Controller Area Network

Abdullah Zafar
Industrial Communication
Hochschule Hamm-Lippstadt
Hamm, Germany
abdullah.zafar@stud.hshl.de

Mustafa Touqir
Industrial Communication
Hochschule Hamm-Lippstadt
Hamm, Germany
mustafa.touqir@stud.hshl.de

Syed Muhammad Saim
Industrial Communication
Hochschule Hamm-Lippstadt
Hamm, Germany
syed-muhammad.saim@stud.hshl.de

***Abstract*—Controller Area Network (CAN) has proved to be a significant winner if we talk about communication protocols in automotive industry, since its invention in 1980.This paper initially discusses the basic aspects related to CAN bus networks. Transmission and Reception of CAN messages are also included in the paper. CAN bus architecture distinguishes CAN from other communication protocols, reference architecture of CAN with its 7 different layers of CAN bus eliminates excessive and intensive wires from the network providing simple and faster communication, which is discussed in the message broadcasting and data transfer.The fault tolerance and error handling in this paper can be applied to automotive fault troubleshooting, analysis, and maintenance. Worldwide acceptance of CAN and its affordable cost has encouraged the manufacturers to use this technology in their products.**

I. Introduction

Controller Area Network (CAN) is a serial bus networking technology designed by Robert Bosch GmbH in Germany in 1983. CAN technology was originally designed for the automotive industry, but after its success in the automation industry, it was then used in other applications as well. CAN is a two-wire, half-duplex, high-speed network system that is far superior to other conventional serial technologies such as RS232 regarding functionality and reliability. CAN networks are also used in micro controllers as an embedded communication or open communication system for Internet of Things (IoT) devices.

There is a range of micro controllers from small 8 bit up-to high-end 32-bit micro controllers. Vehicles nowadays use more micro controllers than before. Each micro controller system is referred to as an electronic control unit (ECU) which includes the engine management ECU, an anti-braking system (ABS) ECU, a dashboard ECU, active suspension, and the radio/CD player, for example. Each of these ECUs manages its control strategies, but they all need to access information relevant to their operation, for example, drawn from engine speed, throttle position, brake pedal position, and engine temperature.

CAN network technology was initially designed for European automotive industry communication standards, which later became a popular bus communication throughout the world. The focus when designing CAN bus network was to design a communication system field bus based on serial communication which reduces wiring and enables faster communication between ECUs in vehicles. Consider an example, where TCP/IP is designed for the transport of large data amounts, CAN is designed for real-time requirements and with its 1 MBit/sec baud rate can easily beat a 100 MBit/sec TCP/IP connection when it comes to short reaction times, timely error detection, quick error recovery and error repair [5].

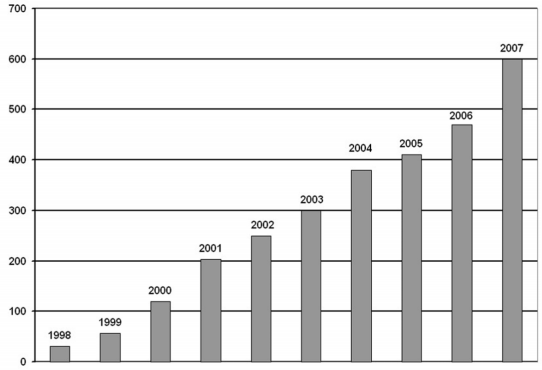

Fig. 1. Number of Million CAN nodes per User. [5]

CAN has no distinct application, its widely used widely from automotive, space and aviation to household appliances. Main application of CAN bus is suited where the elimination of expensive and intensive wiring replaces a simple architecture with better performance of distributed multi-processor system. Some of the primary applications for CAN are in the domain of HTV and LTV, Maritime Electronics, Industrial automation, Lifts and Escalators, Medical equipment and many more. Fig 1 illustrate the increase increase in number of nodes per users with time.

II. Architecture

From the hardware implementation of the Medium Access Control (MAC), down to the application-level programming interface (API) and Logical Link Control (LLC) layers inside the CAN adapter, A CAN communication system craves the execution of a complex protocol stack. The design choices rely on the various attributes of communication at all levels such

as, the structure of the middle layers and operating system, the I/O management scheme inside the device driver, the hardware architecture of the peripheral adapter. The OSEK COM automotive standard [2] defines a depiction of the CAN communication architecture as shown in Figure 2. The paramount components of the architecture are:

- The CAN Controller is the (hardware) component on which the physical access to the transmission medium is dependent.
- The CAN Device Driver, the software component, on which various tasks related to the low-level transmission of messages are dependent.
- The transport layer comes up with processes for controlling the flow and sending acknowledgements resulting in the assistance to establish virtual channels and bring forth an widened addressing venture.
- The Interaction layer brings an API to the application for reading and writing data according to application-defined types.
- The Diagnostic and Network Management layers decide and observe the network configuration and provide information about the network nodes that are active.
- The Calibration layer underpins the CAN Calibration Protocol (CCP) which is a speedy interface based on CAN to measure and calibrate systems. [3]

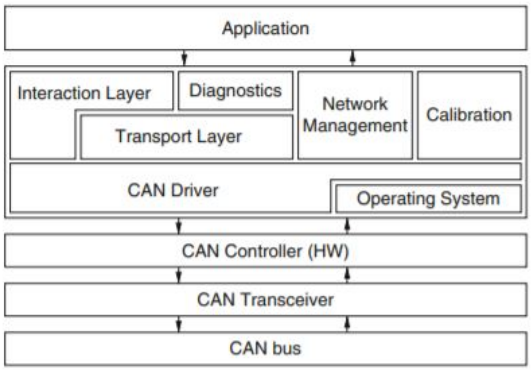

Fig. 2. Typical architecture of a CAN-based system. [3]

III. CAN BUS IMPLEMENTATION

n this section, three automobile examples of distributed control architectures based on CAN are explained. The architectures are applied in a spacecraft, a truck, a boat, and a passenger car.

A. VOLVO Passenger Car

Volvo XC90 consists of two CAN buses in its distributed control architecture as shown in Fig 3. In the diagram, the network on the extreme left is a CAN for power train and chassis subsystems. It has a communication rate of 500 kbps and connects engine and brake control (TCM, ECM, BCM, etc.) as an example. The body electronics such as door and climate control(DDM, PDM, CCM, etc.) are connected by the other CAN and has a communication rate of 125 kbps. Eventually, central electronic module (CEM) is an ECU is acting as a gateway between the two CAN buses.

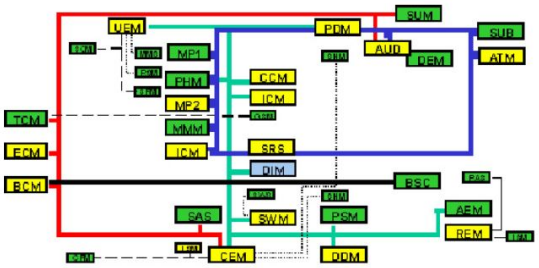

Fig. 3. Distributed control architecture for the Volvo XC90. [4]

B. Scania Truck

Scania truck consists of three CAN buses in its control architecture, denoted green, yellow, and red according to their relative priority. This is depicted in Fig 4. On the extreme left, the CAN contains less evaluative ECUs such as the climate control and the audio system. The middle CAN steers the communication for important subsystems is steered by the middle CAN that are indirectly included in the engine and brake management. Eventually, the most critical CAN bus is located on the extreme right that connects all ECUs for the driveline subsystems. A gateway between the three CAN buses is established via the coordinator system ECU (COO). Connected to the leftmost CAN is a diagnostic bus is connected to the CAN on the extreme left, which is utilized to gather information on the status of the ECUs. Hence, the diagnostic bus can be utilized for error detection and debugging.

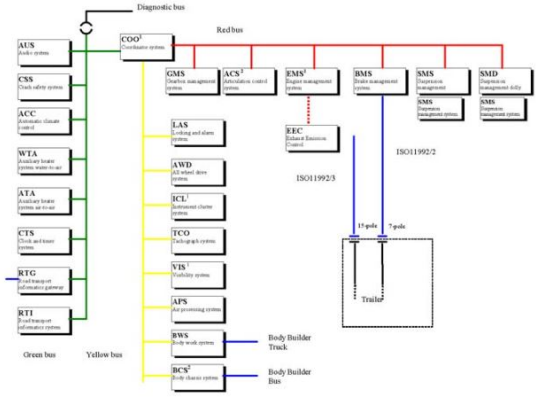

Fig. 4. Distributed control architecture for a Scania truck. [4]

C. SMART-1 spacecraft

The distributed computer architecture of SMART-1 is partly depicted in Fig. 10. The block diagram exemplifies the fragmenting of the system into two parts: one subsystem allocate to the scientific experiments and another for SMART-1 control system. There is one system CAN and one payload CAN as each of them is using a separate CAN. In the middle of the figure, the spacecraft control is executed by the redundant controllers CON-A and CON-B. Most control loops are closed over the system CAN with CAN nodes that provides sensing and actuation capabilities. On the

system bus, CAN nodes include not only the spacecraft controller, but also nodes for telemetry and telecommand (earth communication), hydrazine thruster, sensors and sun star tracker, reaction wheels and reaction gyro, thermal control, electronic propulsion and orientation, and power control and distributions.

Variuos services have been taken to corroborate system robustness and to model all nodes redundant; some in an active, others in a passive fashion. One nominal and one redundant communication path are included in each bus CAN. For error detection, redundancy management and recovery, a strategy and a hierarchy have been defined. The spacecraft controller can occupy the verdict to switch over to the redundant bus. Most other nodes will inspect for the life sign message from the spacecraft controller. The nodes will attempt to switch to the other bus, if the life sign is not available. The vigorous unit has greatest authority in the autonomy hierarchy and will transfer to the redundant spacecraft controller if the primary controller is considered to have failed. In addition, the activation and deactivation of the other nodes are controlled by the power unit.

The ground can intervene manually as a last means for recovery. Another crucial occupation for SMART-1 is radiation tolerance and the detection of radiation-induced errors instigating an impossibility to use ordinary CAN controllers. Therefore, the license was acquired from Bosch and was implemented on a CAN controller in a radiation-tolerant field programmable gate array (FPGA) using VHDL ocde. Specific error detection and error handling mechanisms are some other features that were added to the CAN protocol simultaneously. [4]

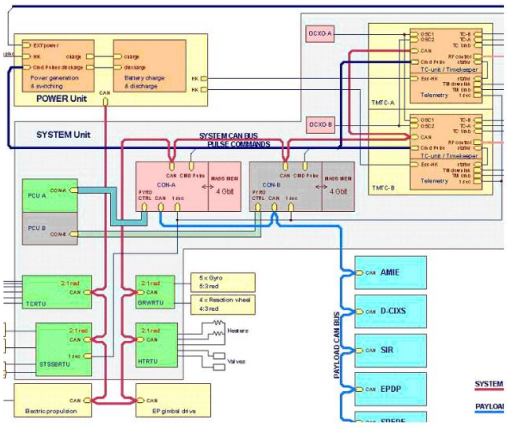

Fig.5.Control architecture for SMART-1 spacecraft. [4]

IV. Disadvantages associated to CAN bus

Recently, the amount of electronic devices in automobiles has staggeringly escalated. This pertains to the number of actuators, electronic control units (ECU) and sensors together with the quantity of electronic devices for navigation and entertainment systems. Bus systems are utilized to expeditiously deal with the enormous quantity of data. Therefore, an understanding must be developed between technological deliberation on the one hand and economical ostent on the other hand. For example, resulting from the data rate and the expected safety principle. An extensive overview discussing the varying bus principle in the automotive field can be exposed in.

For chassis control systems and power train communication the event triggered CAN bus has built up itself as a de facto standard. Since futuristic control concepts, such as X-by-wire demand hugely dependable architectures, just a while ago the demand for time triggered communication systems has exacerbated. For specific applications time triggered principles are envisioned to be superior compared to event triggered principles, since their behavior is quasi stochastic during usual operation. Usually, time slices determine the permission to access the bus (time division multiple access, TDMA) such that the timeliness of all messages can be guaranteed. An other very interesting feature in automotive field is composability. Since the time slices to access the bus are fixed, the functioning along the time axis is decoupled from the actual bus load.

In fact, the fixed stages among the messages are continuous. Therefore, it is viable to develop dissimilar subsystems separately (e.g. by the car manufacturers and suppliers) and subsequently to integrate them into the entire system. There are of course also some drawbacks in contrast with event triggered systems. For example, event triggered systems have a greater real-time performance when reacting to asynchronous external events which are not known in advance. Another edge is their greater flexibility. Hence, some busses try to integrate the advantages of both principles (event and time triggered) as for instance TTCAN or FlexRay. It is not within the scope of this paper to rigorously compare the different characteristics of event and time triggered systems. Those works which address such a comparison are, for example, as already mentioned the CAN bus became very marketable in automotive applications. To aid ultimatum for time triggered architectures, an extension of the CAN bus protocol to TTCAN (Time Triggered CAN) has been designated by the International Standardization Organization in ISO 11898-4. Meanwhile, also silicon implementations are available, e.g. from Bosch or Infineon [1]

V. Message broadcasting and data transfer

A. Transmission

For transmission of a CAN message, the device driver first needs to verify if the transmitter is in the off-line or not. A function exists in the device driver software that requests the transmission of a CAN message. The next step involves the driver ensuring the availability of a TxObject in the controller. In case, there is a unavailability of TxObjects, the message gets stored temporarily in the software transmit queue. When there is a high-priority message in the queue,

some drivers have the ability of evicting one of the messages from the adapter TxObjects and hence making it available for the newly arrived transmission request. Upon acceptance of a message by a TxObject, an acknowledgement code is sent that signals that the transmit request has been accepted by the driver. [3]

In case the transmission message gets rejected for any reason, the application is responsible for notifying and re-trying of the transmission request. The transmission if enabled on the CAN controller as soon as the message content is replicated or copied into the adapter transmit buffer. The entry of an interrupt signal basically confirms the successful completion of the transmission. A polling based management can also be implemented but it is highly discouraged due the fact that it is not suitable in terms of time predictability. [3]

After a successful transmission has been performed, the interrupt service routine gets activated and set of actions are performed. Firstly, the user gets the option of selecting a confirmation mechanism, this can be a confirmation flag or confirmation function. There is a possibility of deploying both as well. In case they are used by the user, the confirmation flag is set and the confirmation function is called. Incase the CAN driver is set up to utilize the transmit queue, it checks whether or no the queue is emp ty or has a pending request that needs to be accommodated. Transmit interrupt routine gets terminated incase the queue in empty.

Queued messages are handled by removing the message, copying it into the available TxObject and enabling the transmission of the message. Selection of the message from the queue is priority based, the message with higher priority is given preference over messages with lower priority. Many drivers use FIFO queue, but the disadvantages associated towards it include multiple priority inversions, and a very difficult time predictability. [3]

B. Reception

Reception of CAN messages occur asynchronously. In the upper software layers, messages reception occurs without any explicit service function call, its done by using a notification function or asynchronously, via mailbox. The CAN driver is responsible for the reception of the message, it involves copying the contents from the RxObject buffer register into the memory area. These desired tasks are performed by the CAN driver when it has been informed about the reception by the CAN controller. There is a possibility for the driver to check the content of the RxObjects periodically using the polling strategy or the controller can simply send an interrupt signal. [3]

After the receive interrupt signal has arrived and the handler routine has been executed, it means that the CAN message has passed the masking and filtering acceptance scheme which is defined in the adapter hardware, and the receive register stored its contents. However, in a CAN controller, this information is not enough to ensure that the message is to be processed by the node. It might be possible that the node is just acting as gateway for the message and needs to process it forward in the shortest possible time. For this purpose, many drivers have the option of defining a receiver callback routine, which is to be called after the hardware acceptance stage.In current scenario(automotive industry), message input is usually polling-based rather than interrupt based, message loss by over writing is tackled by reserving the peripheral objects or buffers to the input stream. [3]

C. Bus-Off and Sleep Modes

CAN driver is responsible for notifying the application if a bus-off state is present, this is done by calling a special call-back function. Some CAN controller supports sleep mode however, the application can restore the functionality of the node in bus-off state by re-initializing the CAN controller using a suitable driver function. Sleep mode can help in reducing the power consumption but the driver in these scenarios provide a service function that facilitates in entering and leaving the sleep mode on request. In case, the CAN controller is automatically awakened by the CAN bus, a special callback function is made to the application that informs it regarding the node wake up. [3]

VI. ERROR MANAGEMENT IN CAN BUS

The CAN protocol provides a robust mechanism that is fault tolerant. It is engineered in a way that can perform error detection and each node has the ability to monitor the broadcast transmissions over the bus, regardless of it receiving or sending data. Corrupt messages are flagged by any node detecting an error, these messages are later on re transmitted automatically. There are a series of error detection techniques that CAN protocol deploy, which contributes towards its fault resistance and high level reliability.The following subsection briefly describes the techniques used by CAN protocol for error management. [5]

A. BIT MONITORING

CAN nodes have the ability to monitor the bus after the messages are sent, in case the bit level in the bus differs from the bit sent, this is regarded as bit error. The dominance of bit monitoring lies in the fact that it allows global error detection, aswell as it enables us to find the location of the error source. [5]

B. BIT STUFFING ERROR

A bit stuffing error is detected when there is a sequence of bits at the same level in a data, within a message field that is subject to bit stuffing. It must be established that Bit stuffing error only occurs between state of frame bit and the cyclic redundancy code sequence, this is due to the fact bit stuffing

is only applied in these specific fields. Figure 6.0 gives a better representation of the but stuffing range. [5]

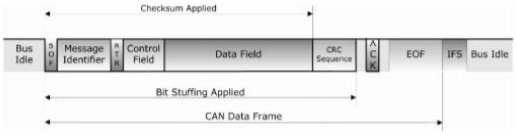

Fig. 6. Number of Million CAN nodes per User. [4]

CRC delimiter, ACK delimiter, End of Frame Field and Intermission Field, are considered static fields due to the fact that their data level is static. These components are also responsible for checking the consistency of a data or remote frame. Presence of one or more dominant bits in either of these components is regarded as frame error. [5]

C. ACKNOWLEDGEMENT ERROR

The receivers acknowledges the successful reception of a message by changing the content of the ACK field to dominant level. Acknowledgement error occurs when the transmitting node does not detect a dominant bit in the ACK field.

VII. BENEFITS OF USING CAN BUS

In early times, the only advantage of bus system was that it reduced wiring and had a little effect on reducing the cost. CAN BUS however has a lot of advantages, few of them include, reliability, improved service, maintenance features, speed, error management, low cost implementation and worldwide acceptance.

The inclusion of big players in the competitive semi-conductor market, including Intel, Philips and Infineon, selling CAN chips have resulted into low chip prices. Deploying CAN into an application is lower in cost as compared to TCP/IP or RS232/485. CAN software libraries uses lower memory footprints and requires less CPU performance as compared to TCP/IP implementation. [5]

CAN BUS has the capability of functioning in extreme electrical environments and a high degree of real time. Real time abilities are possible due to presence of short arbitration times, limited data length and extremely short error recovery times.

The CAN technology is widely accepted around the globe and due to a large use of it in automotive industry guarantees a low price and long-term availability. The wide-spread acceptance of CAN dates back in 1987 where CAN was being used in European cars and trucks.

VIII. CAN BUS ARBITRATION

Similar to other serial communication systems, CAN is based on a two-wired connection between nodes in a network. Collision occurs when there is an instance where two or mode nodes are trying to access the bus at the same point in time. Collision leads to delays and damaging the message. CAN avoid message collision by utilizing the message ID of the nodes. The message with higher priority can access the bus before the messages with lower priority, subsequently, message with lower priority switch to listening mode and wait for their turn. CAN provides a non-destructive bus arbitration as it allows the messages with low-priority that lose their arbitration to start a new arbitration as soon as the bus is available. [5]

Unique message identifier(11/29 bit) are present in CAN data transmission, they assist in representing the priority of the message as well. A lower message ID represents a higher priority. High Priority messages can access the bus in shortest time even if the bus is high caused by a message with a low priority. [5]

Figure 7, demonstrates an example where three nodes are present in a four node CAN network and they try to access the bus at the same instance. The example shows that node C wins the access to the bus within 12 clock times.

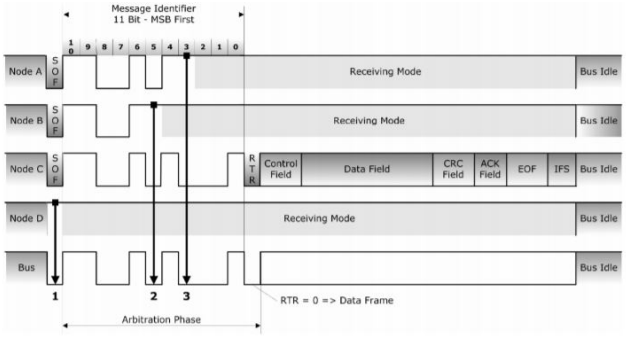

Fig. 7. CAN BUS ARBITRATION. [5]

IX. CONCLUSION

The fact that CAN bus is gaining popularity in industries other than automotive shows on how important this technology has been. Implementing CAN bus technology makes the system easy to use, especially for testing at high speed.As discussed in the paper,CAN bus being an affordable entity is one of the reasons for its wide use and popularity. Real-time, reliability and flexibility, these characteristics make CAN bus a crucial network communication technology used in the automotive network communication industry. However, there are some venues that are still to be tapped and further research must be conducted on topics like: migration to higher level protocols in CAN, Time Triggered CAN (TTCAN) and CANOpen. Developments in bus network length, which is limited to 40 meters, for its use in other industries such as manufacturing industries. If new technologies are not able to compete with the price over performance ratio of CAN and its higher-level protocols, they will not be able to manage to replace Controller Area Network as a preference field bus technology.

REFERENCES

[1] A Albert, R Strasser, and A Trachtler. Migration from can to ttcan for a distributed control system. In *Proceedings of 9th international CAN in Automation Conference*, volume 5, pages 9–16, 2003.

[2] Jean-Luc Bechennec, Mikael Briday, Sébastien Faucou, and Yvon Trinquet. Trampoline an open source implementation of the osek/vdx rtos specification. In *2006 IEEE Conference on Emerging Technologies and Factory Automation*, pages 62–69. IEEE, 2006.

[3] Marco Di Natale, Haibo Zeng, Paolo Giusto, and Arkadeb Ghosal. *Understanding and using the controller area network communication protocol: theory and practice*. Springer Science & Business Media, 2012.

[4] Karl Henrik Johansson, Martin Törngren, and Lars Nielsen. Vehicle applications of controller area network. In *Handbook of networked and embedded control systems*, pages 741–765. Springer, 2005.

[5] Wilfried Voss. *A comprehensible guide to controller area network*. Copperhill Media, 2008.