International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-1, Issue-5, November 2012

Design and Development of Can Sniffer

N. Prabhakar Reddy, K.Sasidha

Abstract- Controller Area Network (CAN) is used extensively in automotive applications, with in excess of 400 million CAN enabled microcontrollers manufactured each year. CAN messages could be calculated and hence guarantees provided that message response times would not exceed their deadlines. This seminal research has been cited in over 200 subsequent papers and transferred to industry in the form of commercial CAN schedulability analysis tools. These tools have been used by a large number of major automotive manufacturers in the design of in-vehicle networks for a wide range of cars, millions of which have been manufactured over the last 8 years. This paper shows that the original schedulability analysis given for CAN messages is flawed. It may provide guarantees for messages that will in fact miss their deadlines in the worst-case. This paper provides revised analysis resolving the problems with the original approach. Further, it highlights that the priority assignment policy, previously claimed to be optimal for CAN, is not in fact optimal and cites a method of obtaining an optimal priority ordering that is applicable to CAN. The paper discusses the possible impact on commercial CAN systems designed and developed using flawed schedulability analysis and makes recommendations for the revision of CAN schedulability analysis tools. The CAN Sniffer Tool is a simple to use low cost CAN bus monitor which can be used to develop and debug a high speed CAN network. The tool supports CAN 2.0b and ISO11898-2 and a broad range of functions which allow it to be used across various market segments including automotive, industrial, medical and marine. The toolkit comes with all the hardware and software required to connect a CAN network to a PC. In CAN bus, the two CAN channels can send/receive CAN messages either with extended or standard ID. All messages received by the CAN interface are sent via UART to the serial port of PC. On the PC the CAN-messages get collected and ordered by CAN-ID. In CAN the communication is done in two-wire, the CAN sniffer can receives the messages based on arbitration process.

Keywords: CAN, UART, CAN-ID, PC.

I. INTRODUCTION

Controller Area Network (CAN) is a serial communications bus designed to provide simple, efficient and robust communications for in-vehicle networks. CAN was developed by Robert Bosch GmbH beginning in 1983 and presented to a wider audience at the Society of Automotive Engineers (SAE) Congress in 1986 – effectively the “birth of CAN”. In 1987 the first CAN controller chips were released by Intel (82526) and Philips (82C200). In the early 1990s Bosch submitted the CAN specification [20] for standardisation, leading to publication of the first ISO standard for CAN (11898) in 1993. Mercedes was the first automotive manufacturer to deploy CAN in a production car, the 1991 S-class. By the mid 1990s, the complexity of automotive electronics was increasing rapidly. The number of networked Electronic Control Units (ECUs) in Mercedes, BMW, Audi and VW cars went from 5 or less at the beginning of the 1990s to around 40 at the turn of the millennium. With this explosion in complexity traditional point-to-point wiring became increasingly expensive to manufacture, install, and maintain due to the hundreds of separate connections and tens of kilograms of copper wire required. As a result CAN was rapidly adopted by the cost-conscious automotive industry, providing an effective solution to the problems posed by increasing vehicle electronics content. Following on from Mercedes other manufacturers including Volvo, Saab, BMW, Volkswagen, Ford, Renault, PSA, Fiat and others all adopted CAN technology.

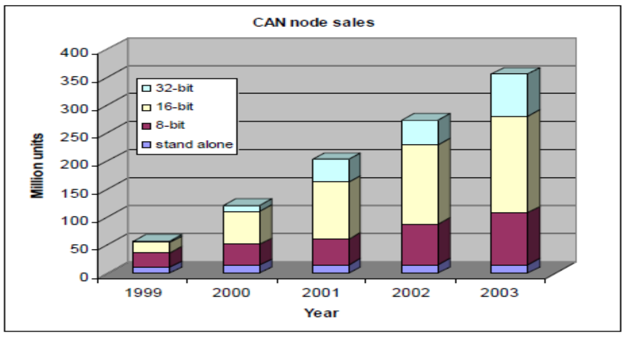
As a result of the wholesale adoption of CAN by the automotive industry, sales of CAN nodes (8, 16 and 32-bit microcontrollers with on-chip CAN peripherals) grew from just under 50 million in 1999 to over 340 million in 2003

By 2004 there were at least 15 silicon vendors manufacturing, in total, over 50 different microprocessor families with on-chip CAN capability.

Today almost every new car manufactured in Europe is equipped with at least one CAN bus. In the United States, the Environmental Protection Agency has mandated the use of CAN, for On Board Diagnostics, in all cars and light trucks sold in the US from model year 2008 onwards.

Message	Priority	Period	Deadline	TX time
A	1	2.5ms	2.5ms	1ms
B	2	3.5ms	3.25ms	1ms
C	3	3.5ms	3.25ms	1ms

CAN Messages Highlighting Flawed Analysis



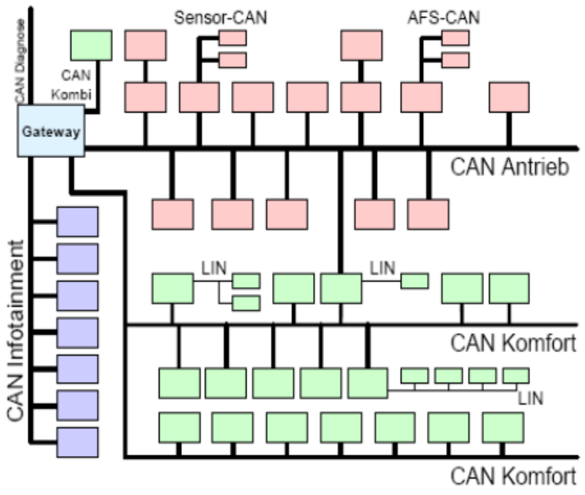
Sales of Microcontrollers with on-chip CAN Peripherals

Automotive Applications

In automotive applications, CAN is typically used to provide high speed networks (500Kbits/s) connecting chassis and power-train ECUs, for example engine management and transmission control. It is also used for low speed networks (100 or 125Kbits/s) connecting body and comfort electronics, for example door modules, seat modules and climate control. Data required by ECUs on different networks is typically gatewayed between the different CAN buses by a powerful ECU connected to both.

Revised Manuscript Received on 30 November 2012.

* Correspondence Author
N. Prabhakar*, Reddy, MTECH:(DS&CE)
K.Sasidha, Associate Professor Head-Incharge,ECM Dept.



VW Passat Network Architecture

The network architecture of the VW Passat [33] shown in Figure 2, reproduced from [15], illustrates how a number of CAN buses are used to connect around 45 ECUs in that vehicle. Also shown in Figure 2 are three Local Interconnect Networks (LIN). LIN is a complementary technology to CAN, and is used to provide inexpensive, low speed (20Kbits/s) connectivity.

II. CONTROLLER AREA NETWORK (CAN)

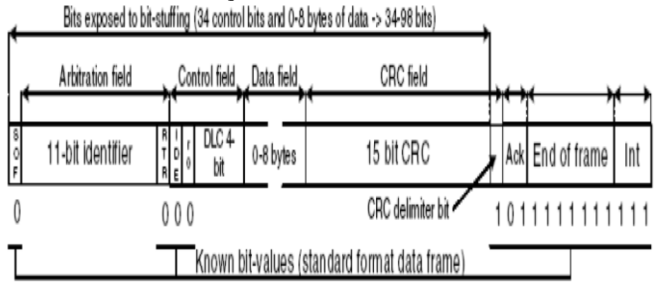
This section describes elements of the CAN protocol and characteristics of a system model that are needed to formulate a schedulability test.

CAN Protocol and Terminology

Controller Area Network (CAN) is a multi-master serial data bus which uses Carrier Sense Multiple Access/ Collision Resolution (CSMA/CR) to determine access.

CAN was designed as a simple and robust broadcast bus capable of operating at speeds of up to 1 Mbit/s. Message transfer over CAN is controlled by 4 different types of *frame*: Data frames, Remote Transmit Request (RTR) frames, Overload frames and Error frames.

The layout of a standard format data frame is shown in Figure 4. Each CAN data frame is required to have a unique identifier. Identifiers may be 11-bit (standard format) or 29-bit (extended format). The identifier serves two purposes beyond simply identifying the message. First, the identifier is used as a priority to determine which message among those contending for the bus will be transmitted next. Second, the identifier may be used by receivers to filter out messages that they are not interested in, and so reduce the load on the receiver's host microprocessor.



Standard Data Frame Format

Priority Based Arbitration The CAN physical layer supports two states termed *dominant* ('0') and *recessive* ('1'). If two or more CAN controllers are transmitting at the same time and at least one of them transmits a '0' then the value on the bus will be a '0'. This mechanism is used to control access to the bus and also to signal errors.

The CAN protocol calls for nodes to wait until a *bus idle period*[4] is detected before attempting to transmit. If two or more nodes start to transmit at the same time, then by monitoring each bit on the bus, each node can determine if it is transmitting the highest priority message (with a numerically lower identifier) and should continue or if it should stop transmitting and wait for the next bus idle period before trying again. As the message identifiers are unique, a node transmitting the last bit of the identifier field, without detecting a '0' bit that it did not transmit, must be transmitting the message with the lowest numerical value and hence the highest priority that was ready at the start of arbitration. This node then continues to transmit the remainder of its message, all other nodes having backed off.

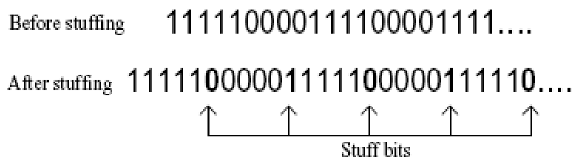
The requirement for a node to be able to overwrite a recessive bit, and the transmitting node detect this change, limits the combination of physical length and speed of CAN bus. The duration of each bit must be sufficient for the signal to propagate the length of the network. This limits the maximum data rate to 1Mbit/s for a network up to 40m in length or to 125Kbit/s for a 500m long network. The arbitration mechanism employed by CAN means that messages are sent as if all the nodes on the network shared a single global priority based queue. In effect messages are sent on the bus according to fixed priority non-pre-emptive scheduling. The above high level description is a somewhat simplified view of the timing behaviour of CAN. CAN does not have a global concept of time, rather each CAN controller typically has its own clock which, within a tolerance specified by the protocol, may drift with respect to the clocks of other nodes. The CAN protocol therefore requires that nodes re-synchronise on each message transmission. Specifically, every node must synchronise to the leading edge of the *start of frame* bit caused by whichever node starts to transmit first. Normally, CAN nodes are only allowed to start transmitting when the bus is idle. Thus, when the bus is idle beyond the 3-bit *inter-frame space* and a node starts to transmit a message beginning with the dominant start of frame bit ("0"), then all the other nodes synchronise on the leading edge of this bit and become *receivers* – i.e. they are not permitted to transmit until the bus next becomes idle. In this case any message that becomes ready for transmission after the leading edge of the start of frame bit has to wait for the next bus idle period before it can enter into arbitration.

However, to avoid problems due to clock drift, the CAN protocol also specifies that, if a CAN node has a message ready for transmission and detects a dominant bit at the 3rd bit of the inter-frame space, it will interpret this as a start of frame bit, and, with the next bit, start transmitting its own message with the first bit of the identifier without first transmitting a start of frame bit and without becoming a receiver. Again the leading edge of the start of frame bit causes a synchronisation. This behaviour ensures that any messages that become ready for transmission, whilst another message is being sent on the bus, are entered into the next round of arbitration, irrespective of any, within tolerance, clock drift.

1) *Error Detection*

CAN was designed as a robust and reliable form of communication for short messages. Each data frame carries between 0 and 8 bytes of payload data and has a 15-bit Cyclic Redundancy Check (CRC). The CRC is used by receiving nodes to check for errors in the transmitted message. If a node detects an error in the transmitted message, which may be a bit-stuffing error a CRC error, a form error in the fixed part of the message or an acknowledgement error, then it transmits an error flag. The error flag consists of 6 bits of the same polarity: '000000' if the node is in the error active state and '111111' if it is error passive. Transmission of an error flag typically causes other nodes to also detect an error, leading to transmission of further error flags.

Bit Stuffing As the bit patterns '000000' and '111111' are used to signal errors, it is essential that these bit patterns are avoided in the variable part of a transmitted message – see Figure 4. The CAN protocol therefore requires that a bit of the opposite polarity is inserted by the transmitter whenever 5 bits of the same polarity are transmitted. This process is referred to as *bit-stuffing*, and is reversed by the receiver.



Practical Implications of the Model Engineers wanting to use the analysis given in section 3 to analyse CAN based systems must be careful to ensure that all of the assumptions of the above model hold for their system.

In particular, it is important that each CAN controller and device driver is capable of ensuring that, at any given time when arbitration starts, the highest priority message queued at that node is entered into arbitration. This behavior is essential if message transmission is to take place as if there were a single global priority queue and for the analysis given in section 3 to be applicable the Philips 82C500 CAN controller cannot in general support this behaviour. Also the Intel 82527 CAN controller has a feature where messages are entered into arbitration in slot order rather than identifier order. In this case it is important that messages are allocated to slots in identifier order to preserve the correct priority based behaviour. Many on-chip CAN controllers have multiple slots that can be allocated to either transmit or receive a specific message. For example some Motorola, National Semiconductor, Fujitsu and Hitachi on-chip CAN peripherals have 14, 15 or 16 such slots. These slots typically have only a single buffer and therefore it is necessary to ensure that the previous instance of a message has been transmitted before any new data is written into the buffer, otherwise the previous message will be overwritten and lost. This behaviour provides an additional constraint on message transmission: the deadline of each message must be less than or equal to its period.

III. DISCUSSION

In this section we consider various characteristics of CAN systems and discuss whether flaws in the existing analysis can result in erroneous guarantees under specific circumstances that are relevant to real-world systems.
We seek to answer the following questions.

1. Can the existing analysis give faulty guarantees to messages of any priority?
2. If the bus utilization is low, can the existing analysis still result in optimistic response times?
3. Do error models give sufficient engineering margin for error to account for the flaw in the analysis?
4. Does the omission of diagnostic messages during normal operation reduce interference / blocking enough to ensure that the deadlines of the remaining messages will be met?
5. Which message guarantees can we be sure are not at risk?

Priorities of Messages at Risk We have found that, in general, the existing analysis gives the correct worst-case response times for the highest priority and the 2nd highest priority message. However; it can compute incorrect worst-case response times for messages from the 3rd highest priority to the lowest priority.

Message Omission Many CAN applications allow for 8 data byte diagnostic messages, which are not transmitted during the normal mode of operation. These messages are transmitted only when the system is in diagnostic mode[9] and linked to service equipment. In this section, we consider whether the omission of diagnostic messages provides sufficient reduction in interference / blocking to ensure that messages do not miss their deadlines during normal operation, despite being given potentially optimistic worst-case response times by the existing analysis.

Message Guarantees not at Risk In this section, we consider the circumstances under which the first instance of a message in the busy period is guaranteed to have the longest response time. Under these circumstances, despite its flaws, the existing analysis gives correct results.

Priority Assignment Policies The analysis presented in section 3 is applicable independent of the priority ordering of CAN messages. However, choosing an appropriate priority ordering is important in obtaining a schedulable system and in maximising robustness to errors. Priority ordering is determined by a priority assignment policy. A priority assignment policy *P* is referred to as *optimal* if there are no systems that are schedulable using any other priority assignment policy that are not also schedulable using policy

Future Work A considerable body of academic work has grown up from Tindell's seminal analysis of CAN. The flaws in that original work may have partly undermined some of the subsequent research built upon it. Authors that have cited the original CAN analysis in their work are encouraged to check the implications. In particular the academic work most likely to be affected is that which extends the original analysis and pushes system schedulability to its limits, for example work on error models.

IV. CONCLUSION

In this paper we highlighted a significant flaw in long-standing highly cited and widely used schedulability analysis of CAN. We showed how this flaw could lead to the computation of optimistic worst-case response times for CAN messages, broken guarantees and deadline misses. This paper provides revised analysis that can be used to calculate correct worst-case response times for CAN.

In addition, we showed that:

1. The existing analysis can provide optimistic worst-case response times for messages from the 3rd highest priority to the lowest priority.
2. The existing analysis can lead to broken guarantees and hence deadline misses in systems with low bus utilisation.
3. Where an error model has been considered, the flaw in the existing analysis is not sufficient to lead to CAN configurations that will result in missed deadlines when no errors are present on the bus. The desired robustness to errors may not however be achieved.
4. The omission of a single maximum length diagnostic message, accounted for by the existing analysis, reduces interference / blocking enough to ensure that the deadlines of all the remaining messages are met during normal operation.
5. Despite its flaws, the existing analysis gives the correct response time for any message where there is at least one lower priority message with the same or greater transmission time / message length.

REFERENCES

1. N.C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", *Technical Report YCS 164*, Dept. Computer Science, University of York, UK, December 1991.
2. R.J. Bril. “Existing worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred pre-emption is too optimistic”. *CS-Report 06-05*, Technische Universiteit Eindhoven (TU/e) The Netherlands, February 2006.
3. L. George, N. Rivierre, and M. Spuri. “Pre-emptive and non- pre-emptive real-time uni-processor scheduling. *Technical Report 2966*, Institut National de Recherche et Informatique et en Automatique (INRIA), France, September 1996
4. S. Punnekkat, H. Hansson, C. Norstrom. “Response time analysis under errors for CAN”. In Proceedings 6th Real-Time Technology and Applications Symposium, pp. 258-265, IEEE Computer Society Press May/June 2000.
5. J. Lehoczky. “Fixed priority scheduling of periodic task sets with arbitrary deadlines”. In *Proceedings 11th IEEE Real-Time Systems Symposium*, pp. 201–209, IEEE Computer Society Press, December 1990.
6. K.W. Tindell and A. Burns. “Guaranteeing message latencies on Controller Area Network (CAN)”, In *Proceedings of 1st International CAN Conference*, pp. 1-11, September 1994.