Can controller implementing features for reliable communication[1]

J.C. Campelo, A. Rubio, F. Rodríguez, J.J. Serrano
Dept. of Computer Engineering, Technical University of Valencia (SPAIN)
{jcampelo, alicia, prodrig, jserrano}@disca.upv.es

CAN networks are becoming one of the most used industrial local area network in many applications. Philips Semiconductors has been one of the manufacturers devoted to offer stand alone CAN controllers to make possible the connection to this network. The PCA82C200 has been used in multitude of CAN based microcontroller systems. At present, this circuit has been replaced by the SJA1000. This new controller offers new interesting features. An analysis of this new CAN controller, and its comparison with its antecedent is done in this paper. We use different methods, as simulation languages and queuing networks, in order to obtain the main performance parameters of these controllers.

Introduction

CAN (*Controller area network*) networks are becoming one of the most used industrial local area network in many applications. Beyond its original use, in automotive environments, this network can be found nowadays in many industrial applications, as the base of distributed industrial control systems, in agricultural or medical systems, and many other examples that need an industrial local area network (*fieldbus*).

From the origins of this network, Philips Semiconductors has been one of the manufacturers devoted to offer integrated circuits to make possible the connection to this network. This manufacturer has developed, in addition to different microcontrollers with on-chip CAN controllers specific circuits, known as *stand alone CAN controllers*, in order to provide a simple interconnection to CAN for different microcontrollers.

The PCA82C200 ([6]) has been used in a plethora of CAN based microcontroller systems. It is easily connected to both Intel-like and Motorola-like microcontrollers. At present, this circuit has been substituted by the SJA1000 ([1]). This new CAN controller offers new interesting features.

An analysis of this new CAN controller, and its comparison with its antecedent, is done in this paper. *What does this circuit provide? Which is its effect on the system performance? Has some features that makes it more suitable in real-time systems?* These questions are going to be answered in this article.

In order to do a performance analysis we can use different techniques. We can obtain the results using a real system (prototype) using these controllers. So, with the aid of a monitorization system, we can measure different factors: bus utilization, throughput, response time, etc. and do the comparison between these controllers. The drawback of this technique resides in the development of the monitorization system. To avoid the use of prototypes and the development of measurement systems we can do the study using simulation and analytic methods. Simulation languages and queuing network theory are valid methods in order to compare and to analyse systems.

[1]This work is supported by the Spanish *Comisión Interministerial de Ciencia y Tecnología* under project CICYT-TAP96-1090-C04-01

This paper is organized as follows: after the introduction an analysis of the most important characteristics of both devices is carried out. In the third section simulation models accomplished are commented and, in the last sections the obtained results and conclusions are presented.

PCA82C200 and SJA1000 characteristics

In this section the characteristics that differentiate both circuits are going to be presented. As cited previously, the last one is the successor that Philips Semiconductors offers to its widely used PCA82C200. Since the objective of this paper is to analyze how the new improvements introduced in the SJA1000 affect the system performance only those with have a clear effect in the performance are emphasized.

PCA82C200, due to its low cost and its easy connection with different microcontrollers has become one of the most used CAN controllers in microcontroller based systems. Nevertheless, the continuous development of both microcontrollers and the improvements emerged to the original CAN standard, has made necessary its replacement. Specifically, the 82C200 is only prepared for *basic CAN* and it is *CAN* 2.0B passive. Furthermore, the development of microcontrollers, the constant increase in its work frequency implies that PCA82C200 can not be connected due to its read/write access time to the last microcontrollers. For example, PCA82C200 runs without problems with the Intel 8031 family with frequencies up to 16 Mhz. If we want to use new, faster microcontrollers, only those that permit to insert wait states (and therefore loosing performance) can be used.

This last characteristic is one of the improvements of the SJA1000: access time, reading and writing, setup, hold and pulse width of control signals are now much shorter. It makes possible its direct connection to the last microcontrollers of several manufacturers (think about Intel 251 family, Philips 8051XA, Siemens 166 family, etc.)

Both devices are exactly equal with respect to their package, making possible the direct replacement of the PCA82C200 with the SJA1000. Furthermore, to allow its use on already designed systems based on the PCA82C200 functionality, SJA1000 provides a compatible operation mode with the previous one.

Reception buffer

One of the improvements provided by the SJA1000 is an enhanced reception buffer. While the PCA82C200 only has a two message reception buffers, in the SJA1000 this capacity has been increased until 64 bytes. Depending on the length and type of the message (extended or basic CAN) it will be able to store several messages (more than two). This feature, which does not influence on the application developed, is claimed to be an important advantage due to the smaller probability of overrun errors. Later, in this paper, the number of lost messages due to overrun errors is going to be studied for each of these devices.

PeliCAN mode

In addition to the compatible mode, SJA1000 offers a new mode known as *PeliCAN*. Main new features are ([1]):

- Reception and transmission of standard and extended frame format messages
- Receive FIFO (64 bytes) (also in compatible mode)
- Single/Dual acceptance filter with mask and code register for standard and extended frame.
- Error counters with read/write access
- Programmable error warning limit
- Last error code register
- Error interrupt for each CAN Bus error
- Arbitration lost interrupt with detailed bit position
- Single-shot transmission (no re-transmission on error or arbitration lost)

- Listen only mode (monitoring of the CAN bus, no acknowledge, no error flags)
- Hot plugging supported (disturbance-free software driven bit rate detection)
- Disable CLK OUT by hardware

From these characteristics, the enhanced reception buffer and the new *single-shot* are the improvements that can affect the system performance. So, with *single-shot* mode, if a message has not been able to transmit correctly due to:

- arbitration loose
- bus errors[2]:
    - *bit errors*
    - *bit stuff errors*
    - *CRC errors*
    - *form errors*
    - *acknowledge errors*
    - *overload errors*
    - *overload frame form errors*
    - *inconsistent overload errors*
    - *multiple consecutive errors*
    - *multiple successive error*

it is not re-transmitted. In this case an interrupt is generated and the buffer is released.

If *single shot* mode does not exist (e.g. PCA82C200), the message that is not correctly transmitted is automatically re-transmitted. This implies that, depending on the message priority, it will compete in the next arbitration phase with the rest of the messages of the other nodes and it will delay other messages in the same node. In real-time systems this fact implies that worst case messages response time is unbounded.

With *single shot* mode, the message that can not correctly arrive is lost (in many applications it is “better not to arrive than to arrive late” since the passage of time may invalidate the information of the message [10]) (of course, application level can re-transmit that message if needed). We are going to assume in the rest of the paper that a message that suffers from some error is not re-transmitted. In this way, the response time for the other messages will not be altered due to an error in one of them. (We are analysing only the circuit, not other software levels such as the application level of the communication that can alter this operation)

Simulation models

In order to compare the different behaviors of both CAN controllers, the transmission and the reception will be studied separately.

Transmission

From the transmission point of view, the features to evaluate will be:

- The response time of the messages: the time since the message is ready to access the bus until it reaches to its destination. The aim of this comparison is to check whether this time remains constant or if it increases as the number of errors increase. So, the response time of each of the messages proposed in the *SAE benchmark* [9] will be obtained.

- Bus utilization: the relationship between the total busy time and the time employed in data transmissions (both the ones which arrive successfully and the transmissions which suffer from some error).

- The number of lost messages: this value will inform us about the number of messages that have not been correctly received due to some kind of error.

- The number of messages that miss their *deadline*: the number of messages that have been received but have not met their temporal constraints.

We used the SMPL [4] simulation language to develop the simulation model. The first step was to implement the model of the CAN bus behavior, then, we introduced a fault rate. This fault rate will

be modified in order to study how the parameters evolve.

The objective of this model is to compare the value of the previously cited parameters according to the use of the *single shot* mode (SJA1000) or the normal operation of the PCA82C200.

The simulation model used the *SAE benchmark* [9] as workload. This benchmark describes the set of messages used in an electrical car prototype composed of seven subsystems: the batteries, the vehicle controller, the inverter/motor controller, the instrument panel display, driver inputs, brakes and the transmission control. The network connecting these seven subsystems is required to handle a total of 53 messages, both sporadic and periodic signals. A periodic message has a fixed period, and requires the latency to be less than or equal to this period. Sporadic messages have latency requirements imposed by the application. There is some unspecified behavior in the benchmark: the maximum rate at which sporadic messages can occur as well as the queuing jitter values. In this study, the same values as in Tindell *et al.* [7] will be assumed, and, in order to reduce the bus utilization, piggybacking for all the messages sent from the same source will be employed. For sporadic signals the approach is to send a *server* message periodically (when the server message has to be sent, the sender task polls for the occurred signals and fills the contents of the message).

Reception

The main objective in this point is to compare the number of *overruns* in the two CAN controllers. The 64 bytes reception buffer of the new CAN controller (SJA1000) should be responsible of an important decrease of this number.

When there is no place in the reception buffer to allocate a new message an *overrun* error is produced. There is certainly a little improvement in the new CAN controller: in the PCA82C200 the *overrun* error is activated when there is no place to allocate the new message, just after it has successfully passed the acceptance test, while in the SJA1000 this error is activated when a correctly arrived message has no place in the reception buffer. So, a message that does not fit in the reception buffer but suffers from some kind of bus error does not cause an *overrun* error.

An important aspect to take into account if we want to know when an *overrun* error will be produced is the time the microcontroller needs to release the reception buffer. Depending on the microcontroller, its clocks frequency, and its application (the number of interrupts it has to attend,...), different performance can be obtained.

When a new message arrives, the CAN controller generates an interrupt to the microcontroller. After the interrupt latency time, the microcontroller executes the interrupt handler that gets the message from the CAN controller and usually stores it in its memory in order to process it afterwards. Therefore, the number of *overrun* errors depends on the time the microcontroller takes to remove the message.

To compare both CAN controllers we use an 8031 microcontroller from Intel that usually operates at 12 or 16 Mhz. As previously exposed, there are several microcontrollers of this family, from different manufacturers, which operate at higher frequency but the PCA82C200 does not grant its time requirements. Only microcontrollers that are able to insert wait states in the access to the CAN controller could be employed if we want to operate at higher frequencies. It also supposes a greater cost. It would be possible, for example, to connect the PCA82C200 with the new 251 Intel microcontroller, the successor of the 8031, but this implies that one wait state has to be added. As the aim of our study is to test the functionality of the 64 bytes reception buffer of the new SJA1000, we will base our models on the standard 8031 at 16 Mhz even though some comments about the 251 will be made.

In this sense, if we will know how much time the 8031 microcontroller will need to remove the message from the CAN controller, the next equation (obtained from a real system developed by the authors) could be used: ((112+5L)/Mhz)*12 μs, where L is the number of bytes of the message and Mhz the crystal frequency of the microcontroller. This equation could be different depending on the specific implementation, the optimizations of the employed compiler, or if we program in assembler or a high level language. If we employ the 251 microcontroller, to access the CAN controller one wait state has to be added. So, the following equation has to be employed: ((211+10L)/Mhz)*2 μs.

With these ideas, a simulation model has been developed. In this case we used the QNAP2 (Queuing Network Analysis Package) [5] language to implement the model, and the parameters modified were the message arrival rate and the message length (from 1 to 8 data bytes). First, we modeled a general environment with the arrival of messages of different lengths at the highest frequency allowed in CAN. This model was later modified to obtain a bound of the worst performance (the highest *overrun* probability): the reception of messages of 0 data bytes, without *stuff* bits, at the highest frequency (one message each 47 μs) and at the highest CAN transmission speed (1 Mbaud). Another of the studied aspects was the performance of both CAN controllers in the presence of message bursts. In this case we modified the model to study different burst intervals, number of messages per burst and size of the messages. We obtained an estimation of the suitable message burst intervals from the *SAE benchmark*. To compare the message reception in both CAN controllers we used the *overrun* probability, i.e., the probability to discard a new message due to the lack of space in the reception buffer.

Results

Transmission

From the transmission point of view, the message response time and bus utilization changes are analyzed. In Figure 1 the response time of message 9 from the standard SAE benchmark can be seen. While in the PCA82C200 (200-Msg 9) this time increases when a different error occurs, in the SJA1000 the response time remains constant. So, with the new CAN controller the time the system needs to send successfully a message can be bounded. This is very important in real-time environments.

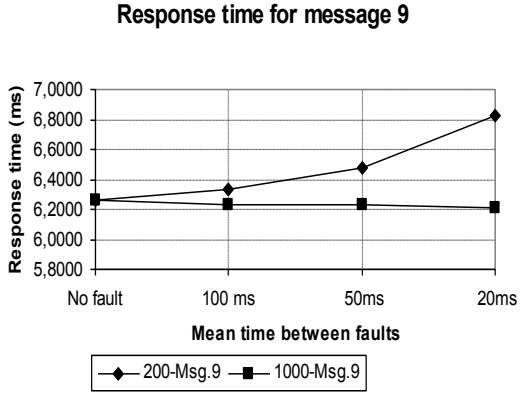

Figure 1: Response time of message 9

In the developed simulations, a transmission speed of 125 Kbaud was used. At this transmission speed we obtained a 85% bus utilization [7,8]. These circumstances (very high bus traffic) can help us to study how the previously commented parameters evolve depending on the CAN controller we use.

In Figure 2 another interesting aspect can be observed: when the error rate is too high, e.g. faults with an exponential distribution of 20 ms as average or lower, it is possible that the response time with the SJA1000 not only remains constant but also decreases. For example, in Figure 2 the response time decreases only a little when the fault rate is not very high, but when it increases (20 ms between two faults), the response time significantly decreases.

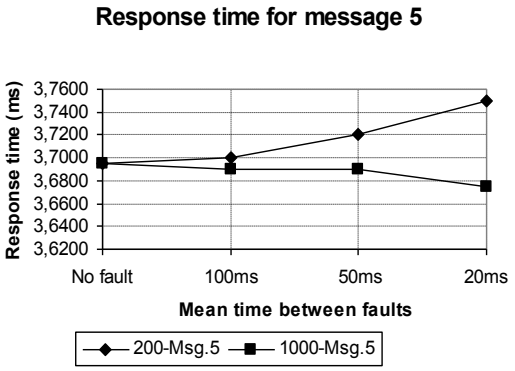

Figure 2: Response time of message 5

In the SJA1000 when an error occurs while a message is being transmitted, it is automatically discarded (even though it has not been completely sent since errors can appear during the transmission of the first bits of the message). So, bus utilization can decrease and messages response time can also be slightly improved. The results of the PCA82C200 show the same behavior as the one earlier observed in Figure 1: the response time notably increases.

In Figure 3 a similar behavior can be observed. In this figure the bus utilization changes as the fault rate increases can be seen.

Just like the response time, in the PCA82C200 the bus utilization increases as the fault rate does (due to the message retransmissions and the transmission of error frames) while in the SJA1000 it remains constant and even decreases when the fault rate increases.

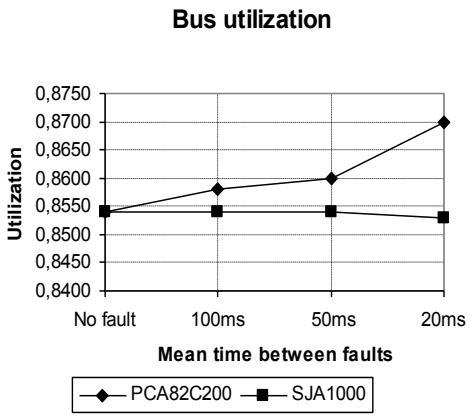

Figure 3: Bus utilization

But there are not only advantages. In Figure 4, the number of messages that have not been sent can be observed, i.e. messages that suffered from some error and therefore not successfully arrived, in the SJA1000 are relatively high. Nevertheless, in the worst case the number of lost messages does not exceed the 2.8%.

Depending on the application goal to develop it could be more interesting to bound the response time even loosing some messages, or to send all the messages increasing the response time (and therefore missing some *deadlines*).

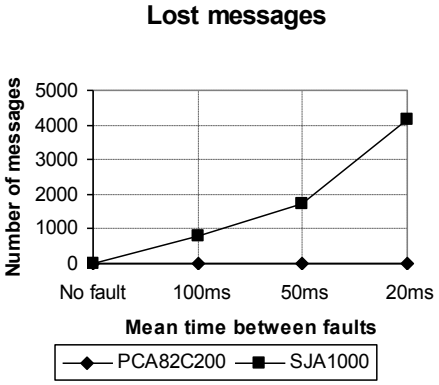

Figure 4: Lost messages

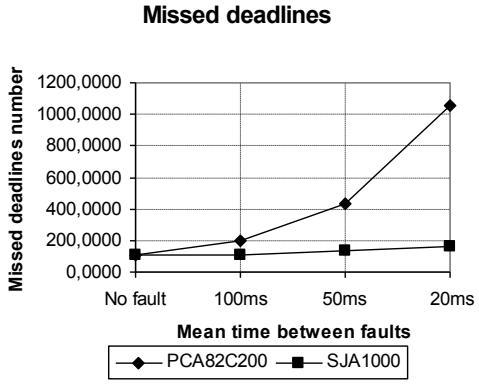

Figure 5: Missed *deadlines*

Figure 5 shows the number of missed *deadlines.* In the SJA1000 the number of correctly received messages that miss their *deadlines* is lower than in the PCA82C200.

Reception

First the behavior of the two CAN controllers in the worst case will be studied, the reception of 0 data bytes messages (without *stuff* bits) at the highest possible frequency. The transmission speed will be modified from 500 Kbps (1 bit transmission time = 2 μs) to 1 Mbps (1 bit transmission time = 1 μs). In Figure 6 the obtained *overrun* probability with the CAN controllers connected to the 8031 microcontroller at 16 Mhz can be seen. This probability is similar in both CAN controllers with only a little improvement in the SJA1000. This is due to the fact that the arrival rate exceeds the reception slot empty time.

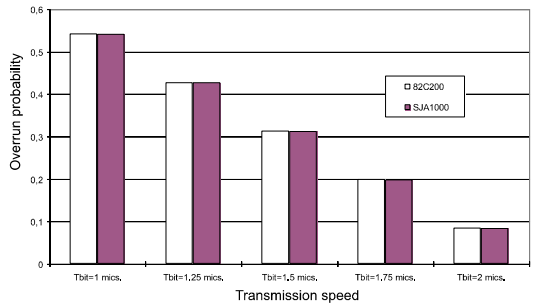

Figure 6: Bound of the worst performance

For the 251 microcontroller (at 16 Mhz) no *overrun* probability was obtained for both CAN controller at any transmission speed.

Then the previous models were modified to use a mix of messages of different sizes (from 0 to 8 data bytes) as workload. The messages were again sent at the maximum frequency. The results obtained with the 8031 microcontroller can be seen in Figure 7.

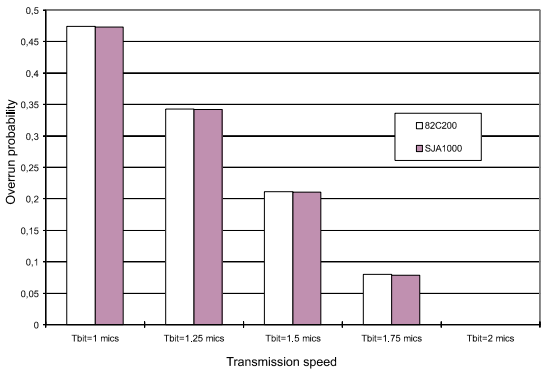

Figure 7: Overrun probability with a mix of messages.

With this workload only a very little improvement was obtained when the SJA1000 was used. Using the 251 microcontroller no difference could be appreciated since with both CAN controllers no *overrun* occurred.

Finally, the reception of burst of messages varying the time between two bursts, the number of messages in a burst and the size of the messages was modeled. In Figure 8 the results for the PCA82C200 connected to an 8031 (16 Mhz) receiving 1 data byte message bursts with a frequency of one burst each 5 ms. can be seen. The variation of the burst interval (10, 50, 100 ms) did not add any additional information to the study since the reception slot empty time is lower than the burst interval.

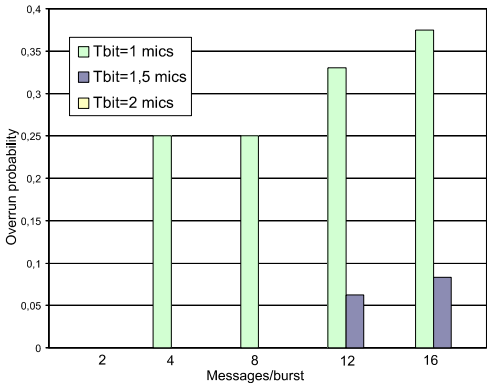

Figure 8: Overrun probability in the PCA82C200 under bursts of messages.

The probability of *overrun* increases as the number of messages in each burst increases. Similar results were obtained when the size of the messages was modified. With identical workload no *overrun* occurred with the SJA1000.

A better performance can be obtained for both CAN controllers employing the 251 microcontroller (16 Mhz) since there was no *overrun*.

Conclusions

In this paper, performance variation has been analysed depending on using the

PCA82C200 or the SJA1000 CAN controller in microcontroller systems. This study has attended to the circuit only, so it does not assume anything about the application level. In this way, it can be concluded that the new circuit provides, in terms of transmission, the possibility of preventing the increase of messages response time due to errors appearance, meaningful fact in terms of real-time systems.

On the other hand, in terms of reception, the addition in the new circuit of an enhanced buffer minimises significantly the number of lost messages due to overrun errors when burst of messages occurs. These bursts of messages are quite common in control applications. These conclusions are valid for the 8031, since in case of working with 251-like microcontrollers there are not significantly differences. This is due to the fact that in the 251 the time necessary to empty the reception slot is considerably lower than in the 8031.

References

[1] SJA1000. *Stand alone CAN Controller. Data sheet. Objective specification.* Philips Semiconductors. November 1996.

[2] J.Rufino, P. Veríssimo. *A study on the inaccessibility characteristics of the Controller Area Network.* Technical University of Lisboa. Proceedings of the 2nd International CAN Conference, 1995.

[3] A. Rubio, J.C. Campelo, R. Ors, J.J. Serrano. *Checkpointing in a CAN based distributed computer control system.* Pre-prints of the 14th IFAC Workshop on Distributed Computer control Systems, DCCS'97. Seoul, Korea, July 1997.

[4] M. H. McDougall. *Simulating Computer Systems. Techniques and tools*. MIT Press. 1987.

[5] QNAP2. *The Queuing Network Analysis Package. Reference Manual*. Simulog. 1996.

[6] PCA82C200. *Stand alone CAN Controller. Data sheet.* Philips Semiconductors. February 1992.

[7] K. Tindell, A. Burns. *Guaranteeing Message Latencies on Control Area Network (CAN).* Proceedings of 1st International CAN Conference. September 1994.

[8] K. Tindell, A. Burns, A.J. Wellings. *Calculating controller area network (CAN) message response times.* Control Engineering Practice, Vol. 3, nº8 pp.1163-1169, 1995.

[9] *Class C Application Requirements Considerations.* SAE Technical Report J2056/1. 1993.

[10] M.G. Rodd. *Real time and communication isues.* "Application of artificial intelligence in process control". Pages 255-273. Edited by L. Boullart, A. Krijgsman and R.A Vingerhoeds. Ed. Pergamon Press, 1992.

Company: Technical University of Valencia.
Dpt. of Computer Engineering
Address: Camino de Vera s/n
46022 - Valencia – SPAIN
Phone: +34 96 3877577
Fax: +34 96 3877579
Email: jcampelo@disca.upv.es
Homepage: http://www.disca.upv.es