International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 6 1568 – 1573

Implementation of CAN Bus Protocol

Ms. Ashwini S. Shinde
Departement of ENTC
AITRC Vita
Vita, India
ashushinde16@gmail.com

Ms. Aarti S. Deshpande
Departement of ENTC
AITRC Vita
Vita, India
aartideshpande11@gmail.com

Mr. Pradnyant N Kalamkar
Departement of ENTC
AITRC Vita
Vita, India
pnkalamkar4587@gmail.com

Mr. Arjun R. Nichal
Departement of ENTC
AITRC Vita
Vita, India
arjunnichal@gmail.com

***Abstract*—** Controller Area Network (CAN) is a vehicle bus standard protocol designed specifically for automotive application. ECUs (Electronic control units) within vehicle can communicate with each other using CAN Bus standard protocol. It is high speed, bandwidth efficient network. In order to reduce point to point wiring harness in vehicle automation, CAN is suggested as a means for data communication within the vehicle environment. The benefits of CAN bus based network over traditional point to point schemes will over increased flexibility and expandability.

Keywords- *Vehicle Automation, Controller Area Network (CAN), Electronic Control Unit (ECU)*

*****

I. INTRODUCTION

Vehicle system is composed of automotive electrical architecture consists of a large number of electronic control units (ECU) carrying out a variety of control functions. From vehicle system we generally want greater safety, more comfort and less fuel consumption. A modern vehicle may have many electronic control units (ECUs) for various subsystems. Different such subsystems are airbags, antilock braking, engine control, audio system, windows, doors, mirror adjustment etc. Some of these subsystems form independent subsystems, but communication among others is essential. Traditional electronic control system can improve a vehicle dynamics, economy and comfort. But some problems also have come up, such as the body wiring complexity, space constraints and some reliability issues. In order to solve these problems, the vehicle network technology has been created.

In vehicle networking protocols must satisfy requirements which include significant reduction of wiring harness, reducing body weight and costs, improving the efficiency of fault diagnosis, low latency time, configuration flexibility and enhancing the level of intelligent control. Sub systems (ECUs) require the exchange of particular performance and position information within defined communication latency. Therefore the requirement for each ECU is to communicate via some kind of network technology such as CAN (Controller Area Network) bus. At present, some vehicle buses have been already put into use, such as CAN bus, LIN (Local Interconnect Network) bus, Flexray bus etc.

. Before invention of CAN, D2B network is used in car which uses ring topology which has disadvantages that it is not reliable, if one node becomes faulty then whole system collapse. Before CAN, electronic control units are connected in mesh topology. More wiring is required to connect the various ECUs. By using CAN protocol ECUs are connected in bus topology. It offers high speed, less wiring is needed, and it is more reliable because we can connect or delete any node according to our requirement.

II. OVER VIEW OF CAN BUS PROTOCOL

CAN bus is a serial data communication protocol invented by German BOSCH Corporation in the year 1986. CAN is a network protocol which is designed for the car industry. Since data communication in car often have many sensors transmitting small data packets, CAN supports data frames with sizes only up to 8 bytes. Meanwhile, the 8 bytes will not take the bus for a long time, so it ensures real-time communication. CAN protocol has a large amount of overhead, which is combined with a 15-bit CRC, makes CAN very secure and reliable. The CAN protocol standardizes the physical and data link layers, which are the two lowest layers of the open systems interconnection (OSI) communication model. CAN protocol belong to the class of protocols denoted as carrier sense multiple access /collision avoidance (CSMA/CA). The communication rate of CAN based network, depends on the physical distances between the nodes. If the distance is less than 40m, the rate can be up to 1Mbps. CAN bus Protocol has the following properties:

1. Prioritization of messages
2. Guarantee of latency time
3. Configuration flexibility
4. Multicast reception with time synchronization
5. System wide data consistency
6. CAN protocol support Multi-master communication
7. Error detection and signaling
8. Automatic retransmission of corrupted messages as soon as the bus is idle again
9. Distinction between temporary errors and permanent failures of nodes and autonomous switching off of defective nodes.

There are two message formats: Base frame format with 11 identifier bits and Extended frame format with 29 identifier bits.

Start of Frame	Arbitration Field	Control Field	Data Field (up to 8 bytes)	CRC Field	ACK Field	End of Frame

Figure 1. CAN Data Frame

 ISSN: 2321-8169
1568 – 1573

A. Hierarchical structure of CAN BUS

Architecture of CAN protocol based on OSI reference model is as shown in figure 2. CAN protocol contain only three layers, physical layer, data link layer and application layer. Application layer has different protocols such as SAE J1939, CANopen, DeviceNet, etc.

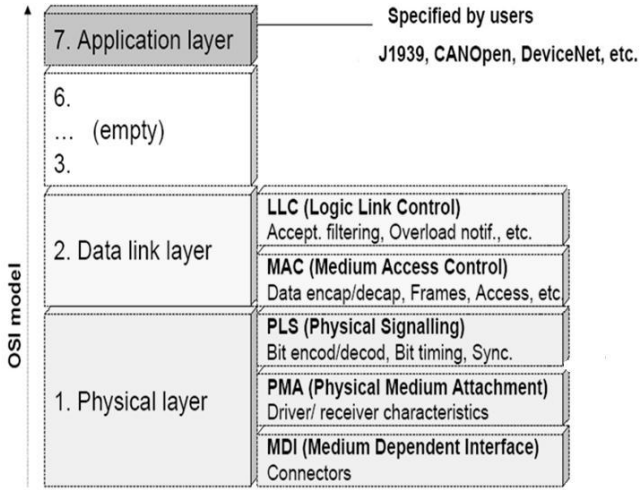

Figure 2. Hierarchical structure of CAN BUS

1. Physical Layer

The physical layer defines how the signals are actually transmitted. Tasks include: signal level, bit representation and transmission medium. Physical layer defines physical and electrical characteristics of the network. Physical layer is same for all the nodes on the same network. Physical layer is related to bit encoding, decoding, bit timing and synchronization.

2. Data link layer

The data link layer is divided into two sub layers, logical link control sub layer (LLC) and medium access sub layer (MAC). Logical link control sub layer (LLC) accept the messages, supports overload notification and recovery management. Medium access sub layer (MAC) performs message framing, arbitration, acknowledgment, error detection and signaling.

3. Application Layer

Application Layer is specified by user. CANOPEN, DeviceNet, SAEJ1938 be the implementation of CAN application layer.

B. CAN Message Frames

CAN has four frame types

1. Data frame: a frame containing node data for transmission
2. Remote frame: a frame requesting the data
3. Error frame: a frame transmitted by any node detecting an error
4. Overload frame: a frame to inject a delay between data and/or remote frame

C. CAN Errors

1. CRC Error

A 15-bit Cyclic Redundancy Check (CRC) value is calculated by the transmitting node and this 15-bit value is transmitted in the CRC field. All nodes on the network receive this message, calculate a CRC and verify that the CRC values match. If the values do not match, a CRC error occurs

2. Acknowledge Error

In the Acknowledge Field of a message, the transmitting node checks if the Acknowledge contains a dominant bit. This dominant bit would acknowledge that at least one node correctly received the message. If this bit is recessive, then no node received the message properly. An Acknowledge Error has occurred.

3. Form Error

If any node detects a dominant bit in one of the following four segments of the message: End of Frame, Inter frame Space, Acknowledge Delimiter or CRC Delimiter, the CAN protocol defines this to be a form violation and a Form Error is generated.

4. Bit Error

A Bit Error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit when monitoring the actual bus level and comparing it to the bit that it has just sent.

5. Stuff Error

CAN protocol use a Non-Return to-Zero (NRZ) transmission method. This means that the bit level is placed on the bus for the entire bit time. Bit stuffing is used to allow receiving nodes to synchronize. If, between the Start of Frame and the CRC Delimiter, six consecutive bits with the same polarity are detected, then stuff Error occurs.

III. OVER VIEW OF SPI PROTOCOL

MCP2515 is Stand-Alone CAN Controller with SPI Interface. The MCP2515 interfaced with microcontroller (LPC2148) via an industry standard Serial Peripheral Interface (SPI). SPI is a full duplex serial interfaces. It can handle multiple masters and slaves being connected to a given bus. Only a single master and a single slave can communicate on the interface during a given data transfer. During a data transfer the master always sends 8 to 16 bits of data to the slave, and the slave always sends a byte of data to the master. 4 pins SCK0, SSEL0, MISO0 and MOSI0 of LPC2148 are used for SPI interface. SCK0 is input/output SPI clock signal used to synchronize the transfer of data across the SPI interface. SSEL0 is input SPI slave select signal is an active low signal that indicates which slave is currently selected to participate in a data transfer. MISO0 is input/output Master In Slave Out signal. The MISO signal is a unidirectional signal used to transfer serial data from the slave to the master. MOSI0 input/output Master Out Slave In signal. The MOSI signal is a unidirectional signal used to transfer serial data from the master to the slave. The SPI contains 5 registers[6].

1. S0SPCR: SPI Control Register, this register controls the operation of the SPI.
2. S0SPSR: SPI Status Register, this register shows the status of the SPI.
3. S0SPDR: SPI Data Register, this bi-directional register provides transmit and receive data for the SPI.
4. S0SPCCR: SPI Clock Counter Register, this register controls the frequency of a master SCK0.
5. S0SPINT: SPI Interrupt Flag, this register contains the interrupt flag for the SPI interface.

A. Master operation

The following sequence describes a data transfer with the SPI block when it is set up to be the master,

 ISSN: 2321-8169
1568 – 1573

1. Set the SPI clock counter register to the desired clock rate.
2. Set the SPI control register to the desired settings.
3. Write the data to transmit to the SPI data register. This write starts the SPI data transfer.
4. Wait for the SPIF bit in the SPI status register to be set to 1 the SPIF bit will be set after the last cycle of the SPI data transfer.
5. Read the SPI status register.
6. Read the received data from the SPI data register (optional).
7. Go to step 3 if more data is required to transmit.

B. Slave operation

The following sequence describes how one should process a data transfer with the SPI block when it is set up to be a slave.

1. Set the SPI control register to the desired settings.
2. Write the data to transmit to the SPI data register (optional). Note that this can only be done when a slave SPI transfer is not in progress.
3. Wait for the SPIF bit in the SPI status register to be set to 1 the SPIF bit will be set after the last sampling clock edge of the SPI data transfer.
4. Read the SPI status register.
5. Read the received data from the SPI data register (optional).
6. Go to step 2 if more data is required to transmit.

IV. HARDWARE DESIGN

The block diagram for CAN bus communication system is as shown in figure 3. Different nodes are nothing but ECUs of vehicle system, which are connected to each other through, CAN bus. Monitor node monitors information on bus and provide it to computer for further analysis. VB6 software is used for collection of data, analysis as well as fault diagnosis of the system.

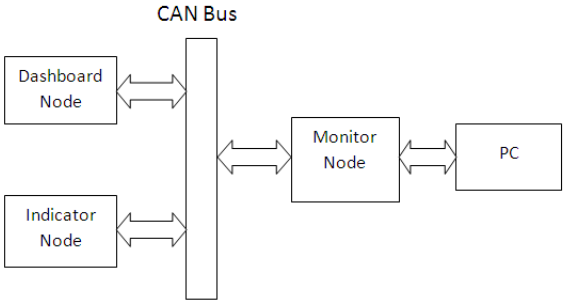

Figure 3. Block Diagram

The devices that are connected by a CAN network are typically sensors, actuators, and other control devices. These devices are not connected directly to the bus, but through a host processor and a CAN controller as shown in above figure IV. CAN node consist of host processor, CAN controller and CAN transceiver. LPC2148 ARM7 TDMI-S processor is used as host processor. MCP2515 is Stand-Alone CAN Controller with SPI Interface. MCP2551 is used as CAN transceiver. The figure 5 shows hardware module of CAN controller and CAN transceiver. This module is connected to LPC2148 host processor. TxCAN and RxCAN pins of the MCP2515 are connected TxD and RxD pins of the MCP2551 respectively. CANH and CANL pins of MCP2551 are connected to CAN bus.

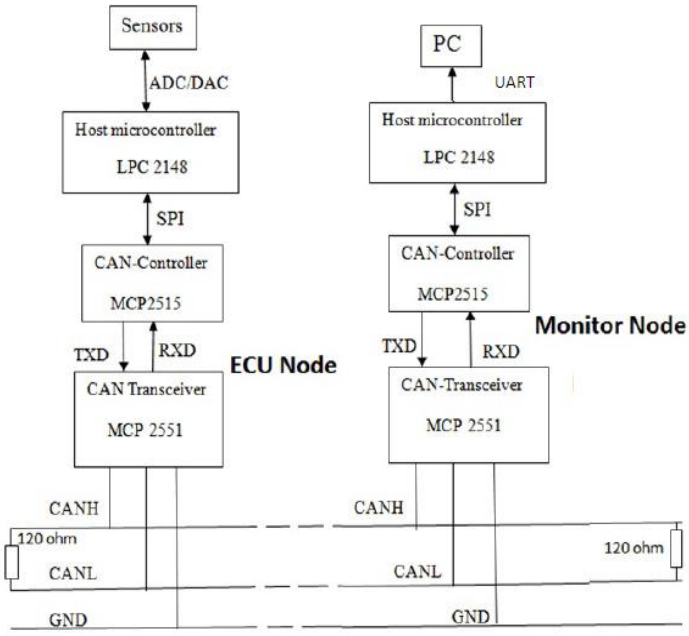

Figure 4. Detailed Block Diagram

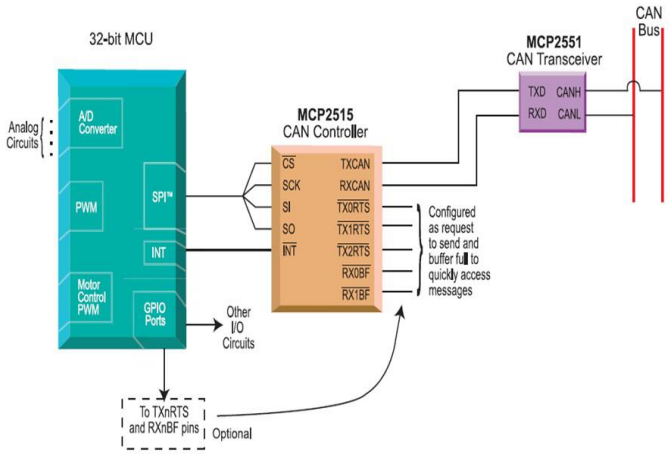

Figure 5: Block Diagram of node

V. SOFTWARE DESIGN

The software part of main program unit is divided into: CAN initialization unit, message sending unit, message receiving unit and interrupt service unit. We have designed three nodes, named as Dashboard node, Indicator node and Bus monitor node. Dashboard node and Indicator node are communicating with each other while Bus monitor node monitors messages available on CAN BUS and these messages are send to PC. Dashboard node and Indicator node operates in normal mode while Bus monitor node operates in listen only mode. Block diagram of Dashboard node is as shown in figure 6. Brake switch is connected at P0.9 pin of LPC2148. AC Ok indicator LED is connected at P0.31, relay is connected at p0.12 and LCD display indicates the status of AC.

 ISSN: 2321-8169
1568 – 1573

ID distribution of Dashboard node is as shown in following figure 7.

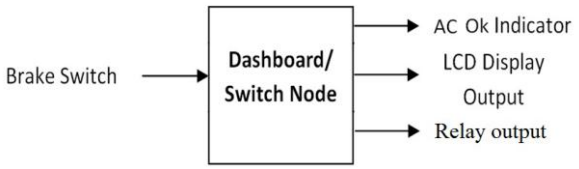

Figure 6: Dashboard node

Back up field	Node ID	Type Field	IDH	IDL	Message Name
b10 b9 b8	B7 b6 b5 b4	b3 b2 b1 b0			
0 0 0	0 0 0 1	0 0 0 1	0x02	0x20	Brake Switch
0 0 0	0 0 0 1	0 0 1 0	0x02	0x40	Reply to AC Switch

Figure 7: Dashboard node ID

Block diagram of Indicator node is as shown in figure 8. AC switch is connected to P0.9 pin of LPC2148. Brake Ok indicator LED is connected at P0.31 pin of LPC2148, LCD display indicates the status of Brake Switch. ID distribution of Indicator node is as shown in following figure 9. Block diagram of Bus monitor node is as shown in figure 10.

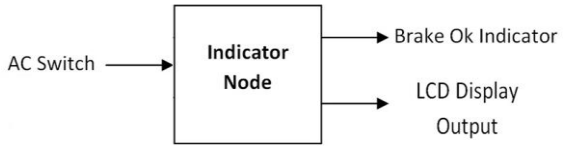

Figure 8: Indicator node

Back up field	Node ID	Type Field	IDH	IDL	Message Name
b10 b9 b8	B7 b6 b5 b4	b3 b2 b1 b0			
0 0 0	0 0 1 0	0 0 0 1	0x04	0x20	AC Switch
0 0 0	0 0 1 0	0 0 1 0	0x04	0x40	Reply to Brake Switch

Figure 9: Indicator node ID

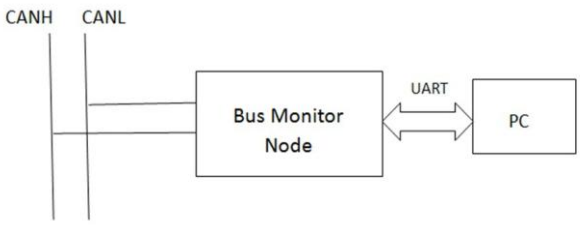

Figure 10: Bus Monitor node

A. CAN Controller Initialization

CAN controller initialization include initialization of transmit and receive buffer of CAN controller MCP2515. When initializing the CAN registers in the MCP2515, the system immediately clear the read and write buffer, Configures the operating mode, bit timing register, filter and mask register, and interrupt enable register.

B. CAN Message Sending

After initialization, the MCP2515 is in normal communication status and is ready to work. The CAN message sending adopts inquiry mode or it sends message when remote request is received. In inquiry mode, after specific time interval, LPC2148 creates packet and sends data to CAN Controller transmit buffer for further transmission.

C. CAN Message Receiving

There are two ways for messages receiving: inquiry mode and interrupt mode. In system design, interrupt mode is implemented. In interrupt mode, if MCP2515 CAN controller receives valid message in receive buffer, it generates interrupt signal on INT pin. Message is read by LPC2148 by using SPI commands of MCP2515.

Flow chart for Dashboard node and Indicator node is as shown in figure 11 & 12

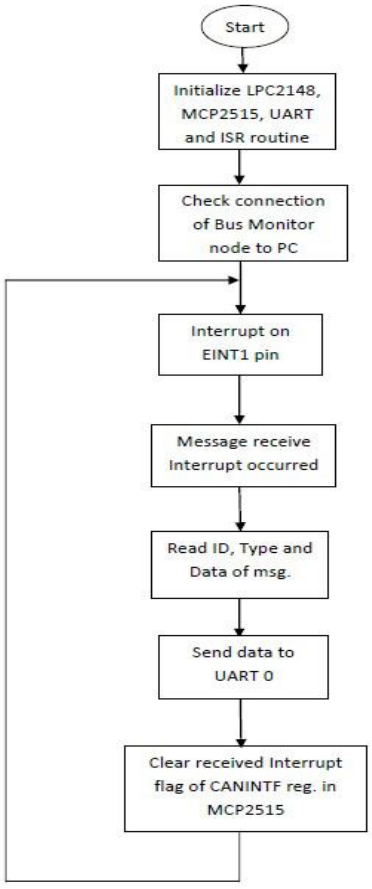

Figure 11: Flow chart for Dashboard node

ISSN: 2321-8169
1568 – 1573

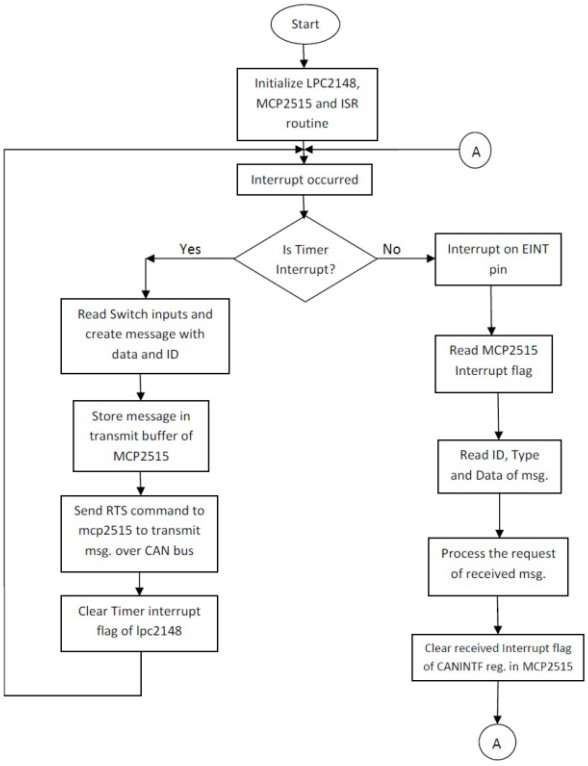

Figure 12: Flow chart for Indicator node

VI. RESULTS

By using UART interface the CAN monitor achieves the reception of CAN messages and time oriented buffering even when dealing with very high bus loads and baud rates. CAN monitor window allows the visualization of the BUS data at the package level and its columns means the following,

1. Serial no. : Indicates number of the received message
2. Date and Time: Indicates the date and time at which data is received
3. ID: Indicates the Identifier of the message
4. Std: Indicates that received message is standard or extended
5. Data: Indicates the data byte of the CAN package
6. Data Length: Indicates the length of the data byte of the CAN package

Different buttons are provided on CAN monitor window to allow user to interact with CAN bus system. "Connect" button checks the connection of PC with hardware. "Start" button starts the collection of messages and buffering, it also saves the messages in database. "Stop" button stops message reception. "Clear All" button clears all contents of database. "Exit" button exits from CAN monitor window.

The output of the CAN Bus is observed on the DSO. The output wave-form can be observed at TxCAN or RxCAN pins of MCP2515 CAN controller with respect to ground terminal. The output waveform shows that bit stream in a CAN bus message is coded according to the Non-Return-to-Zero (NRZ) method.

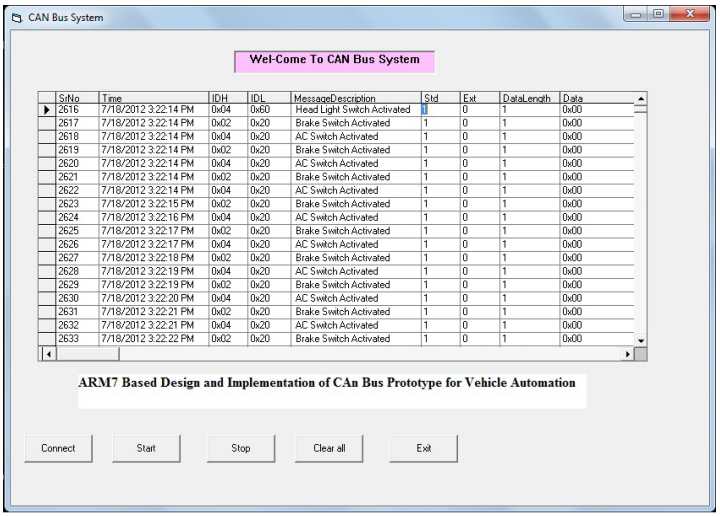

Figure 13: CAN Analyzer

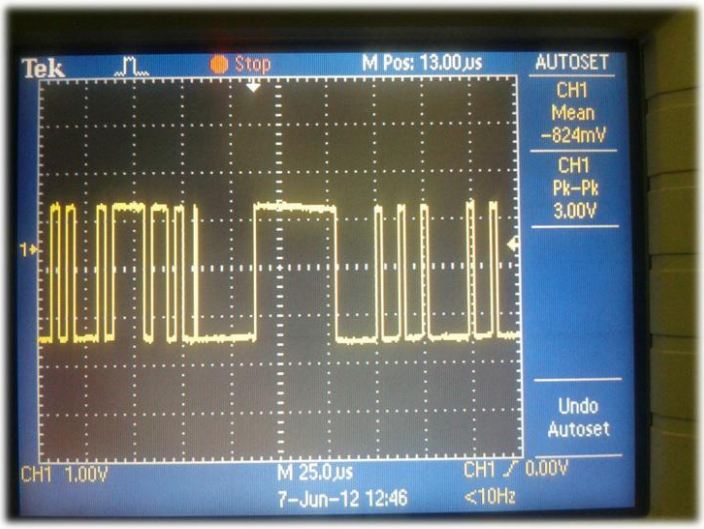

Figure 14: CAN Bus output observed on the DSO

VII. CONCLUSION

The basics of CAN Bus protocol can be understood. The software and hardware can be used to construct a fully working CAN bus for vehicle automation, at various levels of complexity. Basics of packet structure can be seen and understood. CAN Analyzer collects CAN messages and allows time oriented buffering of messages even when dealing with very high bus loads and baud rates. It allows the visualization of the data. Designed CAN Bus system may be adapted for actual ECU node design of vehicle system. We can adapt GUI design to make CAN Analyzer more powerful, versatile tool for the development, testing and servicing of Controller Area Network based systems.

VIII. REFERENCES

[1] Li Ran, Wu Junfeng, Wang Haiying, Li Gechen, "Design Method of CAN BUS Network Communication Structure for Electric Vehicle", IFOST 2010 Proceedings IEEE.

[2] Yujia Wang, Hao Su, Mingjun Zhang, "CAN Bus Based Communication System Research for Modular Underwater Vehicle", 2011 IEEE DOI 10.1109/ICICTA.2011.

[3] He Yu1, Guihe Qin1,2,3, Zhizhong Tian1, Jinnan Dong1,3, " Network-based In-Vehicle Body Bus Control System" , 2010 IEEE.

[4] Chin E. Lin, Hung-Ming Yen, "A PROTOTYPE DUAL CAN-BUS AVIONICS SYSTEM FOR SMALL AIRCRAFT TRANSPORTATION SYSTEM", 1-4244-0378-2/06/20.00 2006 IEEE..

[5] Chris Quigley,Richard McLaughlin, "ELECTRONIC SYSTEM INTEGRATION FOR HYBRID AND ELECTRIC VEHICLES"

[6] UM10139, LPC214x User manual, Rev. 3, 4 October 2010

[7] LPC2141/42/44/46/48, Product data sheet, Rev. 03, 19 October 2007

[8] MikroDes MD2148 ARM7 kit Lab Manual.

[9] MCP2515 Data sheet, 2005 Microchip Technology Inc.

[10] MCP2551 Data sheet, 2007 Microchip Technology Inc.

[11] AN228, Application Note on "A CAN Physical Layer Discussion", 2002 Microchip Technology Inc.

BIOGRAPHY

Ms. A. S. Shinde received her B.E. degree in Electronics Engineering from Shivaji University at PVPIT, Budhgaon in 2009 and received M.Tech in Electronics Engineering from Walchand College of Engineering, Sangli in 2012. Her area of interest is embedded system. She published 1 International journal paper and 1 Ebook.

Ms. A. S. Deshpande received her B.E. degree in Electronics and telecommunication from Pune University in 2009 and received M.Tech in Electronics from Walchand College of engineering, Sangli in 2012. Her area of interest is Digital Image Processing and Communication Engineering. She published 1 International journal papers

Mr. P. N. Kalamkar received his B.E. degree in Electronics from Dr. Babasaheb Ambedkar Marthwada University, Auragabad in 2009 and received M.Tech in Electronics from Walchand College of engineering, Sangli in 2012. His area of interest is Digital Image Processing and VLSI system. He published 1 International journal papers.

Prof.A.R.Nichal received his B.E. degree in Electronics and telecommunication from Shivaji University at Ashta in 2010 and received M.Tech in Electronics from Walchand College of engineering, Sangli in 2012. His area of interest is Digital Image Processing and embedded system. He published 8 International journal papers, 1 Conference paper, 1 Ebook and He has one blog on Fundamentals of Image Processing, Matlab Basics and Embedded System.
URL: www.imagelpcmatlab.blogspot.com