FAULT-MANAGEMENT ON SYSTEM'S LEVEL IN CONTROLLER AREA NETWORK BASED "SHARED-CLOCK" ENVIRONMENTS

Muhammad Amir*[1], Syed Waqar Shah[1], Michael J. Pont[2]

ABSTRACT

Shared-Clock (SC) architectures based on the Controller Area Network (CAN) protocol have been inherently plagued by flexibility and fault-management issues since they were introduced. The easiest way out of such issues is to adopt the more flexible/fault-manageable but expensive protocols such as the Time-Triggered Protocol (TTP) and FlexRay. Looking at the cost-effective nature of CAN and its widespread use, we started working on making such SC architectures more flexible and fault-manageable on embedded level as demonstrated by our previous work "Improving flexibility and fault-management in CAN-based "Shared-Clock" architectures", published in ELSEVIER Journal of Microprocessors and Microsystems (Volume 37, 2013, issue 1, pages 9-23). In this paper, we intend to show that the use of a Port Guardian (PG) mechanism can also improve fault-management on system's level.

KEYWORDS: *Controller Area Network (CAN); Fault-Management; Topology; Time-Triggered Systems*

INTRODUCTION

The Controller Area Network (CAN) protocol is extensively used in automotive, process control and industrial applications (Fredriksson, 1994), (Pazul, 1999), (Farsi & Barbosa, 2000) & (Etschberger, 2001). Due to such utility, most modern microcontroller families provide support for it (Philips, 1996), (Siemens, 1997), (Infineon, 2000) & (Philips, 2004). As mentioned earlier in (Amir *et al*., 2013) with several limiting issues. Of particular concern in this paper is the non-support of CAN for fault-management on system's level due to its original bus topology. Building on the CAN topology migration platform built in (Amir et al., 2013), in this paper, the same Port Guardian (PG) mechanism is used in order to demonstrate (with results) that fault-management on system's level in Shared- Clock (SC) architectures can also be achieved.

This paper is organized as follows: Section 2 presents the setup used for simulating the conditions in which certain system's level faults may occur. A model of such particular "Setup-related-faults" is also given in this section. In Section 3, we describe how system level faults are simulated inside a CAN-Star-based SC environment and present the system's response timings of managing such faults. Section 4 presents limitations of the proposed methodology and suggests further improvements. Finally, Section 5 presents our conclusions.

ENGAGED SETUP

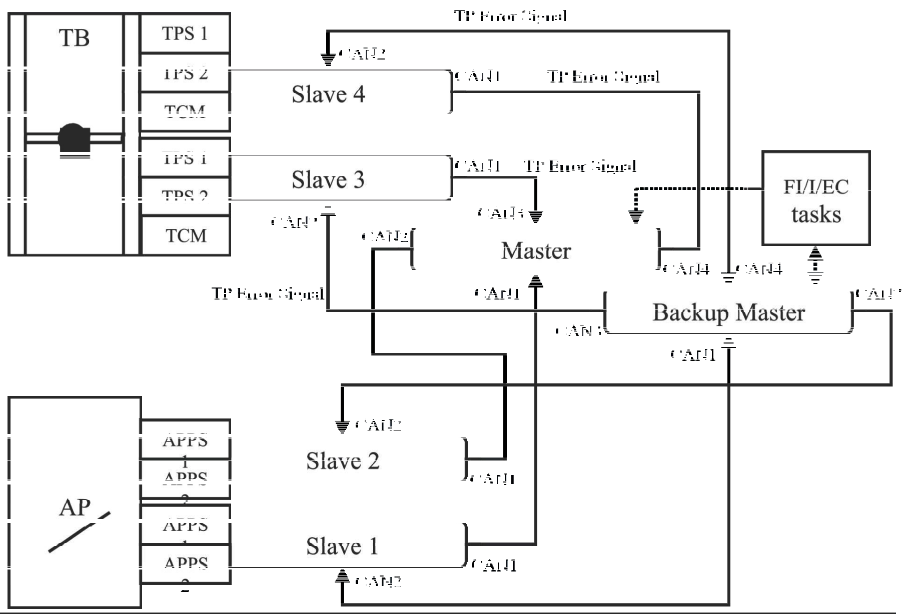
Electronic Throttle Control System (ETCS)

The setup used for the purpose of this paper is shown in Fig. 1 and is the same setup presented in (Amir et al., 2013). The ETCS shown in Fig. 1 is a hardware prototype assembled using Olimex development boards (Olimex, 2017). The Slaves in this setup, in electronic throttle control mode, run a single task (Slave 1and Slave 2 each run a sensing task which senses the accelerator pedal position while Slave 3 & 4 each run an actuation task which actuates the throttle control motor).

In Fig. 1, Slave 2 acts as a backup for Slave 1 while Slave 4 backups Slave 3. The Masters on the other hand each run three tasks (i.e. Fuel Injection/Ignition/Emissions control) while deploying the TTC-SC7 protocol (Amir et al., 2013) using the PG mechanism. Regarding scalability, the ETCS setup in Fig. 1 is capable enough to perform its main function and can also schedule other automotive-management tasks (e.g. Cruise Control, ABS Control, Traction Control etc) using the time-triggered co-operative task schedulers on the Master and Slave nodes. It is worth mentioning here that the methods proposed in this paper as well as in (Amir et al., 2013) are not just confined to ETCSs. The ETCS is used here as a system test case in order to show the ability of the proposed approach in improving fault-management in CAN-based safety-critical embedded designs. Other

1 Department of Electrical Engineering, University of Engineering and Technology, Peshawar, Pakistan.*

2 Safety Systems TM Ltd, Registered Office 15 Nether End, Great Dalby, LE14 2EY, UK.



TP – Throttle Position
TB – Throttle Body
TCM – Throttle Control Motor
TPS – Throttle Position Sensor
AP – Accelerator Pedal
APPS – Accelerator Pedal Position Sensor
FI/I/EC – Fuel Injection/Ignition/Emissions control
(LPC2378-STK development boards were used as Slaves due to a support for ADCs for sensing and Digital to Analog converters (DACs) for exciting the TCM)

Figure 1: Electronic Throttle Control System (ETCS) schematic.

safety-critical applications of the proposed approach where embedded or system level faults may occur include: Industrial Process Control; Static and Portable Traffic Lights Control; Level Crossing Signal Control; Air Traffic Control; Weapons Systems (e.g. Surface to Air Missile Batteries); Intra-Satellite Networking; Automated Motor Vehicles (e.g. Mars Rover or Bomb Disposal Robots); Medical Monitoring Equipment; Locomotive Braking Systems; Avionics; Fire Alarm Systems; Intruder Alarm Systems (Security Systems) and Industrial Robotics. The proposed design in this paper is based on the TT architecture and as a well established fact; systems based on such architectures are highly predictable and reliable (Kopetz, 2000) and (Short & Pont, 2007). Also, such features are an essential requirement in safety-critical applications (Kopetz, 2000).

System's Level Faults in the ETCS Setup

Faults related to the setup used for this case study are described in this section as follows.

Sensor Failure or Malfunction

In ETCS based automobiles, sensors are deployed at both ends i.e. for sensing the position of the accelerator pedal as well as for sensing throttle position. A malfunction at any end will provide erroneous data to the onboard computer and will degrade the efficiency and safety of the driver's environment. A sensor failure at either end will also have the same effect on the integrity of the environment.

Throttle Control Motor (TCM) Failure or Malfunction

A TCM failure or malfunction will not replicate the proportional mechanical movement of throttle as dictated through the data sent from the onboard computer. In turn, for example, the TCM will not bring the speed of the vehicle down (at this time the driver is removing or has removed his/her foot from the accelerator pedal) as it might have failed or malfunctioned at a full-open-throttle position. This fault will put the safety of the driver in jeopardy as the vehicle will speed up instead of slowing down.

Stuck Throttle Fault

This fault might be a consequence of TCM failure/ malfunction or some other mechanical anomaly like Gunk accumulation in throttle body. A stuck throttle fault will render the throttle movement stuck at one place in relation to the accelerator pedal position sensor's data sent by the onboard computer.

Stuck Accelerator Pedal (SAP) Fault

There are different reasons such as mechanical or design based anomalies which may cause an accelerator pedal to get stuck at any position. An SAP fault at a maximum position is most worrying as it creates hysteria for the driver which will cause further mistakes e.g. turning the ignition OFF which causes the power steering to shutdown. Consequently, it makes it pretty difficult to bring the vehicle to a safe stop.

FAULT SIMULATION METHODS & RESULTS

System's level faults described in previous section were simulated through an ETCS (shown in Fig. 1) designed by using LPC-E2294 rev.B development boards (Olimex, 2017) as Masters and LPC2378-STK development boards (Olimex, 2017) as Slaves. Fault simulation methods are described in Fig. 2. In Table 1, SWT represents *"System Wait Time"*; *"Fault Duration"* is represented by FD while the *"Response Time"* of the PG mechanism is represented by RT as were in (Amir et al., 2013).

As was for embedded level faults in (Amir et al., 2013), the PG mechanism (given as reference in Appendix) checks *"Fault Intermittency"* for certain system's level faults as well using the SWT as shown in Table 1. Which means that if the fault duration is greater than the system wait time? Only then the PG mechanism will deploy its redundancy routine by switching to a backup node (see Appendix). Otherwise, it will keep on running the setup in its original configuration as shown in Fig. 1. Original configuration here means the state of the setup before fault-injection.

Table 1: PG mechanism's response timings (RTs) related to faults on system's level.

Fault		SWT – FD	PG Response	RT
Sensor failure or mal-function	Slave 1 transmits an empty Ack message	3s – 6s	Slave 2 Engaged	1.1084 ms
		3s – 2s	Slave 1 Continued	-
	Slave 3 transmits an empty Ack message	3s – 6s	Slave 4 Engaged	1.1086 ms
		3s – 2s	Slave 3 Continued	-
	Slave 2 transmits an empty Ack message	3s – 6s	Delete Task Array	14.45 µs
		3s – 2s	Slave 2 Continued	-
	Slave 4 transmits an empty Ack message	3s – 6s	Delete Task Array	14.45 µs
		3s – 2s	Slave 4 Continued	-
TCM failure or mal-function	Slave 3's APF (Returns Error)	3s – 6s	Slave 4 Engaged	1.1083 ms
		3s – 2s	Slave 3 Continued	-
	Slave 4's APF (Returns Error)	3s – 6s	Delete Task Array	14.46 µs
		3s – 2s	Slave 4 Continued	-
Stuck throttle fault		RT approximately same as TCM failure or malfunction		
Stuck Accelerator Pedal (SAP) fault	APPSs value max + EINT0 Occurrence		Delete Task Array	3.32 µs
	APPSs value max + EINT1 Occurrence		Delete Task Array	3.322 µs
Stuck Accelerator Pedal (SAP) fault	APPSs value (any non-zero position) + EINT0 Occurrence		Delete Task Array	3.31 µs
	APPSs value (any non-zero position) + EINT1 Occurrence		Delete Task Array	3.32 µs

1. Sensor failure or malfunction: A conditional loop was kept inside the scheduler Update function of a particular Slave (i.e. Slave 1, Slave 2, Slave 3 or Slave 4) which caused the Slave's scheduler to send empty Acknowledgements (Ack messages) to the Master for the fault duration (i.e. loop duration). In the current simulations, the conditional loop took effect 30 s after system power up. An approximate response time was observed on the oscilloscope when the ADC's on the Slaves were deliberately mismatched.

2. TCM failure or malfunction: For this simulation, the values of the ADC's on Slave 1 and Slave 3 were kept the same. A conditional loop was kept inside the APF for Slave 3 on the Master node which subtracted a constant value from the matched ADC value sent by Slave 3 thus causing a mismatch between the value sent by Slave 1 and the value sent by Slave 3. Consequently, the APF for Slave 3 on the Master node returned an "ERROR" for the duration of the fault (loop duration). The same procedure was adopted for Slave 1 and Slave 4.

Figure 2: Descriptions of system's level faults simulations.

The fault-management response times for the engaged setup shown in Fig. 1 and the PG mechanism (Appendix) towards system level faults are given in Table 1. Where, APF means *"Acknowledgement Processing Function"* inside the Master scheduler. Please note that EINT0 and EINT1 in Table 1 are two redundant external interrupt lines connected to the brake of the vehicle. These lines in turn simulate a Brake Override Mechanism (BOM)). The BOM works in an event if the accelerator pedal is stuck at the maximum depressed position (or any non-zero position) and the driver is trying to brake.

It is our understanding that in normal driving practice drivers does not press the accelerator and brake pedals at the same time. In such an event the Master gives priority to the brake and deletes the fuel injection task from the task array. This action reduces the engine torque and the driver will be able to bring the vehicle to a controlled stop through the brake. The reason for this action is that we know; braking against maximum torque of the engine is difficult and dangerous (dangerous in a sense that it may burn-out the brakes and the driver will lose all safe-vehicle-stopping- control). Simulation methods that were used for such faults are described in Fig. 2 above.

LIMITATION & FURTHER WORK

The only obvious limitation of the proposed methodology as was described in (Amir et al., 2013) is the hardware limit of the Master that only houses four CAN interfaces in order to interact with the Slaves. For removing this limitation, further work will be carried out in the near future to come up with an FPGA (Xilinx, 2017) based CAN-Star design with enhanced CAN interface selectivity needed for a range of other demanding applications.

CONCLUSIONS

The main concern of our research is safety; as we all put our lives on a daily bases in the hands of embedded systems that are invisible to us. We expect such systems to be flexible towards fault-management and behave in a safe manner. Here in this paper, we have presented System's Response Timings to a certain set of simulated System's Level Faults using a proposed methodology. Finally, it can be concluded that besides managing faults on embedded level, the proposed star topology with a SC protocol-based Port Guardian (PG) mechanism can help towards managing faults on System's level as well.

ACNOWLEDGEMENT

We would like to thank the research community for striving towards enhancing the safety-critical features of embedded systems and treating them of having paramount importance.

REFERENCES

1. *Amir, M., & Pont, M. J., (2013), "Improving flexibility and fault-management in CAN-based "Shared-Clock" architectures", Journal of Microprocessors and Microsystems, Vol. 1, Issue 37, pp. 9-23.*

2. *Etschberger, K., (2001), "Controller Area Network: Basics, Protocols, Chips and Applications", IXXAT Automation GmbH.*

3. *Farsi, M., & Barbosa, M., (2000), "CANopen Implementation: Applications to Industrial*

Networks", U.K.: Research Studies Press Ltd.

4. Fredriksson, L. B., (1994), "Controller area networks and the protocol CAN for machine control systems", Mechatronics, Vol. 4, Issue. 2, pp. 159-192.

5. http://www.olimex.com/Products, accessed on 29/12/2017 at 2200PST.

6. http://www.xilinx.com, accessed on 29/12/2017 at 2200PST.

7. Infineon., (2000), "C167CR Derivatives 16-Bit Single-Chip Microcontroller", Infineon Technologies.

8. Kopetz, H., (2000), "A comparison of CAN and TTP", Annual Control Review, Vol. 24, pp. 177–188.

9. Pazul, K., (1999), "Controller Area Network (CAN) Basics", Microchip Technology Inc, Preliminary DS00713A, Page 1 AN713.

10. Philips., (1996), "P8×592 8-bit Microcontroller with on-Chip CAN Datasheet", Philips Semiconductor.

11. Philips., (2004), "LPC2119/2129/2194/2292/2294 Microcontroller User Manual", Philips Semiconductor.

12. Short, M. J., & Pont, M. J., (2007), "Fault-Tolerant Time-Triggered Communication Using CAN", IEEE transactions on Industrial Informatics, Vol. 3, Issue 2, pp. 131-142.

13. Siemens., (1997), "C515C 8-bit CMOS Microcontroller User's Manual Siemens".