The impact of bit stuffing on the real-time performance of a distributed control system

Mouaaz Nahas, Michael Short and Michael J. Pont,
Embedded Systems Laboratory, University of Leicester

The bit-stuffing mechanism utilised by CAN causes the message transmission time to become (in part) a complex function of the contents of the data fields. This variation in transmission times makes it difficult to predict the precise behaviour of real-time systems implemented using CAN. Previous work in our laboratory has led to the development of a software-based compensation method which significantly reduces the impact of CAN bit stuffing on message transmission times. In the present paper, we focus on the impact of bit stuffing on a system implemented using a "Shared-Clock" scheduling method. We use a detailed Hardware-in-the-Loop (HIL) testbed to explore the behaviour of an Adaptive Cruise Control (ACC) system for use in a passenger car. Through the use of the testbed, we present quantitative results which demonstrate the impact of variations in the message transmission times on the performance of the ACC system. We go on to demonstrate the improvement in performance which results when the previously-mentioned compensation technique is employed. Finally, the memory and CPU resources required to implement this compensation are discussed.

1. Introduction

Over recent years, we have considered various ways in which time-triggered software architectures can be employed in low-cost embedded systems where reliability is a key design consideration (e.g. Pont. 2001; Pont, 2003; Pont and Banner, 2004). Our previous work in this area has focused on the development of both single- and multi-processor designs. In the case of multi-processor designs, we have sought to demonstrate that a "Shared-Clock" (S-C) architecture provides a simple, flexible platform for many systems (Pont, 2001). In such designs, the Controller Area Network (CAN) protocol - introduced by Robert Bosch GmbH in the 1980s (Bosch, 1991) - provides high-reliability communications at low cost (Farsi and Barbosa, 2000; Fredriksson, 1994; Thomesse, 1998; Sevillano *et al.*, 1998). Since the CAN protocol has become widely used in many sectors, such as automotive and automation (Farsi and Barbosa, 2000; Fredriksson, 1994; Thomesse, 1998; Sevillano *et al.*, 1998; Pazul, 1999; Zuberi and Shin, 1995; Misbahuddin and Al-Holou, 2003), most modern microprocessor families now have members with on-chip support for this protocol (e.g. Philips, 1996; Siemens, 1997; Infineon, 2000; Philips, 2004).

In a Shared-Clock CAN (SCC) network, tasks on the slave will suffer from timing jitter if there is any variation in the time taken to send "Tick" messages between the Master and the Slave. One source of such a variation is the frame length, which may be indirectly affected by the bit-stuffing mechanism incorporated in the CAN hardware (see Section 2).

As a result of previous work in this area, we have developed a software-based compensation method, suitable for use with SCC systems, which significantly reduces the impact of CAN bit stuffing: this approach has been shown to be effective on a variety of hardware platforms (Nahas and Pont, 2005).

In the study reported in this paper, we used a detailed Hardware-in-the-Loop (HIL) testbed to explore the impact of bit stuffing on the behaviour of an Adaptive Cruise Control (ACC) system for use in a passenger car. We also considered the costs (in terms of memory requirements and CPU load) of implementing the compensation scheme in real applications.

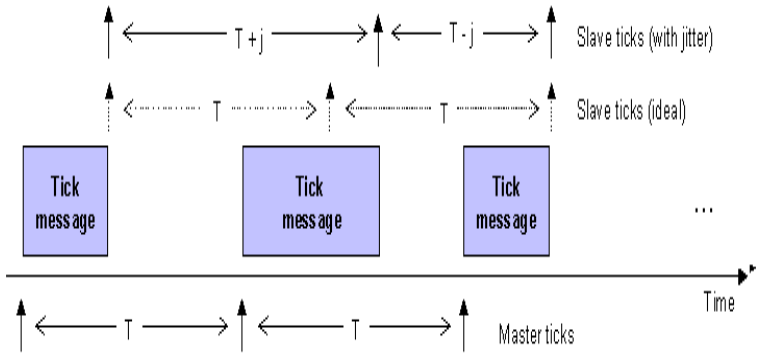

Figure 1: Impact of frame length on the timing of Slave ticks in the SCC system

2. The CAN bit-stuffing mechanism

The CAN protocol uses "Non Return to Zero" (NRZ) coding for bit representation. Under such a scheme, drift in the receiver's clock can occur when a long sequence of identical bits has been transmitted. Such a drift might, in turn, result in message corruption.

To avoid the possibility of such a scenario, the CAN communication protocol (at the physical level) employs a *bit-stuffing* mechanism which operates as follows: after five consecutive identical bits have been transmitted in a given frame, the sending node adds an additional bit, of the opposite polarity. All receiving nodes remove the 'inserted' bits to recover the original data (Farsi, 2000).

Whilst providing an effective mechanism for clock synchronization in the CAN hardware, the bit-stuffing mechanism causes the frame length to become a complex function of the data contents.

It is useful to understand the level of message variation that this process may induce. When using (for example) 8-byte data and extended CAN identifiers, the minimum message length will be 111 bits (without bit stuffing) and the maximum message length will be 135 bits (with the worst-case level of bit stuffing): see Nolte (2003) for details. At the maximum CAN baud rate (1 Mbit/sec), this translates to a possible variation in message lengths of 24 _s.

These variations in message transmission times can have important implications in any real-time systems in which it is important to be able to predict event timing at the microsecond level. For example, in systems using a Shared-Clock scheduler (e.g. Pont. 2001; Pont, 2003; Pont and Banner, 2004), variations in the duration of "Tick" messages can have a significant impact on the levels of task jitter in the Slave nodes (see Nahas et al., 2004). This process is illustrated in Figure 1.

3. Software–based compensation

To reduce the impact of bit stuffing in the CAN protocol on the behaviour of the SCC scheduler, we have developed a software-based compensation algorithm (Nahas and Pont, 2005) in which each frame is encoded (before transmission) in order to reduce the impact of the CAN bit-stuffing mechanism. A decoding process must also take place in the receiver nodes to restore the original data.

4. The testbed

As noted in the introduction, the main aim of this paper was to use a detailed Hardware-in-the-Loop (HIL) testbed to explore the impact of bit stuffing on the behaviour of an Adaptive Cruise Control (ACC) system for use in a passenger car.

ACC is a relatively new technological development in the automotive field, and is said to reduce driver fatigue and the rate

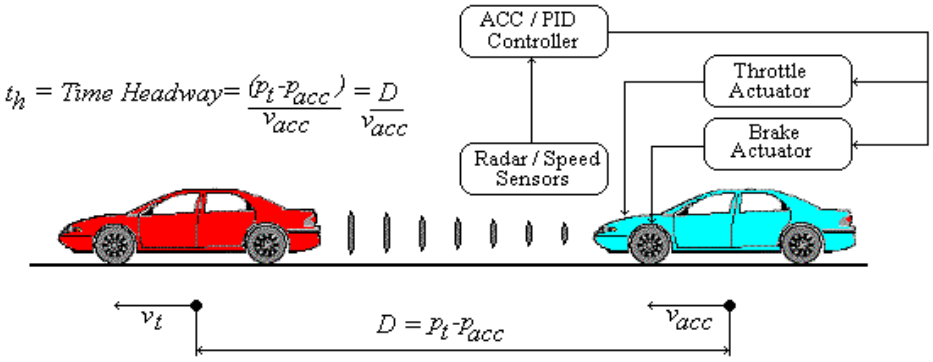

Figure 2: An overview of the operation of the ACC

of auto accidents, whilst increasing fuel efficiency (Stanton, 1997). The main system function of ACC is to control the speed of the host vehicle using information about [1] the distance between the subject vehicle and any forward vehicles; [2] the motion of the subject vehicle; and [3] driver commands.

Based upon the information acquired, the controller sends commands to the vehicle throttle and brakes to either regulate the vehicle speed to a given set value, or maintain a safe distance to a leading vehicle (if the speed of the vehicle in front is slower than the set value). It also sends status information to the driver.

The system under consideration in this study is a Type 2b ACC system: such a system has an automatic gearbox and active braking. Vehicle acceleration is limited to 2.0 m/s^2, deceleration to 3.0 m/s^2 in order to comply with ISO standards (ISO, 2003).

Figure 2 shows the principle of operation. The controller that has been implemented is based of a modified version of that presented by Yi et al (2000) and is shown in schematic form in Figure 3. The ACC testbed was based on Infineon C167CS microcontrollers (one per node) running at a 20 MHz oscillator frequency. Each microcontroller had two on-chip CAN interfaces. In total 10 nodes were used. All the HIL system testing takes place using a realistic test facility developed with the University of Leicester (Short et al. 2004a, b, c).

A schematic of the overall system is shown in Figure 4. The nodes were connected using twisted-pair CAN links running at 500 kbaud.

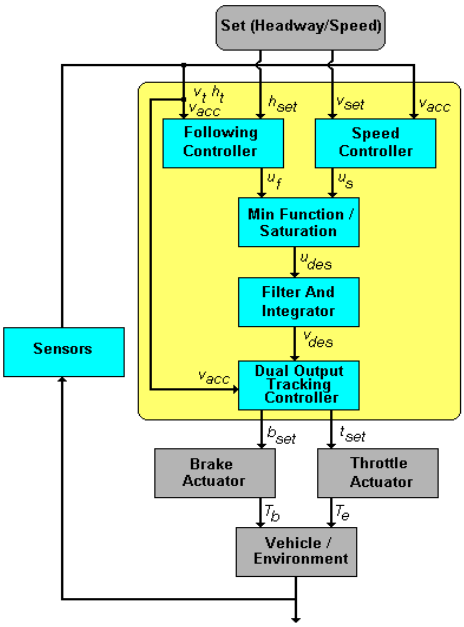

Figure 3: The ACC implementation: adapted from Yi et al. (2000)

5. Experimental methodology

The methodology used to assess the impact of the software-based compensation methodology on the ACC system is discussed in this section.

5.1 Three systems

One consequence of the use of the software-based compensation scheme is that a limit is placed on the amount of user data that may be transferred in each CAN frame, since the encoding process requires two data bytes (Nahas and Pont, 2005).

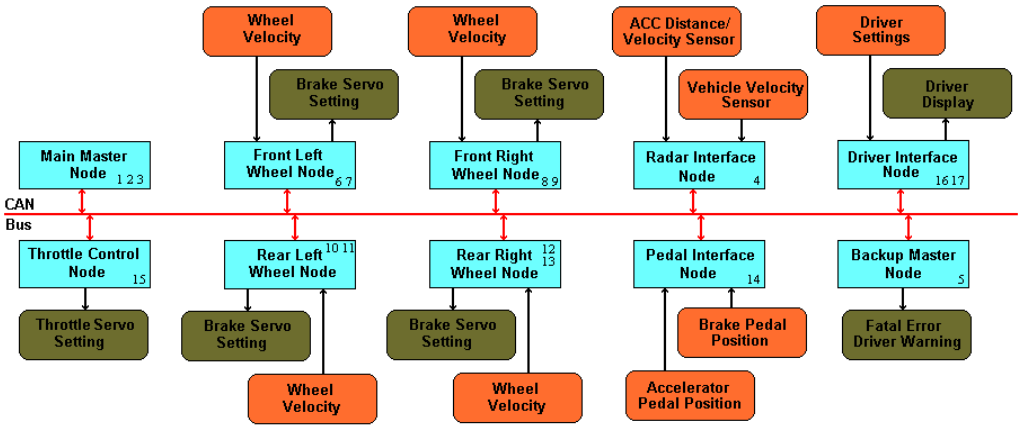

Figure 4: The multi-node ACC implementation

The original system design had to be altered to accommodate the reduced data payload. This, in turn, resulted in a reduction in the sampling rate of the system traction controller. To enable a meaningful comparison, measurements were also taken for an uncompensated implementation using this reduced sampling rate.

To summarise: the three sets of results were obtained. These are labeled as follows: “Original” (the original 8-byte system with no compensation); “Uncompensated” (a 6-byte system with no compensation); and “Compensated” (a 6-byte system with compensation).

5.2 Jitter measurement

To obtain data regarding real-time stability, the latency between Master and Slave clock ticks was recorded for a period of 10,000 samples for each system. To make these measurements, a pin on the Master node was set high (for a short period) at the start of the Master interrupt service routine (ISR). Another pin on the Slave (initially high) was set low at the start of Slave ISR. The signals from these two pins were then AND-ed (using an 74LS08N chip: Texas Instruments, 2004), to give a pulse stream. The widths of the resulting pulses were measured using a National Instruments data acquisition card ‘NI PCI-6035E’ (National Instruments, 2004), used in conjunction with appropriate software LabVIEW 7.1 (LabVIEW, 2004). The average jitter was taken as the standard deviation of the total latency of the entire sample range. Worst Case Transmission Time (WCTT) was represented by the longest delay between the occurrence of a clock Tick on the Master node and the corresponding Tick on the Slaves.

5.3 Jerk and IAE measurement

To provide an indication of the control performance of each system, the maximum positive and negative vehicle ‘jerk’ (rate of change of acceleration) was recorded over a 300 second test period in which the ACCS was put through a series of typical manoeuvres. The jerk was averaged over a 1-second time period in accordance with ISO test specifications (ISO, 2003).

In addition to measuring the vehicle ‘jerk’, the performance of the vehicle while executing speed, and time-gap control was recorded. The IAE (Integral of Absolute Error) criterion was used to provide the performance measure in this case, as defined in Equation 1. The IAE represents the error between the measured speed (or time-gap) and the reference, with the test duration, *T*, equal to 300 seconds.

$$IAE = \int_0^T \left| e(t) \right| \, dt$$

Equation 1

Table 1: Results from the ACC study

Test	Original	Uncompensated	Compensated
IAE Velocity	1.64	1.68	1.53
IAE Distance	11.60	11.84	10.98
Ave. Jitter (μs)	3.66	3.83	2.40
Diff. Jitter (μs)	23.53	24.03	11.20
WCTT (μs)	339.17	339.57	310.77
Max Pos Jerk (m/s^3)	2.24	2.39	2.37
Max Neg Jerk (m/s^3)	-1.81	-1.75	-1.60

Each velocity test was for a speed setpoint of 70 MPH. Each distance test was performed whilst following a lead vehicle at 50 MPH (distance setpoint of 33.53 m for a 1.5 s headway).

6 Results

The results obtained in the studies detailed in Section 5 are presented in this section.

6.1 System performance

Each of the tests detailed in Section 5 was repeated three times, and the results obtained were averaged (see Table 1). Please note that the IAE measurements are “unit less” values, and are best viewed as a performance measure (the lower the better).

It can be seen from the results that the measured WCTT, average jitter and latency difference have all been reduced considerably by the compensation technique. For example, the overall reduction in the difference jitter was approximately 50%.

When comparing the control behaviour of the compensated system to that of the original system, it can be seen that the performance has improved in all areas (by approximately 9%), except in the case of positive jerk.

When comparing the uncompensated system to the original system, it is clear that the control performance of the uncompensated system is comparatively poor: this is a direct consequence of the 25% reduction in the data bandwidth of the network. However, when the compensated and uncompensated systems (with the same bandwidth restrictions) are compared, the use of compensation is seen to improve performance in all areas (including positive jerk).

6.2 Memory and CPU requirements

The compensation operation took an average of 0.7 ms on the processors used, and the corresponding decoding operation had an average duration of 0.6 ms.

The extra RAM required by the compensation technique was 72 bytes and 56 bytes for the Master and Slave, respectively. The corresponding ROM (program memory) increases were found to be 1,317 and 985 bytes respectively, for the Master and Slave.

Please note that the C167 boards used in this study have 256 kBytes ROM and 256 kBytes RAM (PhyCORE, 2003). The overall increases in memory do not, therefore, represent large percentages of the available resources.

7. Discussion and conclusions

From the results presented, it can be seen that applying the software-based compensation technique produces measurable decreases in the variation of the transmission time of CAN messages.

These measurable improvements may have an impact on a particular implementation, such as the control system outlined here, where excessive jitter can cause reductions in performance.

Inevitably, due to the required message coding and decoding, the system places an extra computation and memory load on each microcontroller. In addition, by reserving two data bytes for the compensation information, the overall information throughput of the network is reduced. These factors must be taken into account when deciding whether to implement such a compensation technique.

Acknowledgements

The project described in this paper was supported in part by the Leverhulme Trust (F / 00212 / D), and in part by the UK Government (EPSRC-DTA award). Work on this paper was completed while MJP was on Study Leave from the University of Leicester.

References

Bosch (1991), Robert Bosch GmbH “CAN Specification Version 2.0”.

Farsi, M. and Barbosa, M. (2000) “CANopen Implementation, applications to industrial networks”, Research Studies Press Ltd, England.

Fredriksson, L.B. (1994) “Controller Area Networks and the protocol CAN for machine control systems”, Mechatronics Vol.4 No.2, pp. 159-192.

Infineon (2000) “C167CR Derivatives 16-Bit Single-Chip Microcontroller”, Infineon Technologies.

ISO 15622 (2003) “Adaptive Cruise Control Systems – Performance Requirements And Test Procedures”, International Standards Organisation, Geneva, Switzerland.

LabVIEW 7.1: WWW webpage http://www.ni.com/labview/upgrade.htm [accessed Dec 2004]

Misbahuddin, S.; Al-Holou, N. (2003) “Efficient data communication techniques for controller area network (CAN) protocol”, Computer Systems and Applications, 2003. Book of Abstracts. ACS/IEEE International Conference on, Pages:22.

Nahas, M. and Pont, M. (2005) [submitted]

Nahas, M., Pont, M.J. and Jain, A. (2004) "Reducing task jitter in shared-clock embedded systems using CAN". In: Koelmans, A., Bystrov, A. and Pont, M.J. (Eds.) Proceedings of the UK Embedded Forum 2004 (Birmingham, UK, October 2004), pp.184-194. Published by University of Newcastle upon Tyne [ISBN: 0-7017-0180-3].

National Instruments; PCI-6035E data sheet and specs; WWW webpage: http://www.ni.com/pdf/products/us/4daqsc202-204_ETCx2_212_213.pdf [accessed May 2004]

Nolte, T. (2003) “Reducing Pessimism and Increasing Flexibility in the Controller Area Network”, PhD thesis, Malardalen University.

Pazul, K. (1999) “Controller Area Network (CAN) Basics”, Microchip Technology Inc. Preliminary DS00713A-page 1 AN713.

Philips (1996) “P8x592 8-bit microcontroller with on-chip CAN, datasheet”, Philips Semiconductor.

Philips (2004) “LPC2119/ 2129/ 2194/ 2292/ 2294 microcontrollers user manual”, Philips Semiconductor.

PhyCORE-167 (2003) “QuickStart Instructions”, Phytec Technology.

Pont, M.J. (2001) “Patterns for time-triggered embedded systems: Building reliable applications with the 8051 family of microcontrollers”, ACM Press / Addison-Wesley. ISBN: 0-201-331381.

Pont, M.J. (2003) “Supporting the development of time-triggered co-operatively scheduled (TTCS) embedded software using design patterns”, *Informatica*, 27: 81-88.

Pont, M.J. and Banner, M.P. (2004) “Designing embedded systems using patterns: A case study”, *Journal of Systems and Software*, 71(3): 201-213.

Sevillano J L, Pascual A, Jiménez G and Civit-Balcells A (1998) “Analysis of channel utilization for controller area

networks" Computer Communications, Volume 21, Issue 16, Pages 1446-1451

Short, M., Pont, M.J., and Huang, Q. (2004a) *"Simulation Of Vehicle Longitudinal Dynamics"*, Technical report ESL 04/01, Embedded Systems Laboratory, University of Leicester.

Short, M., Pont, M.J., and Huang, Q. (2004b) *"Simulation Of Motorway Traffic Flows"*, Technical report ESL 04/02, Embedded Systems Laboratory, University of Leicester.

Short, M., Pont, M.J., and Huang, Q. (2004c) *"Development Of A Hardware-In-The-Loop Test Facility For Distributed Embedded Systems"*, Technical report ESL 04/03, Embedded Systems Laboratory, University of Leicester.

Siemens (1997) "C515C 8-bit CMOS microcontroller, user's manual", Siemens.

Stanton, N. A., Young, M. S. & McCaulder, B. (1997) "Drive-by-wire: The case of driver workload and reclaiming control with adaptive cruise control", Safety Science, Vol. 27, pp. 149-159.

Texas Instruments: 74LS08 Datasheet, WWW webpage: http://www.cs.amherst.edu/~sfkaplan/courses/spring-2002/cs14/74LS08-datasheet.pdf [accessed May 2004]

Thomesse, J. P. (1998) "A review of the fieldbuses" Annual Reviews in Control, Volume 22, Pages 35-45

Yi, K., Cho, Y., Lee, S., Lee, J. and Ryoo, N. (2000) "A Throttle/Brake Control Law for Vehicle Intelligent Cruise Control", Seoul 2000 FISITA World Automotive Congress, June 12-15, Seoul, Korea.

Zuberi, K. M. and Shin, K. G. (1995) "Non-Preemptive Scheduling of Messages on Controller Area Network for Real-Time Control Applications", in Proc. Real-Time Technology and Applications Symposium, pp. 240-249.

Mouaaz Nahas
Embedded Systems Laboratory,
University of Leicester,
University Road,
Leicester LE1 7RH,
UK.
Phone: ++44 (0) 116 252 2578
Fax: ++44 (0) 116 252 2619
mn59@le.ac.uk

Dr Michael Short
Embedded Systems Laboratory,
University of Leicester,
University Road,
Leicester LE1 7RH,
UK.
Phone: ++44 (0) 116 252 2578
Fax: ++44 (0) 116 252 2619
mjs61@le.ac.uk

Dr Michael J. Pont
Embedded Systems Laboratory,
University of Leicester,
University Road,
Leicester LE1 7RH,
UK.
Phone: ++44 (0) 116 252 5052
Fax: ++44 (0) 116 252 2619
M.Pont@le.ac.uk
http://www.le.ac.uk/engineering/mjp9/