Faculdade de Ciências da Universidade de Lisboa

Instituto Superior Técnico

DARIO: Distributed Agency for Reliable Input/Output

Project FCT POSC/EIA/56041/2004

CANELy Prototype Board Schematic Specification

DARIO Technical Report RT-05-04

R. Pinto, J. Rufino, C. Almeida

December 2005

Faculdade de Ciências da Universidade de Lisboa

Instituto Superior Técnico

CANELy Prototype Board Schematic Specification

To be submitted for publication: please do not distribute

Technical Report: DARIO RT-05-04
Authors: R. Pinto, J. Rufino, C. Almeida
Date: December 2005

This work was partially supported by the FCT through Projects POSC/EIA/56041/2004 (DARIO) and the Large-Scale Informatic Systems Laboratory (LASIGE).

CANELy Prototype Board Schematic Specification

Ricardo Pinto
IST-UTL*
rncp@rnl.ist.utl.pt

José Rufino
FCUL†
ruf@di.fc.ul.pt

Carlos Almeida
IST-UTL
cra@comp.ist.utl.pt

Abstract

Fault-tolerant distributed applications based on fieldbuses may take advantage from the availability of highly-dependable communication systems. In this paper, we address this problem in the context of CAN, the Controller Area Network, to conclude that CAN native mechanisms alone are unable to fulfill all the attributes of fault-tolerant communication protocols. The paper discusses how existing CAN controllers can be complemented with some simple machinery and low-level protocol modules, handling the problem effectively. The result is an enhanced CAN infrastructure able to extremely reliable communication.

This documents presents the schematic specification of a hardware infrastructure supporting the dependability enhancement mechanisms required for hard real-time communication in CAN-based systems.

1 Introduction

The design and implementation of distributed computer control systems intended for real-world interfacing, i.e. integrating sensors and/or actuators, have increasingly been based on standard fieldbuses as an alternative to specialized and thus costly architectures [4]. The development of applications for such environments may greatly benefit from the availability of services such as clock synchronization, reliable group communication, membership and failure detection.

However, the migration of fault-tolerant communication systems to the realm of fieldbuses presents non-negligible problems, some of them addressed by our ongoing research in the context of CAN, the Controller Area Network [15, 13, 10, 8]. CAN is a fieldbus that has assumed increasing importance and widespread acceptance in application areas as diverse as shop-floor control, robotics or automotive [2].

This paper outlines our approach on how to use CAN as an off-the-shelf component in the design of fault-tolerant real-time distributed systems.

2. CANELy: a CAN-based Fault-Tolerant Real-Time Distributed System

In the course of analyzing existing studies of CAN limitations with respect to the provision of strict availability, reliability and timeliness attributes [3], we have realized that what was missing in the native CAN fieldbus to attain levels of dependability comparable to those of similar technologies, such as the Time-Triggered Protocol [4], was indeed a set of fault tolerance and timeliness-related services. Moreover, we have shown that these can be provided off-the-shelf (i.e. without modifications to the CAN standard or to existing CAN controllers), through the use of properly encapsulated additional software/hardware components. We call the materialization of this concept **CAN Enhanced Layer** (CANELy).

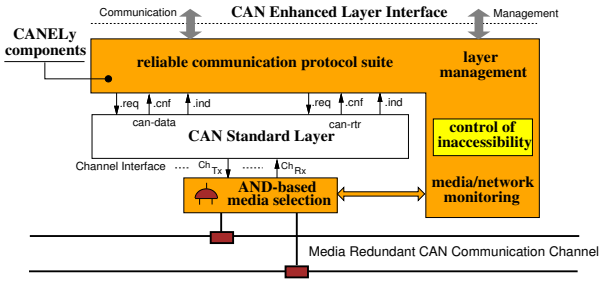

Figure 1. CAN Enhanced Layer architecture

The central component of the CANELy architecture (Figure 1) is naturally the standard CAN layer, complemented/enhanced with some simple machinery and low-level protocols, which include: a network infrastructure resilient to physical partitioning [11]; a reliable communication protocol suite, offering a set of broadcast/multicast primitives [13]; clock synchronization [8]; node failure detection and membership services [12].

The objective of this paper is to discuss the mechanisms and the techniques used in CANELy to enforce system correctness in the time-domain despite the occurrence of network errors, that is *control of inaccessibility*. We show that support for *inaccessibility flushing* can be provided effectively in CANELy through an extremely simple method.

*Instituto Superior Técnico - Universidade Técnica de Lisboa, Avenida Rovisco Pais, 1049-001 Lisboa, Portugal. Tel: +351-218418397 - Fax: +351-218417499. NavIST Group CAN WWW Page - http://pandora.ist.utl.pt/CAN.

†Faculdade de Ciências da Universidade de Lisboa, Campo Grande - Bloco C5, 1700 Lisboa, Portugal. Tel: +351-217500254 - Fax: +351-217500084. Navigators Home Page: http://www.navigators.di.fc.ul.pt. This work was partially supported by FCT through Project POSC/EIA/56041/2004 (DARIO).

CAN Standard Layer

The CAN fieldbus is a multi-master network that uses a twisted pair cable as transmission medium [2, 1]. The network maximum length depends on the data rate. Typical values are: 40m @ 1 Mbps; 1000m @ 50 kbps. Bus signaling takes one out of two values: *recessive* (r), otherwise the state of an idle bus; *dominant* (d), which always overwrites a recessive value. This behavior, together with the uniqueness of frame identifiers, is exploited for bus arbitration. A *carrier sense multi-access with deterministic collision resolution* policy is used. When several nodes compete for bus access, the node transmitting the frame with the lowest identifier always goes through and gets the bus. A *frame* is a network-level piece of encapsulated information. It may contain a *message*, a user-level piece of information. In CAN, a *data frame* is used for that purpose. However, it may consist of control information only, such as a *remote frame*.

The CAN standard layer is made from a CAN controller and the corresponding software driver that includes the following primitives (cf. Figure 1): *request* the transmission (.req) of data (can-data) or control (can-rtr) messages[1]; *confirm* to the user a successful message transmission (.cnf); *indicate* a message arrival (.ind).

3 CANELy Engineering

This section provides an overview of how the different CANELy dependability enhancement units can be implemented in a cost-effective way, using commercial off-the-shelf components, through the right integration of a comprehensive set of (simple) machinery resources and specific low-level protocols.

The overall architecture of CANELy is depicted in Figure 2 and comprises the CANELy processing infrastructure and the specialized support to low-level machinery functions.

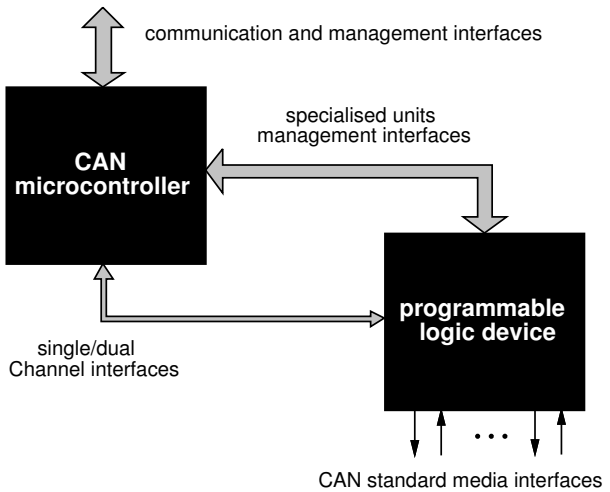

Figure 2. CANELy Hardware Support

[1]Control messages are encapsulated in remote frames.

Support to low-level machinery functions

This component is implemented by a single, medium capacity, programmable logic device (e.g. Xilinx FPGA[2]). It comprises the machinery required to:

- the implementation of bus media redundancy in CAN, including media quarantine schemes;
- the implementation of the mechanisms securing system correctness in the time-domain, despite the occurrence of network errors [9].
- support to optimized operation of the reliable group communication protocol suite;
- support to an innovative scheme to handle CAN site membership.

This device interfaces with the transmission media, through standard CAN media interfacing components, and with the CAN Processing Infrastructure, through the *Channel* and the network management interfaces (cf. Figure 2);

CANELy Processing Infrastructure

This highly integrated component comprises a single/dual CAN controller and the resources (e.g. microcontroller, memory) required for the execution of CANELy low-level protocols (group communication, node failure detection and site membership, clock synchronization). This component also includes the interfaces with the CANELy low-level machinery resources and with the high level components of the system (cf. Figure 2).

In our prototype implementation this component is materialized using the state of the art Dallas/Maxim DS80C390 High-Speed Microprocessor [6].

In addition to the dependability enchacement protocols the CANELy processing infrastructure is also required to support the implementation of different message transmission scheduling policies The automatic scheduling of a frame for retransmission is provided after a loss in the arbitration process or upon the occurrence of an error. Inside the node, there is no standardized method of selecting the message to be scheduled for transmission, at a given time. The attributes of a relevant set of commercial CAN controllers with that regard. One CAN controller implements a policy that selects for transmission the message with the lowest identifier. Most controllers transmit first the message stored in the buffer with the lowest number reference. One particular controller (MCP2510) uses a special-purpose priority field that software components can co-relate with either of the previous parameters, upon submission of a transmit request.

The management of message transmit buffers is of fundamental importance to prevent priority inversion, namely when the ordering of message transmissions by the CAN

[2]Field Programmable Gate Array

controller depends on the buffer numbering. If priority inversion occurs, the results of a traditional message schedulability analysis may be invalid. A comprehensive set of priority inversion scenarios, their degree of severity and impact in terms of message schedulability, is analyzed in [7].

In any case, the main conclusion to be drawn from our previous analysis of the CAN arbitration process is that the definition of message/frame identifiers is crucial to the provision of CAN timeliness guarantees. Despite its importance, we do not address this issue in detail, since it has been thoroughly studied by a number of authors:

- in [14], the message identifiers are assigned fixed-priorities defined accordingly with a deadline-monotonic (DM) scheduling, given a static set of messages. The guarantee that the message timeliness requirements are met is provided by an off-line feasibility test. This method was applied with success to the SAE automotive control systems benchmark;
- in [16], it is used a combination of dynamic and static scheduling. Hard real-time messages with tight deadlines are scheduled using a mixture of EDF[3] and DM methods while other hard real-time messages are scheduled using the DM method alone. Non real-time messages are ordered by a fixed-priority scheduling;
- an approach combining dynamic and static scheduling is also followed in [5]. A calendar-based resource reservation scheme allocates in advance all the time slots required to hard real-time communication. Soft real-time traffic is scheduled according to an EDF strategy and non real-time messages are assigned fixed priorities.

The use of dynamic scheduling requires the on-line update of message identifiers. This is possible because most of the currently available CAN controllers: provide a dual-port access to the message buffers; reload the internal outgoing buffer, each time an arbitration process is started.

Finally, note that the functions assigned to the message identifier field, are flexible enough to allow the implementation of a comprehensive set of message scheduling techniques, including those described in the works cited earlier.

4 CANELy Board Schematic

The schematic specification of a prototype board providing the support for the CANELy architecture is presented as an annex to this document.

[3]EDF, Earliest Deadline First. Actually, the method followed in [16] is a variant of EDF that uses the *slack time* (time to deadline) instead of the deadline itself.

References

[1] CiA - CAN in Automation. *CAN Physical Layer for Industrial Applications - CiA Draft Standard 102 Version 2.0*, Apr. 1994.

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

[3] H. Kopetz. A comparison of CAN and TTP. In *Proceedings of the 15th IFAC Workshop on Distributed Computer Control Systems*, Como, Italy, Sept. 1998. IFAC.

[4] H. Kopetz and G. Grunsteidl. TTP - a protocol for fault-tolerant real-time systems. *IEEE Computer*, 27(1):14–23, Jan. 1994.

[5] M. Livani, J. Kaiser, and W. Jia. Scheduling hard and soft real-time communication in the controller area network (CAN). In *Proceedings of the 23rd IFAC/IFIP Workshop on Real-Time Programming*, Shantou - China, June 1998. IFAC/IFIP.

[6] Maxim/Dallas Semiconductors. *DS80C390 Dual-CAN High-Speed Microprocessor*, Feb. 2005.

[7] A. Meschi, M. Natale, and M. Spuri. Priority inversition at the network adapter when scheduling messages with earliest deadline techniques. In *Proceedings of the 8th Euromicro Workshop on Real-Time Systems*, pages 243–248, L' Aquila, Italy, June 1996. IEEE.

[8] L. Rodrigues, M. Guimarães, and J. Rufino. Fault-tolerant clock syncronization in CAN. In *Proceedings of the 19th Real-Time Systems Symposium*, pages 420–429, Madrid, Spain, Dec. 1998. IEEE.

[9] J. Rufino. *Computational System for Real-Time Distributed Control*. PhD thesis, Technical University of Lisbon - Instituto Superior Técnico, Lisboa, Portugal, July 2002.

[10] J. Rufino, N. Pedrosa, J. Monteiro, P. Veríssimo, and G. Arroz. Hardware support for CAN fault-tolerant communication. In *Proceedings of the 5th International Conference on Electronics, Circuits and Systems*, pages 263–266, Lisbon, Portugal, Sept. 1998. IEEE.

[11] J. Rufino, P. Veríssimo, and G. Arroz. A Columbus' egg idea for CAN media redundancy. In *Digest of Papers, The 29th International Symposium on Fault-Tolerant Computing Systems*, Madison, Wisconsin - USA, June 1999. IEEE.

[12] J. Rufino, P. Veríssimo, and G. Arroz. Node failure detection and membership in CANELy. In *Proceedings of the 2003 International Conference on Dependable Systems and Networks*, pages 331–340, San Francisco, California, USA, June 2003. IEEE.

[13] J. Rufino, P. Veríssimo, G. Arroz, C. Almeida, and L. Rodrigues. Fault-tolerant broadcasts in CAN. In *Digest of Papers, The 28th International Symposium on Fault-Tolerant Computing Systems*, pages 150–159, Munich, Germany, June 1998. IEEE.

[14] K. Tindell and A. Burns. Guaranteeing message latencies on Controler Area Network. In *Proceedings of the 1st International CAN Conference*, pages 1.2–1.11, Mainz, Germany, Sept. 1994. CiA.

[15] P. Veríssimo, J. Rufino, and L. Ming. How hard is hard real-time communication on field-buses? In *Digest of Papers, The 27th International Symposium on Fault-Tolerant Computing Systems*, Washington - USA, June 1997. IEEE.

[16] K. Zuberi and K. Shin. Scheduling messages on Controller Area Network for real-time CIM applications. *IEEE Transactions on Robotics and Automation*, 13(2):310–314, Apr. 1997.


Net naming conventions:
- The first word before the underscore ("_") always refers to the device that originates the signal, the 2nd word designates the function of the signal ;
- In names with more than 2 words, the 3rd word will be just a reference label of the signal, e.g., CPU_INP_2 means that the signal is an input of the CPU, with reference "2";
- A "/" (slash) preceding the function word means that the signal is active low;
Power Supply Signaling
External JTAG connector
System Status / Configuration Words
Power Status Word
Power Configuration Word
System Configuration Word
Serial Comms ("Outside World")
Serial Terminal 1
Serial Terminal 2
Reset & Supervisory
Manual Reset Button
For other sources of reset signal
Bypass capacitors
Capacitors to be placed around FPGA (as close as possible) to supress noise induced in the power rails by the FPGA
Primary CPU
A Bus
B Bus
Reserved for possible hardware redundancy connection
Clock Signal Generation
Clock Signal
Program memory
1M x 8
Boot ROM
128K x 8
Battery backed-up RAM
32k x 8
Pull-ups
Channel A.0
Channel A.1
CONNECTOR DB9
The use of fault-tolerant transceivers is a constraint in terms of the possibility of bypassing the FPGA for bus access, since we need to generate ENABLE and other sort of signals to activate the chips.
Other solution would involve "regular" transceivers and optocouplers to "sense" the state of the bus and compare it with the outputs connected to the transceiver
Secondary CPU
Glue Logic / Lock control
Host System Interface
Host Interface
General interface, to allow interfacing with multiple types of systems
Visual Status Info
7-segment display
Heartbeat LEDs
Missing: Memory-mapped Devices Addresses
Memory map
Data
Program
Program memory
Boot ROM
Interrupt Vector Table
Title
DARIO Main Board
Document Number
<Doc>
Rev
B
Date: Monday, January 23, 2006
Sheet 1 of 1