restart the CAN controller whenever, due to errors, it reaches the *bus-off* state.

With regard to the second issue of the experimental platform, namely the configuration of the network, it covers several aspects that are related to the nodes, to the links and to the bit rate.

Concerning the nodes, it is worth noting that at least three CANivete nodes are needed for fulfilling the requirement of forcing arbitration to take place in the transmission of each frame. This is because of the CAN controller within the 82C592 micro-controller.

The 4th port of our hub was used for fault-injection purposes as will be explained later in this section. Therefore, the network is configured with three non-faulty CAN nodes plus a port for fault injection.

Regarding the other aspects covered in the network configuration, the links and the bit rate, several Ethernet cables of different lengths, as well as different bit rates have been used in order to measure the performance of the network depending on the star diameter and the bit rate. Nevertheless, due to implementation limitations on the clock oscillators of the CANinvete nodes, the maximum bit rate

that has been used for testing the performance is 690kbit/sec.

Finally, the last issue related to the experimental platform is the set of fault-injection mechanisms that are used to validate the fault-treatment capabilities of the hub. As explained in Section 3, the hub is able to detect permanently faulty ports which present stuck-at-recessive faults, as well as to diagnose and isolate permanently faulty ports which present stuck-at-dominant or bit-flipping faults. Stuck-at-recessive faults can be easily injected by disconnecting the link of an operational CAN node from the hub. However, a more complex fault-injection mechanism is needed for stuck-at-dominant and bit-flipping faults.

For injecting these faults, a signal generator device was used to generate different bit stream patterns. Specifically, each bit stream pattern consists of a periodic signal that alternates from the recessive to the dominant value with a given frequency, thus injecting stuck-at-dominant and bit-flipping faults as explained in the following Section.

4.3. Functional tests

The correct operation of the prototype under error-free conditions, as well as in the presence of faults was checked by means of several functional tests. The aspects that have been tested under error-free conditions are the correctness of the:

- Operation of the different state machines that constitute the hub.
- Calculation of the coupled signal upon all node contributions.
- Correct synchronization at bit level and at frame level.
- Assignation of the roles of the nodes after the arbitration phase.

In contrast, the aspects that have been tested in the presence of faults are the correctness of the:

- Increase and decrease of the different event counters during different fault scenarios.
- Detection of ports suffering stuck-at-recessive faults, as well as the isolation of ports suffering stuck-at-dominant or bit-flipping faults.
- Reintegration of ports.

All the issues indicated above were tested at two different levels: at the level of the VHDL design of the hub and at the level of the physical network. However, each one of these two levels imposes different limitations. Thus, the different aspects listed above have been tested in different depths at the two levels.

The first level of testing, i.e. the functional test of the VHDL design of the hub, was done by means of the simulation tool *ModelSim XE II 5.7g* (provided by Mentor Graphics Corporation). Several simulations were done in order to check all the issues specified above and, in all cases, the operation of the hub was correct. Special attention has been paid to check the correct operation of the different state machines that constitute the hub, as well as their correct mutual interaction.

In what concerns the second level of testing, physical limitations in the layout of the FPGA board discourage an exhaustive test of all the state machines that constitute the hub. In contrast, many more fault scenarios can be injected at physical level than at simulation level.

For this physical level of testing different parts of the physical network have been observed by means of a logical analyzer and a digital oscilloscope. In particular, the ports of the hub were observed in order to know which is the contribution of each node as well as the value of the coupled signal. Since the Rx_CAN Module and the Enabling/Disabling units are key modules for synchronizing the hub at bit level and at frame level, as well as for diagnosing and isolating faulty ports respectively, they have also been observed.

For physically testing the correct operation of the network under error-free conditions (like during the phase of the tests of the VHDL design), the correct calculation of the coupled signal, the correct synchronization at bit and at frame level and the correct roles assignment during the arbitration phase have been checked.

With regard to the physical testing of the fault-treatment mechanisms the hub implements, extensive tests that include

stuck-at-recessive, stuck-at-dominant, bit-flipping faults and the reintegration policy have been performed.

Specifically, in order to physically test the actions carried out by the hub in the presence of stuck-at-recessive faults, the link of a previously operating node has been mechanically disconnected. When the link is disconnected a transient bit-flipping behavior is observed in its corresponding port. However, these erroneous bits are not enough for leading the hub to isolate the port. In contrast the contribution of the port quickly stabilizes to the recessive value and then, the hub indicates that it is stuck-at-recessive.

For physically testing the operations the hub performs in the presence of stuck-at-dominant faults, the signal generator was used to transmit a periodic signal that keeps the dominant value during many frames. It was observed that the hub correctly increases the appropriate event counter and isolates the corresponding port whenever the pertinent threshold is exceeded.

In what concerns the fault-diagnosis and fault-isolation operations the hub performs in the presence of bit-flipping faults, two kinds of techniques for injecting them have been used. On the one hand, a bit stream, which has random values, was injected by means of mechanically connecting/disconnecting a given link into its plug. On the other hand, the signal generator was used for injecting a bit stream that changes from the recessive to the dominant value with different frequencies that do not match with the bit rate the nodes use for communicating. In both cases, the beginning of the bit-flipping injection was randomly chosen. Many tests were performed with both techniques and in all situations the results were correct.

Finally, for the physical testing of the reintegration policy, the state of a given port was observed in different situations. After this port was isolated due to a stuck-at-dominant or a bit-flipping fault, a recessive value was forced in this port by disconnecting its link or by compelling its node to send recessive bits. In these cases, the hub re-enabled the contribution of the port, which agrees with the expected behavior related to the reintegration policy.

4.4. Performance measurements

In what concerns the performance tests, some measurements have been made. The values of the FPGA device utilization needed for implementing the hub prototype (with 4 ports) are: 758 out of 3072 slices, 278 out of 6144 Flip-flops, 1396 out of 6144 LUTs, 91 out of 170 IOBs, 4 out of 4 GLCKs. The extra delay introduced by the *internal part* is 35 ns, whereas the average value of the extra delay introduced by the entire hub is 155 ns. Notice that the value of the extra delay introduced by all the hub is of the order of *1/6* of the bit time when operating at the higher bit rate allowed in CAN [10] (1Mbit/sec). In addition, the *internal part* of the hub has been also built with 16 ports. The values of the FPGA device utilization in this case are: 2534 out of 3072 slices, 869 out of 6144 Flip-flops, 4662 out of 6144 LUTs, 91 out of 170 IOBs, 4 out of 4 GLCKs. It has been observed that the extra delay introduced by the *internal part* of the hub does not visibly depend on the number of ports it is provided with. Finally, several Ethernet cables of different lengths, as well as different bit rates have been used in order to measure the performance of the network depending on the star diameter and the bit rate. As said before, due to implementation limitations, the maximum bit rate that has been used is 690kbit/sec. At this bit rate, the maximum star diameter that was used without generating errors is 70 meters which implies a negligible reduction in length when compared with a CAN bus operating at the same bit rate (maximum length of approximately 80 meters) [10]. Moreover, remember that in a star the maximum length applies only to every pair of links, thus the star increases the capacity to interconnect nodes when compared with the bus topology [1].

5. Future work

Currently, there is on-going work to improve the fault-tolerant characteristics of CANcentrate. Specifically, we are

addressing both the issue of the replication of the hub to eliminate the single point of failure, as well as the integration of more fault-diagnosis mechanisms in the hub that may allow further restricting the failure semantics of CAN-based communication systems.

6. Conclusions

The use of CAN in safety-critical applications has been a controversial topic. This is due to a few factors such as the bus topology. In fact simplex bus topologies suffer from several impediments to enforce error containment while replicated buses may exhibit common mode and spatial proximity faults. On the other hand, star topologies may represent a positive step due to the key role that the hub can perform to diagnose and passivate faults. In fact, it allows reducing the number of components whose failure can cause a severe failure of the communication system, to a unique single point of failure, i.e. the hub.

We have designed and implemented an active star topology, called CANcentrate, that makes impossible the occurrence of severe failures. Additionally, our star is compatible with off-the-shelf CAN controllers and can be used with any CAN-based protocol (e.g. TTCAN [11], FTT-CAN [12], Timely CAN [13], MajorCAN [14], etc).

We briefly described the architecture of the central device of CANcentrate, a hub, which can be built using off-the-shelf FPGA technology.

Finally, we explained the implementation of a first prototype of CANcentrate. We checked the correctness of the fault-treatment mechanisms of the hub prototype and we evaluated the performance of CANcentrate. In particular, it has been observed that the extra delay introduced by the hub does not depend on the number of ports. Moreover, this extra delay implies a negligible reduction of length when compared with a CAN bus operating at the same bit rate.

References

[1] M. Barranco, G. Rodríguez-Navas, J. Proenza, and L. Almeida, “CANcentrate: An active star topology for CAN networks”, *WFCS'04. IEEE Workshop on Factory Communication Systems, Vienna, Austria*, 2004.

[2] J. Rufino, P. Veríssimo, and G. Arroz, “A Columbus' Egg Idea for CAN Media Redundancy”, *FTCS-29. The 29th International Symposium on Fault-Tolerant Computing, Winconsin, USA*, June 1999.

[3] I. Broster and A. Burns, “An Analyzable Bus-Guardian for Event-Triggered Communication”, in *Proceedings of the 24th Real-time Systems Symposium (RTSS)*, University of York, UK. Cancun, Mexico: IEEE, Dec 2003, pp. 410.419.

[4] M. Rucks, “Optical layer for CAN”, *1st International CAN Conference*, November 1994.

[5] CiA, “CAN physical layer”, CAN in Automation (CiA), Am Weichselgarten 26, Tech. Rep. [Online]. Available: headquarters@can-cia.de.

[6] IXXAT, “Innovative products for industrial and automotive communication systems”. 2005. [Online]. Available: http://www.ixxat.de/index.php.

[7] G. Cena, L. Durante, and A. Valenzano, “*A new CAN-like field network based on a star topology*”, Polytechnic Institute Torino Std. 23, July 2001.

[8] ISO, “ISO11898. Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication”, 1993.

[9] I. XilinX, “Virtex 2.5v field programmable gate array (product specification)”, 2001.

[10] CiA, “CAN data link layer”, CAN in Automation (CiA), Am Weichselgarten 26, Tech. Rep. [Online]. Available: headquarters@cancia.de

[11] H. Kopetz, “Time-Triggered Protocols for Safety-Critical Applications”, Presentation, Vienna University Of Technology, TU Wien, Karlsplatz 13, 1040 Vienna, Austria, March 2003.

[12] L. Almeida, P. Pedreiras, and J. A. Fonseca, “The FTT-CAN Protocol: Why and How”, in IEEE Transactions on Industrial Electronics – special issue on Factory Communication Systems, vol. 49, no. 6, December 2002.

[13] I. Broster and A. Burns, “Timely use of the CAN protocol in Critical Hard Real-time Systems with Faults”, in Proceedings of the 13th Euromicro Conferencs on Real-time Systems (ECRTS). IEEE, 2001, pp. 95.102.

[14] J. Proenza and J. Miro-Julia, “MajorCAN: A modification to the Controller Area Network to achieve Atomic Broadcast”, IEEE Int. Workshop on Group Communication and Computations, Taipei, Taiwan, 2000.

Manuel Barranco, mbarranco@det.ua.pt
Julián Proenza, julian.proenza@uib.es
Guillermo Rodríguez-Navas, guillermo.rodriguez-navas@uib.es
Systems, Robotics and Vision Group
Dep. Ciències Matemàtiques i Informática,
Universitat de les Illes Balears,
Palma de Mallorca (SPAIN)

Luis Almeida, lda@det.ua.pt
Dep. de Electrónica e Telecomunicações,
Universidade de Aveiro,
Aveiro (PORTUGAL)