Interfacing CAN Bus With PIC32 Microcontroller For Embedded Networking

Umesh Goyal[1], Dr. Neelam Rup Prakash[2]
[1] M.E. Student, E& Ec Department, PEC University of Technology, Chandigarh
[2]Supervisor and Head, E& Ec Department, PEC University of Technology, Chandigarh

ABSTRACT
In this paper, the design of Controller Area Network (CAN) bus interfacing with PIC32MX795F512L is presented. This paper also gives an overview of Controller Area Network bus, describing its data format, signaling format and some other aspects. Controller Area Network is a multi-master, message broadcast system that can operate at maximum signaling rate of 1 Mbps (Megabits per second). CAN bus is a two wire bus used for serial communication. The various advantages of CAN bus are discussed over other serial buses like MOD bus. The various designing components used in interfacing CAN bus with PIC32 bit micro-controller is described with the final design schematic of CAN bus with PIC32MX795F512L.

***Keywords* -** Controller Area Network, Microcontroller, Orcad, PIC32MX795F512L,

I. INTRODUCTION
A Complex automated industrial system usually needs an organized hierarchy of a number of controller modules for its proper functioning. In this hierarchy usually a user sits at top to monitor or operate the system. This top level is linked to the middle layer of Programmable Logic Controllers via a communication system like Ethernet. At the bottom layer , this Programmable Logic Controllers linked via Field buses to that components of the system that actually perform the tasks like actuators, sensors etc. Field Bus is a way to connect equipment in real time distributed control system. Field Buses are the serial buses for serial data communication. Earlier, the computers are connected using RS-232 by which only two devices could communicate at a time while field bus allows multiple devices to communicate with one communication point at controller level. Controller Area Network is an important standard in field bus system.

II. CONTROLLER AREA NETWORK
The CAN Bus was developed by Robert Bosch, a German Automotive System supplier, in mid-1980's for automotive applications in automobile systems. CAN is an International Standardization Organization (ISO) defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The CAN bus has an ability to self-diagnose and automatically repair errors present on the bus. Due to these features, CAN bus is popular in a variety of industries including medical, embedded systems and manufacturing. ISO 11898 is the CAN communications protocol that describes how data is passed between different devices on a network and conforms to the Open Systems Interconnection (OSI) model. The physical layer of OSI model defines the actual communication between devices connected by the physical medium. The ISO 11898 architecture is shown in fig. 1 that defines the lowest two layers of the seven layer OSI/ISO model as the data-link layer and physical layer. These two layers are mostly used for CAN bus communication [1].

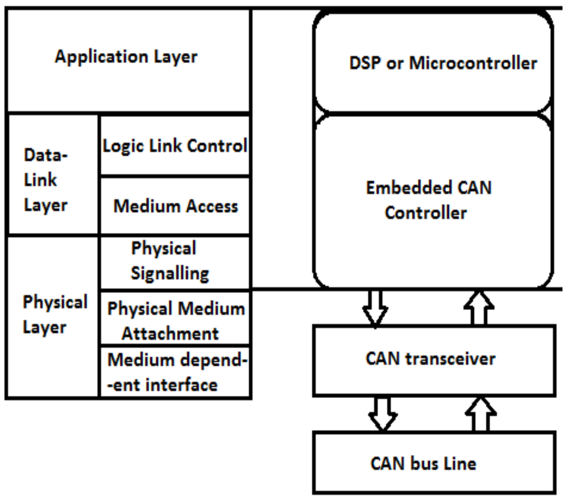

Figure 1. The Layered ISO Standard 11898 Architecture

III. CAN FEATURES
The Controller Area Network protocol has a number of features that makes this protocol so popular that it is now being used in a variety of industries including building automation, medical, and manufacturing. The various features of CAN are as follows:

A. Carrier Sense Multiple Access with Collision Detection
The CAN communication protocol is a CSMA/CD protocol that stands for Carrier Sense Multiple Access with Collision detection. Carrier

Sense means that each node on a system will check the bus line if there is already any transfer of data taking place or not. No node can transfer data until the bus line is free for any data transfer. Multiple Access means that each node on the system can transfer the data without any priority. Each node has an equal priority to transfer the data. Collision Detection means if two nodes on the network start data transmission process at the same time, then nodes on the network will detect the collision and accordingly action will be taken. This action taken is bit wise arbitration that specifies the priorities to the messages and message with high priority will get transmitted [2] .

B. Message Based Communication

CAN protocol is a message-based protocol, not an address based protocol. This means that messages are not transmitted from one node to another node based on addresses. All nodes in the system receive every message transmitted on the bus (and will acknowledge if the message was properly received). It is up to each node in the system to decide whether the message received should be immediately discarded or kept to be processed [3].

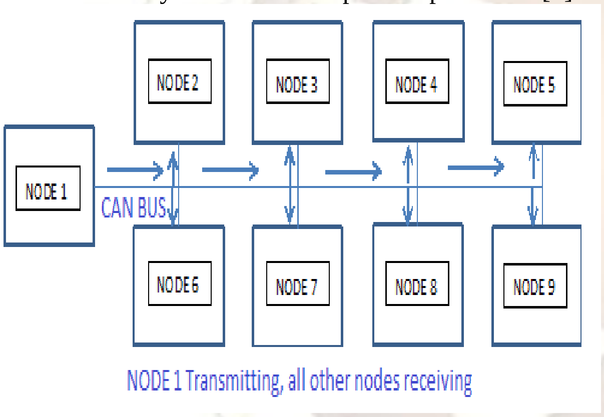

Figure 2. Message Broadcasting by Node 1

C. Remote Transmit Request

Another useful feature of CAN protocol is the ability for a node to request information from other nodes. This is called a Remote Transmit Request (RTR). In this a node is not transmitting the data by itself instead asking for the data from other node.

For example, a safety system in a car gets frequent updates from critical sensors like the airbags, but it may not receive frequent updates from other sensors like the oil pressure sensor or the low battery sensor to make sure they are functioning properly. Periodically, the safety system can request data from these other sensors and perform a thorough safety system check. The system designer can utilize this feature to minimize network traffic while still maintaining the integrity of the network [4].

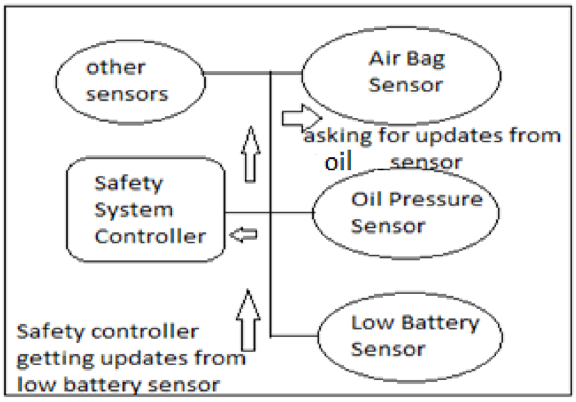
other sensors
Air Bag Sensor
asking for updates from oil sensor
Safety System Controller
Oil Pressure Sensor
Safety controller getting updates from low battery sensor
Low Battery Sensor

Figure 3. Car Safety System

D. Inverted logic

A fundamental CAN characteristic shown in fig 4 is the opposite logic state between the bus, and the driver input and receiver output. Normally, a logic-high is associated with a one, and a logic-low is associated with a zero - but not so on a CAN bus.

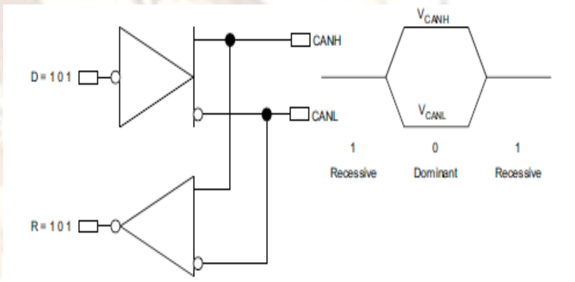

Figure 4. The inverted logic of a CAN bus

On CAN bus, logic 0 is dominant bit and logic 1 is recessive bit as shown above in fig 4.

E. Arbitration

If two nodes try to occupy the bus simultaneously, access is implemented with a non-destructive, bit-wise arbitration. Non-destructive arbitration means that the node winning arbitration continues on with the message. The allocation of priority to messages in the identifier is a feature of CAN that makes it particularly attractive for use within a real-time control environment [5].

The lower the binary message identifier number, the higher its priority. An identifier consisting entirely of zeros is the highest priority message on a network because it holds the bus dominant the longest. Therefore, if two nodes begin to transmit simultaneously the node that sends a last identifier bit as a zero (dominant) while the other nodes send a one (recessive retains control of the CAN bus and goes on to complete its message. A dominant bit always overwrites recessive bit on a CAN bus [6].

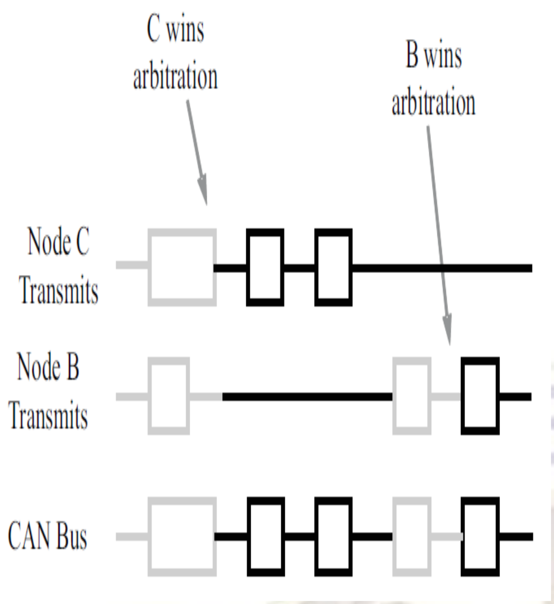

Figure 5. Arbitration on a CAN Bus

Arbitration is clearly shown in fig 5 as shown above.

IV. System Design

This section of paper covers the hardware design of CAN bus system with PIC32 microcontroller, i.e. what components are used in designing the system. The microcontroller used in PIC32MX795F512L, which is a 32 bit microcontroller with two inbuilt CAN modules. We have used ISO1050 CAN transceiver for data transmission and reception.

The power supply for the system is designed which consists of multiple output voltage levels of +5V, Isolated +5V and +3.3V. The power supply is designed from a +9 volt battery for design simplification using voltage regulators of 7805, which provides the +5 volt at its output. Similarly other voltage levels have also been achieved.

JTAG (Joint test action group) programmer is used to program the microcontroller. The other components used are shown in the design itself. In the next section, we see the data format of CAN bus, different message types and error frames that are transmitted and received by different nodes on the network.

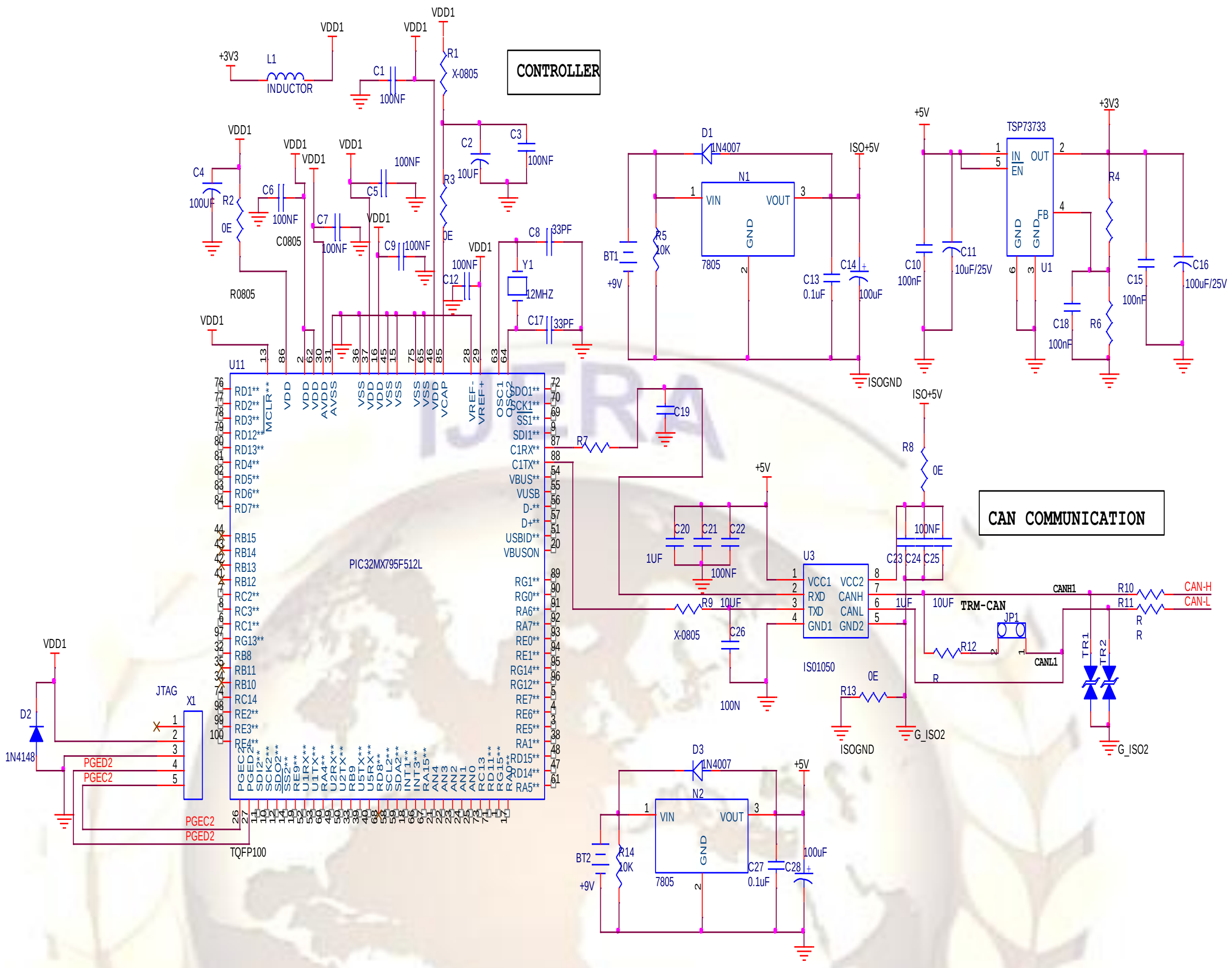

Figure 6. Orcad design of the system

I. Standard Can

S O F	11-bit Identifier	R T R	I D E	r0	DLC	0…8 Bytes Data	CRC	ACK	E O F	I F S

Figure 7. Standard CAN-11 bit identifier

The meaning of the bit fields of Figure 7 are:

- **SOF**- Start of Frame bits tells the start of a message. Start of frame is a single bit always dominant. This bit is used to synchronize the nodes on a bus after being idle.
- **Identifier**-The Standard CAN has 11-bit identifier that establishes the priority of the message. The lower the value of the identifier, the higher its priority.
- **RTR**- This remote transmission request (RTR) bit is used when information is required from another node. This bit is dominant bit. All nodes on the system receive the request, but the identifier determines the specified node by matching its content with node ID. Then that node responds by sending the data. This data is also received by all the nodes and can be used by any interested node.
- **IDE**– This is a single dominant bit naming identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.
- **R0**–Reserved bit (for future use).
- **DLC**–The 4-bit data length code (DLC) contains the number of bytes of data being transmitted. The number of data bytes is obtained by the value written in this data length code.
- **Data**–Up to 8 bytes of application data may be transmitted.
- **CRC**–The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.

- **ACK**–Every node receiving an accurate message overwrites this recessive bit in the original message with a dominate bit, indicating an error-free message has been sent. Should a receiving node detect an error and leave this bit recessive, it discards the message and the sending node repeats the message after re-arbitration. In this way, each node acknowledges (ACK) the integrity of its data. ACK is 2 bits, one is the acknowledgment bit and the second is a delimiter.

- **EOF**–This end-of-frame (EOF), 7-bit field marks the end of a CAN frame (message) and disables bit-stuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is *stuffed* into the data.
- **IFS**–This 7-bit interframe space (IFS) contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area [7].

V. THE CAN BUS

The data link and physical signaling layers of Figure 1, which are normally transparent to a system operator, are included in any controller that implements the CAN protocol, such as PIC32MX795F512L microcontroller with integrated CAN controller. Connection to the physical medium is then implemented through a line transceiver such as ISO1050 CAN transceiver to form a system node as shown in Figure 8.

Balanced differential signaling reduces noise coupling and allows for high signaling rates over twisted-pair cable. Balanced means that the current flowing in each signal line is equal but opposite in direction, resulting in a field-cancelling effect that is a key to low noise emissions. The use of balanced differential receivers and twisted-pair cabling enhance the common-mode rejection and high noise immunity of a CAN bus.

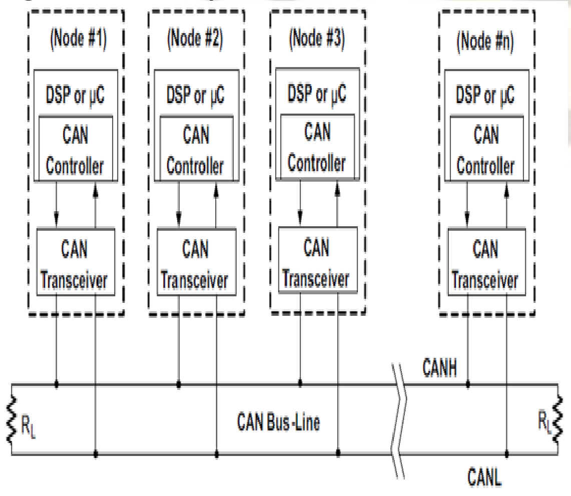

Figure 8. Details of a CAN bus

VI. RESULT

The signals on CAN bus is as shown in Fig. 9.

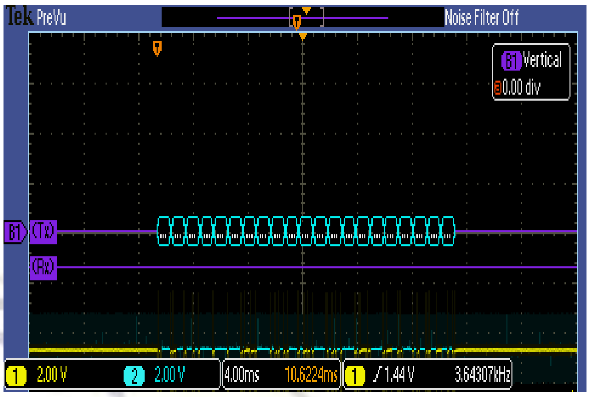

Figure 9. Signals on CAN Bus

REFERENCES

[1] Zeng Xiaohua, Wang Qingnian, Song dafeng, "Application of CAN Bus to HEV based on MPC5xx Microcontroller", (2006), IEEE.

[2] Ping Ran, Baoqiang Wang, Wei Wang, "The design of Communication converter based on CAN Bus," (2008), IEEE.

[3] Xiao-feng WAN,Yi-si XING, Li-xiang CAI, Ahmad Mahin Fallah,"Application and Implementation of CAN Bus Technology in Industry Real-Time Data Communication", International Conference on Industrial Mechatronics and Automation, IEEE (ICIMA 2009).

[4] *Zhifu Zhu*, *Xibin Feng*, "Research on Network for Embedded Numerical Control System Based on CAN Bus", Third International Conference on Genetic and Evolutionary Computing, 2009 IEEE International Conference, pg. 315-318.

[5] Wu Xi-huang, Sun Hai-ping, Chen Li-na, "The Application Research of CAN bus in the City-Bus Control System", 2010 IEEE Conference.

[6] Xiaoming Li, Mingxiong Li, "An Embedded CAN-BUS Communication Module for Measurement and Control System", 2010 IEEE.

[7] Peng Zhou, Ligang Hou," Implementation of CAN bus device driver design Base on Embedded System", 2010 IEEE.