

Experimental Jitter Analysis in a FlexCAN Based Drive-by-Wire Automotive Application

Juan R. Pimentel

Kettering University

1700 W. Third Ave.

Flint, MI 48504, USA

+1-810-762-7990

jpimente@kettering.edu

Jason Paskvan

Mentor Graphics Corporation

27725 Stansbury Blv., Suite 295

Farmington Hills, MI 48334

+1-810-248-699-1022

Jason_Paskan@mentor.com

ABSTRACT

In this paper, we describe several experiments designed to characterize jitter in an actual automotive application designed using FlexCAN, a CAN based communication architecture. Large and variable jitter has been a liability of the CAN protocol. The paper also explains how FlexCAN reduces jitter by using a simple message scheduling technique that synchronized with control system applications.

Categories and Subject Descriptors

C.2.4 [Distributed Systems]: Distributed applications.

General Terms

Measurement, Performance, Experimentation.

Keywords

CAN, FlexCAN, Jitter, Automotive, Drive-by-Wire.

1. INTRODUCTION

CAN is a mature protocol that has become important for many small area applications, specially automotive, mainly due to its error control features, low latency, network wide bus access priority and instant bit monitoring [1,2]. In addition to the aforementioned important features, CAN offers other excellent control features to recover from frame errors (including stuff bit errors, and CRC errors) which are not reviewed here as the reader is referred to the specification [3]. Although there are numerous successful application areas for the CAN protocol, some other applications could benefit if CAN could be made faster, cover longer distances, be more deterministic and more dependable [4]. In fact, because of its limited dependability features, there is an ongoing debate on whether the CAN protocol, with proper enhancements, can support safety-critical applications [5, 6].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2007, June 4–8, 2007, San Diego, California, USA.

Copyright 2007 ACM 978-1-59593-627-1/07/0006 ...\$5.00.

One of the most interesting features of CAN is its bandwidth efficient medium access protocol, based on network-wide fixed priorities, which allows each node to try to transmit at any instant. However, this feature also has the negative side effect of causing a substantial network delay jitter because, without synchronization of the transmission instants, any node will encounter all possible interference patterns from higher priority traffic when trying to transmit. The presence of significant jitter can have a detrimental impact on the performance of many distributed embedded systems. This jitter can be controlled, and minimized, using global synchronization and relative offset adjustments. A low and fixed jitter would make CAN highly deterministic. In the last decade, there has been several proposals to make CAN more deterministic and dependable [4]. One such proposal is the FlexCAN architecture that basically combines the best features of FlexRay and CAN [7].

The aim of this paper is twofold, first, to characterize the time-triggered structure of FlexCAN and explain how it helps to reduce message latency jitter, and second to report some experimental results that measure latency jitter for messages in a FlexCAN-based drive-by-wire application.

2. JITTER IN CAN

There are three sources of jitter in CAN message latencies: due to bit stuffing, due to jitter in scheduled tasks, and due to the dynamic mixture of time triggered (e.g., periodic) and event triggered (i.e. aperiodic) traffic.

Jitter due to bit stuffing has been extensively characterized in several publications and this jitter cannot be eliminated although it can be minimized in some cases. To help bit synchronization at the physical level, when five identical bits (e.g. five 0s) have been transmitted on a CAN bus, a bit of the opposite polarity is “stuffed” (transmitted) by the sender. The receiver follows the reverse mechanism and removes the stuffed bit after the specified number of identical bits, thereby restoring the original data stream. Clearly, bit stuffing can have an impact on the message length and, hence on latency jitter. From the viewpoint of the level of jitter introduced by bit stuffing, the data portion in a CAN frame (from 0 to 8 bytes) may contain “best case”, “worst case”, or “random” data. The best-case data is a sequence of 10101010... since such data requires no bit stuffing. The worst-case data occurs when the data is 11110000011111..., since this causes the maximum level of bit stuffing. Nevertheless the jitter due to bit stuffing is bounded and its maximum size at the highest data rate (1 Mbps) is 24 μ s [8].

The latency jitter due to the jitter in the scheduled tasks is due to variations in the time to actually execute software tasks in a node. It is assumed that these software tasks are responsible for sending CAN messages thus they contribute to message latency jitter. Clearly this jitter is highly dependent on details of the operating system (if any) or task executives in charge of task scheduling. This kind of jitter has also been extensively studied and can also be minimized by proper scheduling techniques [9]. The message latency jitter due to the dynamic mixture of time triggered and event triggered traffic results from periodic messages waiting for higher priority event messages that arrive to the system at arbitrary and unpredictable times. This type of jitter is important for automotive networks as typical traffic is a combination of both periodic and non-periodic (i.e., event) traffic. Most protocols support periodic and non-periodic traffic; indeed both FlexRay and FlexCAN support this mixture in an explicit fashion.

Latency jitter in CAN has been studied in generic terms independent of its applications. FlexCAN includes a task scheduling mechanism that is synchronized to control system applications that is prevalent in automotive applications. The synchronization is done using the concept of control system transactions (CSTs) as described below. Thus FlexCAN enables the study of jitter in the context of actual applications.

3. SUMMARY OF FLEXCAN

The goal of the FlexCAN architecture is to provide additional deterministic and dependable capabilities to the CAN protocol without compromising its flexibility. As depicted in Figure 1, the FlexCAN architecture can incorporate several replicated channels and several replicated nodes (in addition to normally non-replicated ones) in a flexible fashion [5]. Time-domain composability has been a major liability of CAN-based networks for safety-critical applications. FlexCAN adopts the time-triggered paradigm by using reference messages. The interval between reference messages is called the *communication cycle*, and this interval is further divided into a number of *sub-cycles* as shown in Figure 2. The communication cycles, together with their sub-cycles, not only help with time-domain composability but also help to synchronize replicated nodes and channels, and to enforce fail-silent behavior. Furthermore, communication cycles are useful to synchronize applications. FlexCAN is flexible in that it supports both periodic, a-periodic, and multirate message traffic.

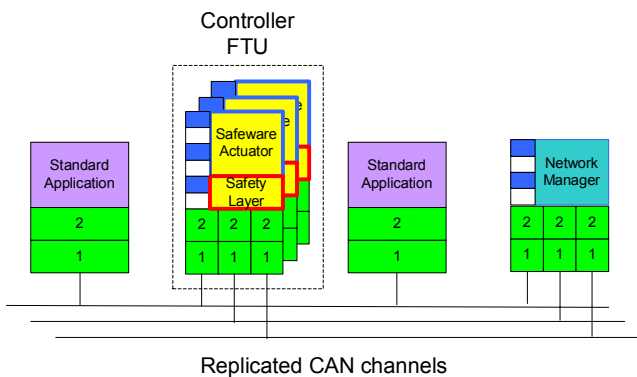


Fig. 1. The FlexCAN Architecture.

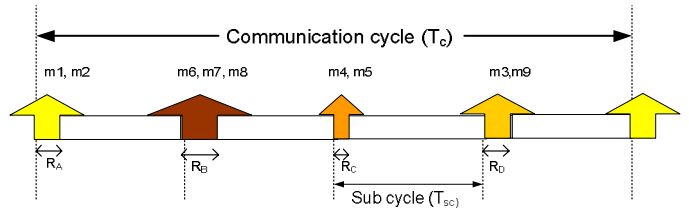


Figure 2. Time triggered structure of the FlexCAN architecture.

FlexCAN requires a synchronization message to explicitly mark the beginning of each cycle without the need of clock synchronization. FlexCAN relies on timers to divide the entire cycle into Q (e.g., 4) sub-cycles.

Time synchronization between a TT global message schedule and end applications are crucial for most control systems. FlexCAN enables the synchronization of any CST event (E1 through E8 in Figure 3) to the communication cycle. Figure 3 depicts the synchronization of message m1 with the beginning of the communication cycle while other messages are synchronized with the beginning of sub-cycles (e.g., m3, m4 or m6).

3.1 How FlexCAN Addresses Deterministic Limitations

It is easy to design and verify message determinist properties on a system based on a TDMA system such as FlexRay because messages occupy specific slots in the communication cycle. It is also possible to design and verify deterministic properties on a CAN based system although the process is more complex because of a lack of a time reference in the CAN protocol and because of widely different message transmission rates (or periods) that are possible.

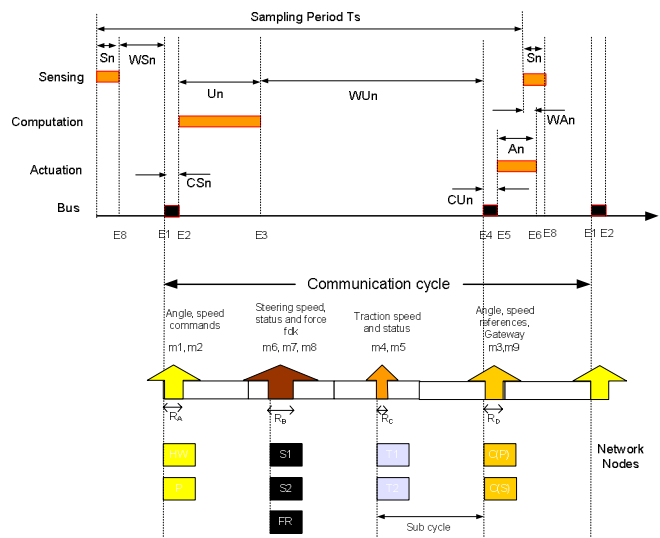


Figure 3. FlexCAN global message schedule synchronized to a CST functional unit.

One difficulty with CAN latency calculations stems from the usage of CAN in early applications where a CAN network was the only network in the system and thus had to support all messages with widely varying message transmission periods. In fact, the application used in [11] has 6 messages with the smallest and largest period of 2 and 240 msec respectively, a factor of 120. With such widely differing factors in the transmission periods, the number of times a message is sent in a calculation period is not known ahead of time and must be calculated. This leads to a convoluted set of equations in closed form that must be solved recursively. Current and future systems have several communication networks each supporting an application type (e.g., entertainment, dash-board, powertrain, steer-by-wire, etc.). These networks are typically configured as a backbone network and several sub-networks each dedicated to one or few functional units of a vehicle. Messages in the sub-networks do not have widely varying message transmission periods, in fact, factors of 4 to 8 are sufficient. Another difficulty stems from the event triggered nature of the CAN protocol that does not use the notion of global time. Because of this, there is uncertainty in the assumptions of values for the queuing jitter and the interference due to lower priority messages.

FlexCAN overcomes these previous two limitations by defining a sub-network for messages with closely related transmission periods and also by defining a time-triggered time reference made of communication cycles divided into a number of sub-cycles. Just as is the case with other TT architectures, FlexCAN requires an off-line global message schedule to be configured. Furthermore, all messages scheduled in a sub-cycle are submitted for transmission at exactly the same time (at the beginning of the sub-cycle).

The above described features greatly simplify the equations for calculating message latencies in the FlexCAN architecture [10]. To begin with, message latency calculations are done on a sub-cycle basis for all sub-cycles. For each sub-cycle, the messages are re-labeled as m_1, m_2 , etc. with m_1 the highest priority message, m_2 the next highest priority message, and so on. Because all messages in each sub-cycle are submitted for transmission at exactly the same time, the jitter is 0 and there is no blocking from lower priority messages, resulting in simplifications in the calculation of message latencies.

Calculating message response times in FlexCAN has the following advantages when compared to that in CAN. First, the calculations are more accurate since they are done on a sub-cycle basis relative to a time-triggered communication cycle with each sub-cycle being independent from the next (i.e., all messages are sent in their respective sub-cycles). Another advantage is that the calculations are easy to evaluate as the equations are not in closed-form. Still another advantage is that the calculations assume that the jitter is zero which is enforced by the FlexCAN message schedule. A further advantage is that the formulae are accurate because there is no uncertainty regarding the blocking due to lower priority messages. A final advantage is that the multi-rate case is taken into account ahead of time by the FlexCAN message global schedule.

3.2 Jitter in FlexCAN

FlexCAN does not try to minimize latency jitter due to bit stuffing because it is inherent to the CAN protocol. However, FlexCAN minimizes message latency jitter due to the jitter in scheduled

tasks and due to the dynamic mixture of time triggered and event triggered traffic. Both, the message latency jitter due to the jitter in scheduled tasks and that due to the dynamic mixture of time triggered and event triggered traffic are minimized by the time-triggered schedule used by FlexCAN that is based on CSTs of the various functional units in an application. For the Padova truck DbW application, such a schedule is shown in Fig. 3.

4. EXPERIMENTAL MEASUREMENTS OF JITTER IN A FLEXCAN BASED DRIVE BY WIRE APPLICATION

In this section, we report some experimental results designed to measure latency jitter for messages in a FlexCAN-based drive-by-wire (DbW) application that includes periodic and non-periodic traffic. The DbW application correspond to the steering and acceleration sub-systems of a re-designed commercial lift truck [12]. The FlexCAN network for this application is depicted in Fig. 4, although for the experiments, just a single CAN bus and no replication of the Control node was used. The details of the messages are listed in Table 1 that also indicates the nodes used to send the messages. All of the messages in Table 1 are periodic except M10 which sent on an event basis. Five Freescale HCS12 microcontrollers (S12-1 through S12-5) and three workstations equipped with CANoe (PC1 through PC3) were used for the experiments. An additional S12 microcontroller, not shown in Table 1, was used to explicitly generate reference messages for implementing the time-triggered structured of Fig. 2. Four reference messages m_A, m_B, m_C , and m_D with CAN identifiers $0x0, 0x1, 0x2$, and $0x3$ respectively were used to delineate sub-cycles 1, 2, 3, and 4 of each communication cycle. The data rate is 500 Kbps.

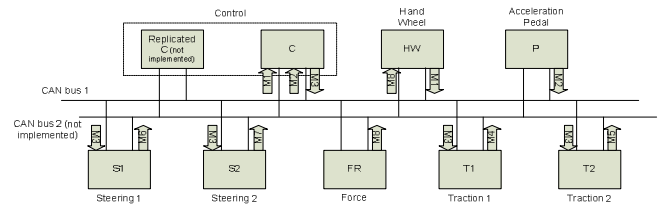


Figure 4. FlexCAN network layout for the drive-by-wire application.

Table 1. Messages in the Drive-by-Wire application.

| Msg | Node | Functional Description | Size in bits | CAN ID |
|-----|-------|--|--------------|--------|
| M1 | S12-1 | Steering angle command | 32 | 0x100 |
| M2 | S12-2 | Acceleration command | 32 | 0x300 |
| M3 | S12-3 | Steering angle plus acceleration reference | 64 | 0x200 |
| M4 | PC1 | Traction-1 speed and status | 56 | 0x400 |
| M5 | PC2 | Traction-2 speed and status | 56 | 0x480 |
| M6 | S12-4 | Steering-1 speed and status | 56 | 0x500 |

| | | | | |
|-----|-------|-----------------------------|----|-------|
| M7 | S12-5 | Steering-2 speed and status | 56 | 0x580 |
| M8 | S12-4 | Force feedback | 32 | 0x600 |
| M9 | S12-3 | Gateway message | 64 | 0x280 |
| M10 | PC3 | Event message | 64 | Var. |

The PC3 node was used to generate event traffic on a random basis. The experiments measure the latency jitter of some periodic messages in three situations: the traffic is only periodic, mixed traffic where event messages have lower priority than the periodic message in question, and mixed traffic where event messages have higher priority than the periodic message in question. For all experiments, the jitter of a particular message was measured by observing the jitter on the instances that the message in question appears on the bus. Detailed data was logged using CANoe for an interval of 8 seconds and statistical analysis was then performed on the logged data to calculate the standard deviation of the jitter for each experiment.

Experiment 1: Only periodic traffic. For this experiment, the standard deviation of the jitter for message m6 was 157 μ s.

Experiment 2: mixed traffic where event messages have lower priority than the periodic message in question. Workstation PC3 was programmed to generate non-periodic messages of maximum size (i.e., 8 bytes) with priority 0x680, i.e., lower priority than any message in the application. The non-periodic messages (event traffic) were generated using a uniform random distribution with inter-arrival times between 2 and 11 ms. The standard deviation of the jitter for message m6 was 148 μ s. This result makes sense as the schedule of the periodic messages are not affected by the event messages that have lower priority.

Experiment 3: mixed traffic where event messages have higher priority than the periodic message in question. The event traffic for this experiment was the same as that for experiment 2 except that the CAN ID was 0x010, i.e., the priority was higher than any other application message in the system. The standard deviation of the jitter for message m6 was 187 μ s. Table 2 summarizes the results of the experiments.

Table 2. Summary of experiments 1, 2 and 3.

| Experiment | Traffic | Jitter of m6 (Stand. Dev.) | Peak Load | Event msg ID |
|------------|----------|----------------------------|-----------|--------------|
| 1 | Periodic | 157 μ s | 11.55% | ----- |
| 2 | Mixed | 148 μ s | 15.40% | 0x680 |
| 3 | Mixed | 187 μ s | 15.36% | 0x010 |

5. CONCLUSIONS

There are three sources of jitter in CAN based protocols: bit stuffing, task schedulers, and interference from dynamic and

unpredictable higher priority traffic. The FlexCAN architecture aims at making CAN more predictable and more dependable. The simple message scheduling used by FlexCAN helps reduce jitter and thus makes CAN more predictable. The message schedule for a FlexCAN based drive-by-wire application has been implemented in the laboratory yielding relatively low values of jitter. These experiments demonstrate that jitter can be easily controlled in the framework of the FlexCAN architecture.

6. REFERENCES

- [1] SAE Automotive Engineering International, May 2006.
- [2] Tindell, K., Burns, A., and Wellings, A.J., Calculating Controller Area Network (CAN) Message Response Time, *Control Engineering Practice*, 3(8), 1163-1169, 1995.
- [3] CAN Specifications version 2.0, *Robert Bosch gmbh*, 1991.
- [4] Proenza, J., Miró-Julià, J., MajorCAN: A Modification to the Controller Area Network Protocol to Achieve Atomic Broadcast, *ICDCS Workshop on Group Communications and Computations*, 2000: C72-C79
- [5] Pimentel, J.R. and Fonseca, J.A., FlexCAN: A Flexible Architecture for highly dependable embedded applications, *RTN 2004 – 3rd Int. Workshop on Real-Time Networks*, held in conjunction with the 16th Euromicro Intl. Conference on Real-Time Systems, Catania, Italy, June 2004.
- [6] Rufino, J., Verissimo, P., and Arroz, G., Node failure detection and membership in CANELy, *Proceedings of the 2003 International Conference on Dependable Systems and Networks*, 331.340, June 2003. IEEE.
- [7] Pimentel, J.R. and Kaniarz, J., A CAN-Based Application Level Error-Detection and Fault-Containment Protocol, *11th IFAC Symp. on Information Control Problems in Manufacturing (INCOM)*, Salvador, Brazil, 2004.
- [8] Nolte, T., Hansson, H., and Norstrom, C., Minimizing CAN response-time jitter by message manipulation, *8th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS'02)*, 197-206, Sept. 2002.
- [9] Cottet, F. and David, L., A solution to the time jitter removal in deadline based scheduling of real-time applications, *5th IEEE Real-Time Technology and Applications Symposium - WIP*, Vancouver, Canada, pp. 33-38
- [10] Pimentel, J.R., Simonot-Lion, F., and Song, Y.Q., Safety Integrity of FlexCAN Based X-by-Wire Architectures Subject to Random External Disturbances, *INRIA-LORIA Research Report*, under preparation.
- [11] Broster, I., Burns, A., and Rodriguez-Navas, G., Comparing Real-time Communication under Electromagnetic Interference, *16th Euromicro Conf. on Real-Time Systems (ECRTS'04)* – Vol. 00, 45-52, 2004.
- [12] Mertoluzzo, M., Buja, G., and Pimentel, J. R. Design of a Safety-Critical Drive-by-Wire System Using FlexCAN, *SAE Congress*, Paper No. 2006-01-1026, April 2006.