RESEARCH PAPER | Engineering | Volume : 2 | Issue : 1 | October 2012 | ISSN - 2249-555X

Reliability in Automotive Embedded System using CAN

KEYWORDS: Reliability, CAN Controller

Balachandra Pattanaik
Sathyabama University,Research Schoolar,Electrical and electronics Engineering, Kumaraguru College of Technology, Anna University, Coimbatore

Dr S.Chandrasekaran

ABSTRACT *The reliability enhancement models for fault tolerant systems work is to propose automotive fault tolerant system models with static and dynamic redundancies based on the services required in the applications to enhance the overall system reliability without affecting system performance, correctness and safety. Embedded systems must meet increasingly high expectations of safety and high reliability Improvement by enhancing performance with system-level analysis. Modelling is needed not only for predictability and comparability when partitioning end-to-end functions at design time levels of reliability. CAN Controllers are used in automotive for fault tolerant embedded system. The existing reliability enhancement models are emphasizing various redundancy techniques both in hardware and software without focusing a formal way of recovery time minimization from the affected or degraded states in the automotive systems. The proposed model minimizes the number of errors that occur and hence the reliability of the system is enhanced. A component-based model fault tolerance with error detection has been developed. This model focuses on the automotive services to achieve a secure fault tolerant system.*

I .Introduction

It is proposed to model the automotive fault tolerant system as a combination of continuous time model and discrete event model. The system is modelled in to enhance the reliability using the redundant components at runtime. The fault tolerant system using a number of interacting CAN designed and the performance of such system is determined in terms of the number of errors detected, corrected and the damage caused by the various injected faults. The proposed model of a hybrid fault tolerant system is implemented using FT CAN with fault detection, damage assessment, error containment, fault recovery and the fault treatment features. As a consequence of its popularity and widespread use, most modern microcontroller families now have one or more members with on-chip hardware support for this protocol. This means, in turn, that FT CAN networks can now be implemented at very low cost. These embedded systems are task specific computing or controlling units. These systems are growing in number and complexity with addition of new functionality and features to modern automobiles. Designing and developing such automotive embedded systems requires a structured approach and a very well defined set of guidelines facilitating this process. The remaining nodes, upon receipt of this message, start local timers (each with different values), which upon expiry allow local tasks to be executed and messages to be transmitted in different timeslots on the network. However, this type of "domino" architecture lacks scalability, as the authors note that "a Flex CAN network for a safety-critical system always has to be characterized by a small number of nodes." If we are to develop reliable embedded systems using CAN, then we need to ensure that we can achieve reliable group communications. This means, for example, that when one node transmits a message, all nodes must receive the same message. One deficiency with CAN is that this condition may not always be satisfied, [1],[2]. most notably during the detection of end of frame (EOF) sequences. This problem can arise as follows. CAN receivers achieve consensus that the accepted message is valid by processing an error-free sequence of bits up to the sixth bit of the EOF sequence. At this point, the receiving CAN controllers accept the message. The sender, however, validates the transmission at the very last bit of the EOF. Potential problem [3]. If the subset of receivers detects an error in the sixth bit of the EOF sequence, they will subsequently reject the message and begin transmission of an error flag in the seventh bit of the EOF. The remaining receiver nodes will already have accepted the message; thus, an inconsistent delivery has arisen. Under normal circumstances, the sender will queue the message for retransmission; therefore, the possibility of inconsistent message duplicates (IMDs) or inconsistent message omissions (IMOs) arises. Previous studies have shown that the probability of this situation occurring in normal CAN is highly dependent on the bit rate, the nature of the bus traffic, and the number of nodes connected to the bus [4]. CAN, which stands for Controller Area Network, is the serial communication protocol internationally standardized by ISO. The automobile industry has hitherto witnessed the advent of various electronic control systems that have been developed in pursuit of safety, comfort, pollution prevention, and low cost. These control systems, however, presented a drawback in that since the communication data types, required reliability, etc. differed between each system, they were configured in multiple bus lines [5][6],

II. Reliability in Embedded systems

series configuration made up of M elements is considered (Taha H.A, 1982). The physical system is expected to operate at the maximum reliability over a time T and the cost of each element shall not exceed C. The system reliability Rs over a given period is given as:

$$R_s = \prod_{i=1}^{M} R_i \qquad (1.1)$$

where Ri is the reliability of the ith element. The problem of selecting maximum redundancy then becomes one of determining a vector n , with ni positive integers as given below: n = (n1, n2,… ni, ….nN) The problem then becomes,

Maximize :

$$R_s(N) = \prod_{i=1}^{M} R_i \qquad (1.2)$$

subjected to

$$\sum_{i=1}^{N} C_{ij}\, n_i \le C_j \qquad j = 1,2...., \qquad (1.3)$$

The problem considered has the following assumptions as each element has at least one redundant unit. The maximum number of redundant units that can be added to each element is known. The data for the reliability Rj (Kj) and the cost Cj (Kj) for the jth component (j = 1,2,3…M), given Kj parallel units are known. To enhance the reliability of the above system, the design requirement is to use one or two standby

Volume : 2 | Issue : 1 | October 2012 | ISSN - 2249-555X

units, which means that each main component may include up to three units in parallel. The proposed algorithm is used with the reliability and cost values for various redundancies as given in the Table 2.1.

Table 1.1 Reliability and cost values for various redundancies

Redundant Unit	j = 1		j = 2		j = 3	
Kj	R1	C1	R2	C2	R3	C3
1	0.6	1	0.7	3	0.5	2
2	0.8	2	0.8	5	0.7	4
3	0.9	3	0.9	6	0.9	5

III. TIME TRIGGERED COMMUNICATIONS

Although CAN was primarily intended to support event-triggered communications between unsynchronized nodes, time triggered communication—which has a number of benefits (see,for example, [12])—may be enforced, if due care is taken at the system design stage. A number of hardware- and software-based protocol extensions and modifications have been proposed to enable time-triggered communications on CAN. These tend to rely on the use of a global clock that, in turn, supports a time division multiple access (TDMA) message schedule. For example, Turski describes a distributed clock synchronization methodology with a potential resolution of bit time,1 using a combination of hardware and software. Pimentel and Fonseca describe a time-triggered system that, although it does not utilize a global clock, controls a cycle of communication via a synchronization message sent by a primary message producer with an accurate clock. The remaining nodes, upon receipt of this message, start local timers (each with different values), which upon expiry allow local tasks to be executed and messages to be transmitted in different timeslots on the network.

IV. RELIABLE GROUP COMMUNICATIONS

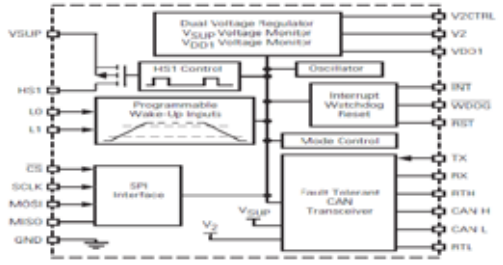
If we are to develop reliable embedded systems using CAN, then we need to ensure that we can achieve reliable group communications. This means, for example, that when one node transmits a message, all nodes must receive the same message. One deficiency with CAN is that this condition may not always be satisfied, most notably during the detection of end of frame (EOF) sequences. This problem can arise as follows. CAN receivers achieve consensus that the accepted message is valid by processing an error-free sequence of bits up to the sixth bit of the EOF sequence. At this point, the receiving CAN controllers accept the message. The sender, however, validates the transmission at the very last bit of the EOF. This presents a potential problem. If the subset of receivers detects an error in the sixth bit of the EOF sequence, they will subsequently reject the message and begin transmission of an error flag in the seventh bit of the EOF[7][8]. Previous studies have shown that the probability of this situation occurring in normal CAN is highly dependent on the bit rate, the nature of the bus traffic, and the number of nodes connected to the bus. The proposed software solutions have been adopted, in some cases, by the protocols described in the previous section; however, it should be noted that software solutions generally have bandwidth and processing overheads involved.

V.RELIABILITY ANALYSIS OF AUTOMOTIVE SYSTEM

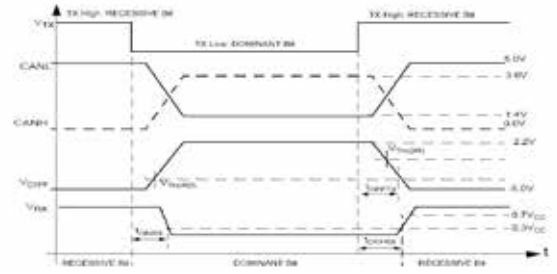
This was unrealistic in practical settings. In this work, a statistical model was developed to evaluate the effect of corrective and preventive maintenance schemes on car performance in the presence of system failure where the scheduling objective is to minimize schedule duration. In this, good bounds are available for the problem of minimizing schedule durations, or the make span. Graham provided the worst-case bound for the approximation algorithm, Longest Processing Time, and Coffman, Garey and Johnson provided an improved bound using the heuristic, multicity. By combining these, Lee and Massey were able to obtain an even tighter bound [9]. These studies, however, assumed the continuous availability of machines, which may not be justified in realistic applications where machines can become unavailable due to deterministic or random reasons. The device has four modes of operation, normal, stand-by, sleep and stop modes.

All modes are controlled by the SPI. An additional temporary mode called .normal request mode. is automatically accessed by the device (refer to state machine) after wake-up events. Special mode and configurations are possible for software application debug and flash memory programming. NORMAL MODE: In this mode both regulators are ON, and this corresponds to the normal application operation. All functions are available in this mode (watchdog, wake-up input reading through the SPI, HS1 activation, and CAN communication). The software watchdog is running and must be periodically cleared through the SPI. STANDBY MODE: Only the Regulator 1 is ON. Regulator 2 is turned OFF by disabling the V2CTRL pin. The CAN cell is not available, as powered from V2. Other functions are available: wake-

up input reading through the SPI and HS1 activation. The watchdog is running. SLEEP MODE: Regulators 1 and 2 are OFF. In this mode, the MCU is not powered. The device can be awakened internally by cyclic sense via the wake-up input pins and HS1 output, from the forced wake function, the CAN physical interface, and the SPI (CS pin). STOP MODE: Regulator 2 is turned OFF by disabling the V2CTRL pin. Regulator 1 is activated in a special low power mode which allows it to deliver 2.0 mA. The objective is to supply the MCU of the application while it is turned into a power saving condition (i.e stop or wait mode). Stop mode is entered through the SPI[10]. Stop mode is dedicated to powering the Microcontroller when it is in low power mode (stop, pseudo stop, wait etc.). In these modes, the MCU supply current is less than 1.0 mA. The MCU can restart its software application very quickly without the complete power up and reset sequence. When the application is in stop mode (both MCU and SBC), the application can wake-up from the SBC side (ex cyclic sense, forced wake-up, CAN message, wake-up inputs) or the MCU side (key wake-up etc.). When Stop mode is selected by the SPI, stop mode becomes active 20 μs after end of the SPI message. The 33889 Internal Block Diagram and the timing diagram being shown.



33889 Internal Block Diagram



0ns 100ns 200ns 300ns 400ns 500ns 600ns

Timing Diagram

Volume : 2 | Issue : 1 | October 2012 | ISSN - 2249-555X

VI. Conclusion:

In the combined hardware software fault tolerant systems, constrained and unconstrained reliability models are optimized genetically to minimize the cost and completion time. Automotive embedded systems make widely use of fault tolerance fault-detection and fault confinement techniques missing critical situation are reconstituted on the basis of other event and more generally, specification and implementation of several degraded functioning modes). Redundancy is used at the wheel angle but seldom at the ECU level because the criticality of the functions does not absolutely impose it. Some future functions, such as brake and accelerator, are likely to require active redundancy in order to comply with the acceptable risk levels and the design guidelines that could be issued by certification organisms. For critical functions that are distributed and replicated throughout the FT CAN, the system will play a central role by providing the services that will simplify the implementation of fault-tolerant applications. This drawback is overcome by a modular and structured design philosophy. The amount of reliability enhancement in hardware systems also depends on the quantitative specifications and the aging of the associated physical devices without which a system cannot be put to into operational condition in that environment assuming the control software is absolutely error free.

REFERENCES

Chopra, K., Kadekodi, G.K. and Murty, M.N. (1990), 'Peoples Participation and Common Property Resources', Economic and Political Weekly, Vol.24, No. 51, pp A189-A195. | Government of India.(GOI),(1961,1971,1981,1991, 2001): 'District Census Handbook', Census of India, Record Structure: Village Directory. | Government of Odisha.(GOO), (1965), Statistical Abstract of Odisha 1965, Directorate of Economics and Statistics, Bhubaneswar, Odisha. | Government of Odisha.(GOO), (2008), Statistical Abstract of Odisha 2008, Directorate of Economics and Statistics, Bhubaneswar, Odisha. | Government of Odisha.(GOO), (2011), Economic Survey 2011-12 and various other issues, Directorate of Economics and Statistics, Bhubaneswar, Odisha | Iyengar, S. (1989), 'Common Property Resources in Gujarat: Some Findings about their Size, Status and Use', Economic and Political Weekly (Review of Agriculture), Vol.24, No. 25, pp. A67-A77. | Jodha, N.S. (1986), 'Common Property Resources and Rural Poor in Dry Regions of India', Economic and Political Weekly, Vol. 21, No.27, pp. 1169-1181.