EDF Consensus on CAN Bus Access for Dynamic Real-Time Applications

Mohammad Ali Livani and Jörg Kaiser
University of Ulm
Department of Computer Structures

Abstract

The paper introduces a mechanism to implement distributed scheduling for CAN-bus resource in order to meet the requirements of a dynamic distributed real-time system. The key issues considered here, are multicasting, distinguishing between hard real-time, soft real-time, and non real-time constraints, achieving high resource utilization for CAN-bus, and supporting dynamic hard real-time computing by allowing dynamic reservation of communication resources.

1 Introduction

A real-time communication system (RTCS) constitutes the backbone for distributed control applications. RTCS substantially differ in many respects from general purpose communication systems. In general, while the goals of general purpose communication systems center around throughput, RTCS focus on predictability of communication. Predictability means that the system exhibits an anticipated behaviour in the functional and the temporal domain.

In the area of industrial automation and the automotive industry, field busses are used to disseminate time critical messages. Field busses exhibit bounded message latency, reliability of transfer, efficiency of the protocol and, to a certain extent, preventing a node from monopolizing the network. Among field busses, CAN bus[13] provides advanced built-in features, which make it suitable for complex real-time applications[8]. Some of these features are priority-based, multiparty bus access control using carrier sense / multiple access with collision avoidance (CSMA/CA), bounded message length, efficient implementation of positive/negative acknowledgement, and automatic fail-silence enforcement with different fault levels.

As a common resource, the CAN bus has to be shared by all computing nodes. Access to the bus has to be scheduled in a way that distributed computations meet their deadlines in spite of competition for the communication line. Since the scheduling of the bus cannot be based on local decisions, a distributed consensus about the bus access has to be achieved.

There exist several alternative approaches to solve this problem based on the assumptions about the behaviour of the system and the environment. The TDMA (Time-Division Multiple Access) protocol does not take advantage of the priority-based bus arbitration mechanism. Relying on the availability of a high-resolution global time-base, each node knows exactly when to send or receive a message.

However, due to its static nature, the TDMA approach is not suitable for dynamic real-time systems. Moreover, the TDMA protocol is applied preferrably on high-bandwidth busses, like Ethernet or FDDI [7][11].

The deadline-monotonic priority assignment [16] achieves meeting deadlines as guaranteed by an off-line feasibility test for a static system with periodic tasks. Although static systems can be scheduled easily by this approach, it does not allow scheduling of dynamic systems, where an offline feasibility test has incomplete knowledge about the future behavior of the system.

A dynamic scheduling of CAN bus has been approached in [18]. However, this approach makes unrealistic assumptions about CAN – e.g. 10 Mbits/s – and exhibits a rather restricted scheduling ability due to a short time horizon (cf. section 4).

In this paper, we introduce a mechanism to assign dynamic priorities to CAN messages, in order to achieve an EDF resource access consensus among the participating nodes. Our motivation is to schedule soft real-time communication optimally with EDF approach, and to guarantee deadlines of hard real-time communication by calendar-based resource reservation. As we have shown in [7], these two scheduling approachs can co-exist in CAN, if they are based on our EDF access mechanism.

We take advantage of the built-in CSMA/CA access protocol of CAN bus to implement the EDF access regulation. The CSMA/CA protocol is comparable with a priority-based dispatcher. Due to this analogy, we express our scheduling decisions for the CAN-bus resource by static or dynamic priority orders.

The paper is organized as follows: Section 2 introduces some features of CAN. In section 3 some of the common approaches to the CAN-bus scheduling are discussed. In Section 4 we describe our dynamic priority assignment mechanism, resulting in EDF access consensus. Based on the EDF access consensus, we introduce in section 5 a real-time scheduling approach for communication on CAN bus. A summary concludes the paper.

2 Basic CAN-Bus Properties

The key to the understanding of the CAN-Bus is its bit synchronization and the fact that all nodes (including the sender) scan the value of a bit while it is transmitted. Hence, all correct nodes have a consistent view of every bit during its transmission.

The nodes of the CAN-Bus are connected logically via a wired-AND or, alternatively, a wired-OR function. This means that we can distinguish dominant and recessive levels on the bus. In a wired-AND connection - as implemented by Intel, Motorola, Siemens, etc. - a ‘1’ is a recessive bit-value and a ’0’ is a dominant bit-value because if different bit-values are put on the bus by different nodes simultaneously, the logical AND function of the values is seen by all nodes.

The consistent view of every bit among all CAN controllers is exploited for a priority-based arbitration mechanism. Whenever the bus is idle, any node may start to transmit a message by transmitting the identifier bits whilst simultaneously monitoring the bus. If a node transmits a recessive bit of the identifier but monitors a dominant bit, it becomes aware of a collision. In this case, the node concludes that it is not transmitting the highest priority message, stops transmission and switches to the receive mode. Eventually, the identifier which does not lead to the detection of a

collision by the sending node wins the arbitration and the attached data is fully transmitted. It will be received by all participants on the net. In CAN terminology, this technique is called „collision avoidance“ (CSMA/CA). It guarantees that the highest priority message of competing transmitters is broadcasted without delay. In the wired-AND implementation of CAN, a lower binary value of the message identifier means a higher priority. This priority-based collision handling requires also, that all messages which compete for the bus, have different identifiers.

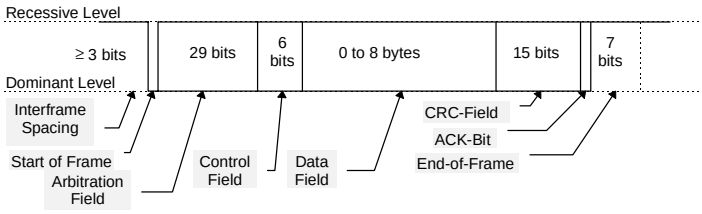


Fig. 1. CAN message format (Data Frame)

The general format of a message, the data frame, is sketched in Fig. 1. A message is tagged by an identifier, which defines also its priority. The identifier consists of 29 bits in extended message format, or 11 bits, according to the older „standard format“. Each individual CAN message can transport up to 8 bytes of data. The length of the data field is indicated in the control field.

CAN provides a connectionless protocol. This means, that the message identifier indicates the content of a message rather than the source or destination. Each communication controller has a programmable filter through which it can associatively detect relevant messages. By setting some of the filter bits as „don’t care“, it is possible to receive groups of messages. However, the combination of this feature with dynamic priority assignment together with the guarantee of identifier uniqueness, requires the usage of a 29-bit identifier which expands the message header. However, since the group name and end-to-end delivery deadline are included in the identifier field, the data field remains free for application use. Given the 29-bit identifier, the fault-free transmission of a CAN message takes between 66 and 154 bit-times, depending on the data field length and bit-stuffing.

3 Existing Scheduling Approachs for CAN

In this section, we provide an overview of a few different scheduling approachs. In the CANopen standard [3], a master node transmits a high-priority SYNC message periodically. The senders of all synchronous messages then transmit their messages with fixed predefined priorities. The CANopen approach, however, has two disadvantages. Firstly, the SYNC message is issued by a master node, and in the case the master node fails to send it, the synchronous message exchange is stopped until a bus master is determined by an assessment protocol. The second problem of CANopen is that asynchronous events have low priorities, and thus they are delayed by the whole transmission time of the SYNC message plus all synchronous application messages. If one stores an asynchronous event and polls it by a synchronous message,

then the length of the SYNC period may not exceed the deadline of the event. Due to these problems, neither safety-critical hard real-time applications nor asynchronous real-time applications can rely on CANopen protocol.

Another fixed priority bus scheduling approach has been made by Tindell and Burns [16], who have applied the Deadline Monotonic[1] scheduling method on the bus resource. The authors have shown that CAN fulfills the requirements of static hard real-time systems, by applying their analysis of the fixed priority bus scheduling policy to the SAE benchmark class C automotive systems (safety critical control applications). However, due to its static nature, the Deadline Monotonic approach is not suitable for dynamic and adaptive real-time applications.

Zuberi and Shin [18] have made an approach to schedule CAN bus by a mixture of static and dynamic priorities. However, this approach makes unrealistic assumptions about CAN – e.g. 10 Mbits/s bandwidth and 30 µs end-to-end transmission delay – and exhibits a rather restricted scheduling ability due to a short time horizon. Moreover, at certain points of time (near the end of each „epoch“) the correct arbitration among more than two competing nodes cannot be guaranteed, because of a too short *time horizon*. The meaning of time horizon will be explained later in this paper.

None of the above solutions considers either the need of distributed real-time system for a group communication protocol, or the facilities provided by CAN controllers to efficiently realize multicasting.

4 EDF Access Consensus on CAN

As we have shown in [7], an EDF bus access consensus among the CAN nodes can serve as a basis for a hybrid bus scheduling, where timeliness of hard real-time messages is guaranteed by resource reservation, while optimal scheduling of soft real-time communication is achieved by EDF scheduling strategy.

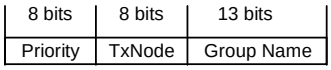
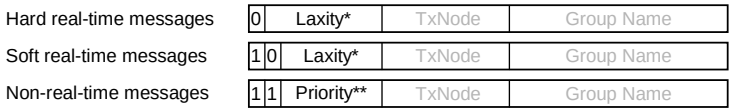


Fig. 2. A partitioned CAN arbitration field

In order to realize the EDF access regulation on CAN bus, we must define a mapping of the transmission deadline into the message priority, such that a message with an earlier transmission deadline wins the bus arbitration against a message with a later deadline. However, we don't use the whole identifier as a deadline-driven dynamic priority because of following reasons: Firstly, we must include the sender node identifier into the identifier field of CAN messages, to ensure that competing messages have always different identifiers. Secondly, we want to encode the message group name into the identifier in order to support multicast message filtering by the CAN controller hardware. Fig. 2 shows how the priority, sender ID, and message group name are composed to a CAN identifier.

Due to our requirement to schedule three types of communication, namely hard real-time, soft real-time, and non real-time, we divide the value domain of the 8-bit

priority field into three distinct ranges. The highest priorities are reserved for hard real-time messages, the middle range is assigned to soft real-time messages, and the lowest priorities are assigned to non real-time messages. This ensures that more critical messages always win the bus against less critical messages. As illustrated in Fig. 3, this range partitioning is realized by assigning the prefix '0' to hard real-time priorities, '10' to soft real-time priorities, and '11' to non real-time priorities.



* this is the time remaining until the latest transmission time or the end of the reserved time slot

** due to the wired-AND implementation of CAN bus, 0 means a higher priority than 1

Fig. 3. Encoding message priorities into the identifier field

In the priority field of a real-time message, we encode the time remaining until its transmission deadline (let's call it *delivery laxity*). The *transmission deadline* is a point of time specified by the sending application object, when a message must be completely transmitted to receiving nodes. This is tightly related to the *delivery deadline*, which specifies the latest time of the end-to-end message delivery. As long as a sending node is pending for the bus, its communication subsystem checks and updates the delivery laxity of the ready message periodically. As soon as the delivery laxity becomes zero, the message is withdrawn, and the sending application object is notified of the deadline failure.

Each value of the delivery laxity is mapped to a portion of future time, a time slot. At the end of each time slot, a pending transmitter increases the priority of its real-time message by decrementing its delivery laxity field. The time slots have a fixed length, and the first one begins at the present time. Only the last time slot is open-end, in order to achieve a complete mapping of the future time. Since we can make no order decision between any two points of time falling into the last time slot, the beginning of the last time slot will be the *time horizon* of our deadline-driven access decision.

Def. *The Time Horizon: By time horizon we mean the point of time, where a time-driven decision mechanism (like EDF scheduler) cannot see beyond. In our case, if the time until the time horizon is only long enough for the transmission of n messages, then the EDF bus scheduler cannot schedule n+1 real-time messages, which are pending simultanously for transmission.*

Having seven bits left for the delivery laxity of hard real-time messages, the time horizon comes after 127 time slots. In order to ensure dynamic priority changing at each bus arbitration round, transmitters must change the message priorities even after the shortest possible message transmission. Hence, a time slot should not be longer than 66 bit-times, which is the transmission time of a message with empty data field and no bit-stuffing in extended format. This results in a maximum time horizon of 8382 bit-times, which allows for scheduling of up to 54 nodes simultanously requesting the bus for hard real-time communication, each requiring the longest

message transmission time of 154 bit-times. In case of soft real-time communication, there are only six bits remaining for the delivery laxity. This results in a time horizon of 4158 bit-times, which enables scheduling of 27 transmitters competing with soft real-time messages.

5 Hybrid Bus Scheduling Based on EDF Consensus

We assume four classes of activities in our system, which have been introduced in [15]. Critical activities are hard real-time and their deadlines are absolute in time. To guarantee the timely execution of critical activities, all their occurrences have to be predicted and respective resources have to be scheduled in advance in a periodic calendar. Essential activities have deadlines relative to their start times. If the system grants an essential task, it guarantees the resources for a timely execution by reserving the appropriate time slots in the resource scheduling calendar. However, the system can refuse a guarantee.

Soft real time activities have deadlines which are considered by the system but no guarantees are given to meet such a deadline. Soft real-time activities are executed on a best effort basis, however, as shown later, an EDF-like scheduling is used for optimal resource utilization. Non real-time activities can only use resources which are not requested by a real-time activity.

The global scheduling in a distributed system requires consensus between all participants about the usage of shared system resources. Particularly, if a joint action will be performed, all local resources must be available and reserved for the respective computation[5]. The global plan has to be enforced by all participants, based on their local information. In a completely static system, a global calendar is available and each participant has its relevant entries referring to its activities in a global time scale. A local activity may only be started according to this schedule[11]. In a more dynamic system where critical, essential, soft real-time and non-real-time tasks coexist, things are more complicated. If a computing resource is free, a less critical task may start computation and request resources. In this case, it must be guaranteed that it does not block an activity with a higher criticality. In this section, the enforcement of the global schedule for the shared bus resource is described.

For hard real-time communication, we guarantee the deadline by reserving a time slot on the bus. The reserved time slots are entered into a calendar, which contains both periodic fixed reservations and dynamic reservations. This calendar is contained in each node of the system, thus the time slot reservation is performed by a single atomic multicast message only.

Our scheduling approach for hard real-time communication requires access to a global time reference. Once a time slot is reserved, the respective action can be started locally. To guarantee that it does not interfere with another time slot, the time reference of all nodes must be synchronized. The lower the clock accuracy, the larger the minimum gap between two subsequent time slots in the global bus schedule. Fig. 4 shows a situation, where a too low clock accuracy causes a collision of different time slots. In order to provide a global time reference with high accuracy, clock synchronization mechanisms have to be applied, e.g. as described in [4] or [10].

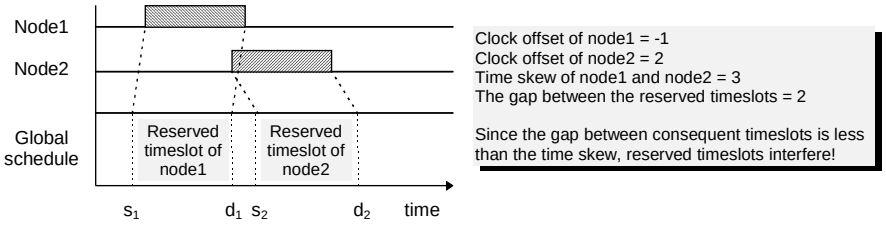


Fig. 4. Collision of reserved time slots due to low clock accuracy

Efficient reservation of time slots for hard real-time communication is supported by defining the end of the reserved time slot of a message as its transmission deadline. The sender of a hard real-time message enforces its access right by dynamically increasing the priority of the message according to its laxity relative to the reserved time slot. Due to this scheme, a hard real-time message gains the highest possible priority at the beginning of its reserved time slot. Fig. 5 illustrates a situation, where several transmitters compete for the bus access, near the reserved time slot of a hard real-time message k. Messages i and k are ready to be transmitted after t_0, where message j is already started. Because CAN message transfer is non-preemptive, message j is completed regardless of its priority. As the bus becomes idle at t_1, messages i and k compete for the bus according to their priorities. We assume that the next reserved time slot after t_1 begins at s_k, and belongs to the message k. Then message k is guaranteed to have the highest priority at t_1, and to win the arbitration process.

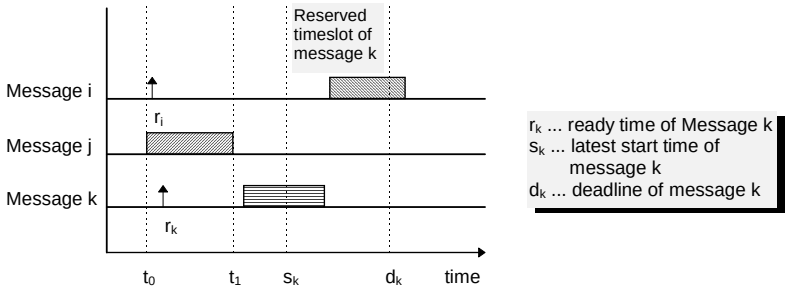


Fig. 5. The competition for the bus near a reserved time slot

Note that k may be delayed by one message, which is started before the ready time of k. We define ΔT_{max} as the longest possible message transmission time. Then, a hard real-time message k – with a reserved time slot beginning at s_k – must be ready before $s_k - \Delta T_{max}$, in order to tolerate the non-preemptive transmission of the longest possible message.

In order to guarantee the collective timeliness of hard real-time communication, the following requirements must be met:

(R1) for each hard real-time message, an exclusive time-slot is reserved, which ends before the transmission deadline of the message,

(R2) the reserved time-slot of each message is as long as the worst-case transmission time of the message, including all overheads,

(R3) the reserved time slots of hard real-time messages do not overlap,

(R4) every hard real-time message is ready for transmission at least ΔT_{max} before the beginning of its reserved time slot. This means that the laxity of every hard real-time message at its ready-time is large enough to allow the longest possible message of the system to be transmitted first,

(R5) the priority of a hard real-time message depends on its laxity (Fig. 3).

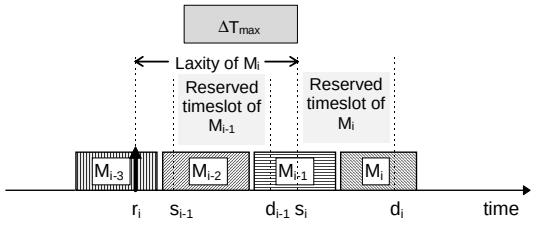


Fig. 6. A hard real-time message missing its deadline

Given these assumptions, every hard real-time message will be transmitted timely under fault-free conditions. Assume (Fig. 6) that M_i is the first hard real-time message after the system startup, which cannot start at the beginning of its reserved time slot s_i, hence missing its transmission deadline d_i. Since M_i is ready at $s_i - \Delta T_{max}$, it participates in at least one arbitration process before s_i. If M_i does not start before s_i, then another message M_{i-1} wins the arbitration against M_i, and following conditions are true:

(C1) M_{i-1} is a hard real-time message,

(C2) the reserved time slot of M_{i-1} ends at d_{i-1}, where $d_{i-1} < d_i$, because of R5, and

(C3) M_{i-1} is not completely transmitted until s_i.

From R3 and C2 we conclude that the transmission deadline of M_{i-1} lies before s_i, i.e. $d_{i-1} < s_i$. Due to C3, M_{i-1} misses its deadline d_{i-1}, because it is completed after s_i, and $d_{i-1} < s_i$. This is a contradiction to the assumption that M_i is the first hard real-time message after system startup, which misses its deadline.

In order to guarantee timely hard real-time message transfer in the presence of faults, redundancy must be provided. Space redundancy would require a second CAN bus. If we apply time redundancy, we have to schedule two subsequent transmissions plus the failure handling mechanism of CAN bus, which consumes bounded time [14]. Our application of time redundancy is similar to strategies used in other real-time communication protocols, e.g. TTP. However, in contrast to statically planned

communication, we can use the redundant time slot for low-priority communication, if the first transmission was successful.

For the soft real-time communication, our mechanism does not guarantee a deadline. This is because it is always possible to dynamically schedule additional hard real-time („essential“) communication. However, we guarantee optimal scheduling of soft real-time messages, because firstly, their priorities are higher than that of non real-time messages, and secondly, the priority of a soft real-time message depends directly on the time remaining until its delivery deadline, thus realizing EDF scheduling, which is known to be optimal[12].

Non real-time messages are assigned fixed priorities, because the importance of a non real-time message does not change by the passage of time. Note that the values ‘11111110’ and ‘11111111’ are not allowed due to the CAN specification[13].

6 Conclusion and Future Research

In this paper, we introduced a dynamic priority access regulation for CAN bus, which enables the scheduling of the bus resource according to EDF, or other time-driven approachs. We have exploited the CSMA/CA protocol provided by CAN, to achieve an access consensus among all nodes, based on the dynamic priorities.

In order to guarantee timely delivery of hard real-time messages, we have introduced a resource reservation mechanism, which coexists with the EDF scheduling used for soft real-time messages. We achieve timeliness in presence of faults, by applying time redundancy. However, in case of non-faulty message transfer, the unused redundant resources are used for low-priority communication.

Currently, we are working on a fault injection component, which systematically injects faults in certain bits of messages. This will provide a means to study the effectiveness of our multicast protocol under different fault conditions. We are also developing a monitoring component which observes the behavior of the CAN controllers upon detecting a fault.

Our concepts will be evaluated in a testbed, a distributed robotics application consisting of cooperative hardware objects, like intelligent sensors and actuators, all connected by a CAN bus.

References

[1] N. C. Audsley, A. Burns, M. F. Richardson, A. J. Wellings: „Hard Real-Time Scheduling: The Deadline Monotonic Approach“, *Proceedings of 8th IEEE Workshop on Real-Time Operating Systems and Software, May 1991.*

[2] S-C. Cheng, J. A. Stankovic, K. Ramamritham: „Scheduling Algorithms for Hard Real-Time Systems – A Brief Survey“, *IEEE Tutorial on Hard Real-Time Systems, J. A. Stankovic and K. Ramamritham, ed., IEEE Computer Society Press 1988.*

[3] CiA Draft Standard 301 version 3.0, „CANopen Communication Profile for Industrial Systems“

[4] M. Gergeleit and H. Streich: „Implementing a Distributed High-Resolution Real-Time Clock using the CAN-Bus“, *Proceedings of the 1st International CAN-Conference, Mainz, Germany, Sep. 1994.*

[5] M. Gergeleit, J. Kaiser, H. Streich: „DIRECT: Towards a Distributed Object-Oriented Real-Time Control System“, *Workshop on Concurrent Object-based Systems, Oct. 1994.*

[6] E. D. Jensen, „A Real-Time Manifesto“, *http://www.real-time-os.com*

[7] J. Kaiser, M. A. Livani: „Invocation of Real-Time Objects in a CAN Bus-System“, *accepted for the 1st Int'l Symposium on Object-Oriented Distributed Real-Time Computing Systems, Kyoto, Apr. 1998.*

[8] J. Kaiser, M. A. Livani, W. Jia: „Membership and Order in a CAN-Bus Based Real-Time Communication System”, *Proc. of Int'l Workshop Advance Parallel Processing Technology, Sep. 1997.*

[9] H. Kopetz and G. Grünsteidl, „TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems“, *Research Report No. 12/92, Institut für Technische Informatik, Technical University of Vienna, 1992.*

[10] H. Kopetz and W. Ochsenreiter: „Clock Synchronization in Distributed Real-Time Systrems“, *IEEE Trans. on Computers, C-36(8):933-940, Aug. 1987.*

[11] H. Kopetz and W. Merker: „The Architecture of MARS“, *IEEE Proceedings of the 15th Fault Tolerant Computing Systems Symposium, 1985.*

[12] C. L. Liu and J. W. Layland, „Scheduling Algorithms for Multiprogramming in a Hard real-Time Environment“, *Journal of the ACM Vol. 20, No. 1, pp. 46-61, 1973.*

[13] ROBERT BOSCH GmbH, „CAN Specification Version 2.0“, *Sep. 1991.*

[14] J. Rufino and P. Veríssimo, „A Study on the Inaccessibility Characteristics of the Controller Area Network“, *2nd International CAN Conference 95, London, UK, Oct. 1995.*

[15] J.A. Stankovic and K. Ramamritham: „The Spring Kernel: A New Paradigm for Real-Time Operating Systems“, *ACM Operating Systems Review 23(3), July 1989*.

[16] K. Tindell and A. Burns: „Guaranteeing Message Latencies on Control Area Network (CAN)“, *Proceedings of the 1st International CAN Conference, 1994.*

[17] K. Tindell, A. Burns, A. Wellings: „Calculating Controller Area Network (CAN) Message Response Times“, *Control Engineering Practice, Vol. 3, No. 8, pp. 1163-1169, 1995.*

[18] K. M. Zuberi and K. G. Shin, „Non-Preemptive Scheduling of messages on Controller Area Network for Real-Time Control Applications“, *Proc. Real-Time Technology and Applications Symposium, pp. 240-249, May 1995.*