International Journal of Power System Operation and Energy Management

Volume 1 | Issue 3 Article 8

January 2012

FlexRay Communication System in Automotives : Design and Development of Power Window Control System

Narayana Raju Samunuri
Nishitha College of Engg. & Tech, Lemoor (V), Hyderabad, India, narayanarajus@gmail.com

Rajasree Pallati
Nishitha College of Engg. & Tech, Lemoor (V), Hyderabad, India, rajasreemantri@gmail.com

Pradeep Y
Nishitha College of Engg. & Tech, Lemoor (V), Hyderabad, India, pradeepyata@gmail.com

FlexRay Communication System in Automotives : Design and Development of Power Window Control System

Narayana Raju Samunuri, Rajasree Pallati & Pradeep Y

Deptt. of ECE, Nishitha College of Engg. & Tech, Lemoor (V), Hyderabad, India
E-mail: narayanarajus@gmail.com, rajasreemantri@gmail.com, pradeepyata@gmail.com

Abstract **-** Today need for deterministic high data rate and high through put communication protocols for security critical applications and safety applications constantly increases. We can find easily many such applications in the automotive industry. As concepts like x-by-wire with x belongs to drive, steer, brake, etc., where steering arms and brake pipes are replaced by electric wires and integrated circuits, more and more popular, robust and fail-safe communication is needed to realize such functionality. Failures of these critical systems will almost immediately lead to severe accidents, personal injury and great loss to the society in all aspects. FlexRay is an option for upgrading existing network systems using CAN in the automotive industry as well as other industrial control applications. It could also be used for new applications in industrial automation, where safety and reliability in a work environment is of utmost importance, due to its deterministic approach to communication of the messages. This is helped by the use of a two channel topology where each channel is able to work independently, but the two channels can also be used to communicate the same information and as such has built in redundancy [1]. This paper will introduce the key features of the FlexRay protocol in automobile and power (Electric) window sliding design and implementation.

Keywords **-** *FlexRay; MFR4310, Power Windows; Free Master, codewarrior.*

I. INTRODUCTION

The introduction of advanced control systems, which combine multiple sensors, actuators, and electronic control units (ECU), has begun to place boundary demands on the existing CAN (controller area network) communication bus found in most of today's automobiles. As a result, initiatives by automobile Manufacturers and suppliers have led to the creation of FlexRay. Flex Ray is intended to be a new standard which will meet and exceed future requirements for a deterministic and fault-tolerant bus system with high data rates for advanced automotive control applications.

This paper aims to propose the design and implementation of an efficient power window system to demonstrate the FlexRay communication between the ECU and power window node using FlexRay communication protocol. To achieve this, an embedded firmware is developed for two nodes, one is power window node and other is ECU node. The designed and implemented system has ECU node and power window node. The ECU node able collects the data sent by power window node, process the received data. It also transmits the same data to the target PC via UART at a baud of 115200 as per free master protocol user interface commands. The FreeMaster is a scope GUI on PC will be then able to receive the data sent by the ECU via UART and plots the graph and display the same in various formats like hex, ASCII and binary etc along with the error status messages.

II. AUTOMOTIVE NETWORKS

A. History

Electronics was initially introduced to commercially available automobiles in the late 1950s and early 1960s. To work effectively the new electronic devices needed to be able to retrieve information from other devices and sensors located in the car requiring many devices to be interconnected individually. As the number of sensors increased, the wiring increased vastly. The increased wiring introduced problems such as an increase in vehicle weight, causing more possible failure points which affected the reliability of the systems. Also, complex wiring harnesses took up large amounts of vehicle volume, limiting expanded functionality. To overcome the problems associated with the vast amounts of wiring which would be required for future electronic systems which would be implemented in automobiles, it was decided to use Local Area Networks (LAN) to connect a vehicles electronic components. To incorporate LANs in automotive subsystems the car manufacturing companies initially developed their own LAN technologies. Some examples are Controller Area

Network (CAN) by Robert Bosch GmbH, Vehicle Area Network (VAN) by Renault and Peugeot, Automobile Bus (ABUS) by Volkswagen, and J1850 by GM. Ford and Daimler Chrysler.

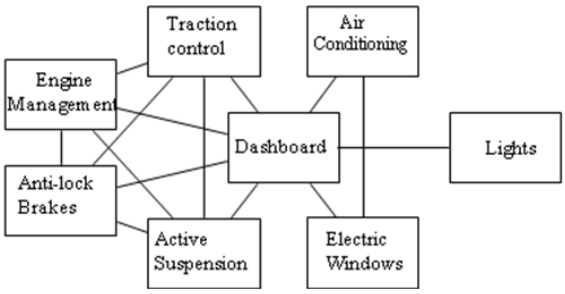

Fig. 1 : Point-to-Point Wiring System

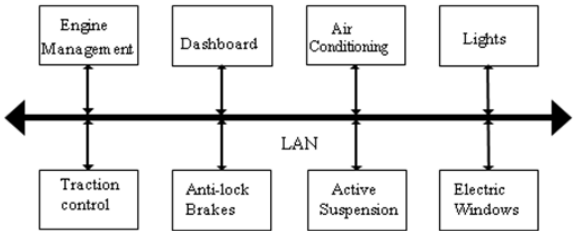

Fig. 2: Local Area Network

However, as many of the automotive vendors share subcontractors there was a need for standardization. One system that became increasing popular in the beginning of the 1990s was the Controller Area Network and it soon became the most used LAN in the automotive industry. Although CAN is a robust, cost-effective general control network, certain newly advanced applications demand more specialized control networks. This need has given rise to a number of new protocols such as Time Triggered CAN (TTCAN), Time Triggered Protocol (TTP), and FlexRay which are fault tolerant, deterministic and have the bandwidth necessary for the next generation of automotive applications. Out of these FlexRay seems set to be-come the de-facto standard.

B. FlexRay Networks

FlexRay is a dual channel, high speed protocol with data rates of up to 20Mb/s (10Mb/s per channel). It is fault-tolerant and deterministic, and is aimed at advanced applications such as X-by-Wire with x belongs to drive, steer, brake, etc., where steering arms and brake pipes are replaced by electric wires and integrated circuits, more and more popular, robust and fail-safe communication is needed to realize such functionality. FlexRay has two channels (A and B) which can be configured separately. This allows data to be transmitted on one channel without the other being used, thus saving bandwidth. For safety critical applications, messages can be transmitted simultaneously on both channels giving a built in redundancy to the network. If one channel gets damaged, transmission will continue without interruption on the other channel. Data is transmitted on the FlexRay bus in both timed and event driven manner. Each message is divided into two sections called the Static Segment and the Dynamic Segment. The static segment is defined during the configuration of the application and transmits the data on a TDMA (Time Division Multiple Access) basis. The Dynamic segment of the message handles data on an event triggered basis.

III. MEDIA ACCESS CONTROL

CAN uses a serial bus priority driven networking scheme but allows for a time triggered communications using a higher protocol such as TTCAN [2]. This means that in a basic CAN system if any two nodes wish to transmit data at the same time, the message with the higher priority can transmit while the other message must wait. In contrast the FlexRay protocol uses a TDMA approach and also allows for a node to send frames in a dynamic way. To do this the protocol defines a recurring cycle called the communications cycle. This cycle has the same format and is of the same time length each time it occurs and in the case of FlexRay is divided into four sections: the static segment, dynamic segment, the symbol window and the network idle time. Fig 3 and figure 4 [3] shows a breakdown of the communication into various sections. Each section is then also broken down into its different slots.

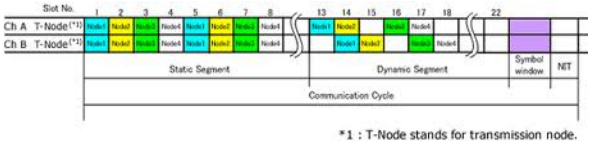

Fig. 3 : Communication cycle

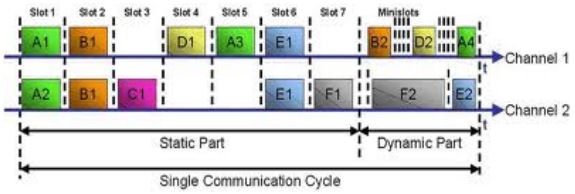

Fig. 4 : Communication cycle

A. TDMA

In a TDMA system the communication cycle is broken down into smaller time segments referred to as slots. The duration of the slots in the static segment is the same. The slots are assigned to a given communication node so that in every communication cycle only that node can transmit at that time. It should be noted that FlexRay does provide a cycle multiplexing system so that information can be sent out every odd cycle for example. This allows another message to be sent in that slot during the even communications cycles and again this message would also be set to that slot in that multiple of the communication cycles. Also a node may get more than one slot per segment depending on the setup of the system and the need to send different messages.

This approach to message arbitration leads to a very deterministic networking scheme making it very suitable for monitoring and safety systems applications

B. Static Segment

The static segment is broken down into smaller sections called static slots. Every static slot is of the same duration. During transmission each slot is assigned to a specific message and only that message can transmit during that slot time.

C. Dynamic Segment

The Dynamic segment is an optional section of the communication cycle. It is broken down into smaller sections known as minislots. If a node wishes to communicate it must wait until its minislot comes around. If no transmission occurs after a given period the minislot counter is incremented and the node with the next message/frame id may begin transmission of data. The data to be sent will only be sent if there is enough time left in the dynamic segment. In this way the dynamic segment is priority driven with the message with the lowest ID having the highest priority, just like CAN.

D. Symbol Window

A symbol is used to signal a need to wake up a cluster amongst other things. This depends on the symbol sent and the status of the controller at the time. Within the symbol window a single symbol may be sent. If there is more than one symbol to be sent then a higher level protocol must determine which symbol gets priority as the FlexRay protocol provides no arbitration for the symbol window.

E. Network Idle Time

The network idle time is used to calculate clock adjustments and correct the node's view of the global time. It also performs communication specific tasks and uses up the remaining time of the communication cycle.

IV. FRAME FORMAT

The frame of a FlexRay message is broken down into 3 sections: the header, payload and trailer section as seen is Fig 5. When compared to the CAN frame format, both standard format and extended format, it can be seen that the frame format of FlexRay is much larger. This is partly due to the extra error checking.

The header section contains status information such as status bits indicating if the fame is a null frame, i.e. contains no payload data, or if the frame should be used for clock synchronization. There are also bits to indicate the length of the payload transmitted and cyclic redundancy check (CRC) bits so the receiver can determine if the header was received correctly.

The payload contains the data to be transmitted over the network. The payload may also be used to transmit more frame information as an option and this would be indicated in the header of the frame. The payload length can vary from 0 to 254 bytes. When compared to the 0 to 8 bytes in a CAN frame this is a significant improvement.

The trailer section contains a 24 bit CRC that is calculated over the payload and header sections. This is used by the receiving node again to determine if the frame was received without any errors.

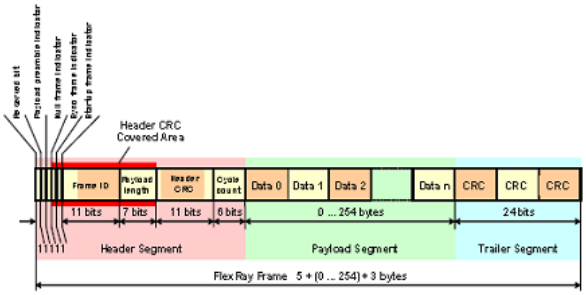

Fig. 5 : FlexRay frame format

V. FLEXRAY IMPLEMENTATION

A. Process Selection

A number of manufacturers offered processors which had the capability to run a FlexRay network most involved a large amount of work interfacing their hardware with the networks, and also a large amount of configuration of the processor. After extensive research, three companies were found to offer viable options for use in FlexRay prototyping. These were:

- Fujitsu FlexRay FPGA Evaluation Kit 369

- Softec Microsystems- SK-S12XDP512-A Development Board containing Freescale HCS12X Processor
- Hitex Infineon TC179x Starter Kit

The major factors to be considered in selecting a processor are Suitability to Automotive Environment, Support for the FlexRay Protocol and Programming Environment. Apart from this the processor is able to operate in the harsh automotive environment also.

B. Suitability to Automotive Environment

For an automotive application, the hardware must be capable of dealing with the conditions s such as vibrations, g-forces, humidity, and temperature extremes. As the components of these boards are all solid state i.e. no moving parts, they are not subject to problems associated with vibrations and g-force. To counter any problems due to humidity, the systems will need to be suitably enclosed for the environment they are placed in. In the automotive environment, ambient operating temperatures can range from -40 C to +125 C. To ensure the processors could work in this environment, their operating temperature ranges were investigated. The results are shown in Table I.

TABLE I : AMBIENT OPERATING TEMPERATURES OF PROCESSORS

processor	**Temperature Range (^{o}c)**
Fujitsu MB91F369	-40 to +105
Free scale MC9S12XDP512	-40 to +125
Infineon TC1796	-40 to +125

C. Supports for the FlexRay Protocol

FlexRay support is the main contributing factor for the selection of a development board. In contrast to the CAN modules all of the development boards have their FlexRay controllers separate from the main processor. These communication controllers are on daughter boards which attach to the development board. Each of the boards has one FlexRay controller and transceiver, and come with driver libraries for interfacing the controllers to the main CPU. Figure: 10, Figure: 11 and Figure: 12 show the Snapshots of Power window node, ECU node and interfacing of ECU node and Power window node respectively.

D. Programming Environment

Programming environment consists of a Freescale's Code warrior Integrated Development Environment (IDE), which is used to develop the code, debugging software, and a hardware interface for transferring the program on to the processor through USB background Debugging (BDM) module. Table II shows the features of the environments supplied with the development boards.

TABLE II: FLEXRAY CAPABILITIES OF DEVELOPMENT BOARDS

Development board	**Controller**	**Controller implementation**	**No. Of flex ray transceivers**	**Flexray driver library**
Softec	Freescale MC9S12x512 Microcontroller	Main Board	1	YES
Softec	1X Free scale MFR4310	Daughter board	1	YES

VI. FLEXRAY HARDWARE

The MC9S12XDP512[4] has a built in CAN module but no built in FlexRay module. This guides us to have a board with hardware to convert the necessary information to the FlexRay protocol. The chip used to meet the above requirement, the board uses the Freescale MFR4310 [5] Communication Controller, which in turn communicates with the Philips TJA1080 FlexRay Transceiver, to transmit / receive messages on the FlexRay bus. Figure 6 shows the physical configuration of the proposed FlexRay network. Each node consists of a HCS12X microcontroller connected to the FlexRay communication controller (CC) which is connected to the FlexRay Transceiver. For this implementation the CC and transceiver are mounted on one module called a daughter card. This is connected to the development board via 2 x 50 pin sockets. The daughter card also contains the various switches, jumpers and other components necessary for interfacing with the FlexRay bus.

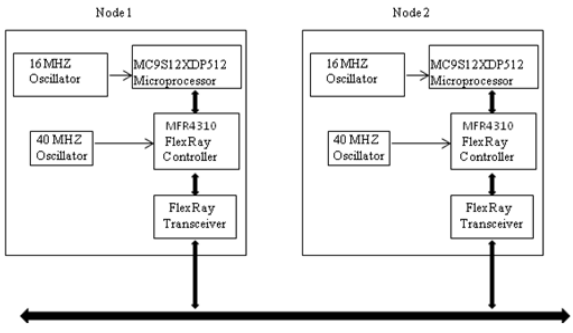

Fig. 6 : Block Diagram of FlexRay Hardware

A. MC9S12XD_Family

The MC9S12XD[4] family will retain the low cost, power consumption, EMC and code-size efficiency advantages currently enjoyed by users of Freescale's existing 16-Bit MC9S12 MCU Family. Based around an enhanced S12 core, the MC9S12XD family will deliver 2 to 5 times the performance of a 25-MHz S12 whilst retaining a high degree of pin and code compatibility with the S12. The MC9S12XD family introduces the performance boosting XGATE module. Using enhanced DMA functionality, this parallel processing module offloads the CPU by providing high-speed data processing and transfer between peripheral modules, RAM, Flash EEPROM and I/O ports. Providing up to 80 MIPS of performance additional to the CPU, the XGATE can access all peripherals, Flash EEPROM and the RAM block.

The non-multiplexed expanded bus interface available on the 144-pin versions allows an easy interface to external memories. The inclusion of a PLL circuit allows power consumption and performance to be adjusted to suit operational requirements. System power consumption can be further improved with the new “fast exit from stop mode” feature. In addition to the I/O ports available in each module, up to 25 further I/O ports are available with interrupt capability allowing wake-up from stop or wait mode.

B. MFR4310 FlexRay Controller

The MFR4310 FlexRay controller has two separate controller host interfaces (CHI) on board - an HCS12 interface, for direct connection to Freescale's HCS12 family of microcontrollers, and an asynchronous memory interface (AMI) for asynchronous connection to all other microcontrollers. The HCS12 interface clock signal, used to synchronize the data transfer, can run at a maximum rate of 8 MHz.

VII. FLEXRAY SOFTWARE

The function of FlexRay software application was to periodically transmit a value from Node 2 onto the FlexRay bus using Slot 1 in the static segment, receive this message on Node 1 where it is incremented, and return the edited value back onto the bus using Slot 4 in the static segment. From there it is received again by Node 2. The FlexRay network was configured with the values shown in Table III

TABLE III : FLEXRAY CONFIGURATION

Bit Rate	**10 Mb/s**
Macrotick Length	1_S
Communication Cycle Length	5000MT
Static Segment	3000MT
No. of Static Slots	60
Static Slot Length	50MT
Dynamic Segment	880MT
Max. No. of Minislots	22
Minislots Length	40MT

The below flowchart in figure: 7 shows how the messages are transmitted from one node to another node.

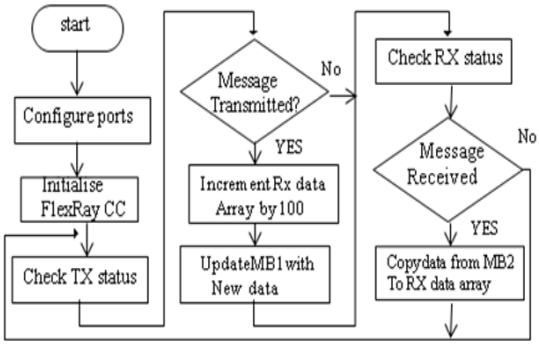

Fig. 7 : Flowchart of FlexRay Node 1 Software Operation

A. Node 2 Software Implementation

A flowchart of the software used in Node 2 is shown in Figure: 8. this runs on an interrupt driven basis. It receives data from the bus and retransmits it on the same bus, but in a different slot in the dynamic segment.

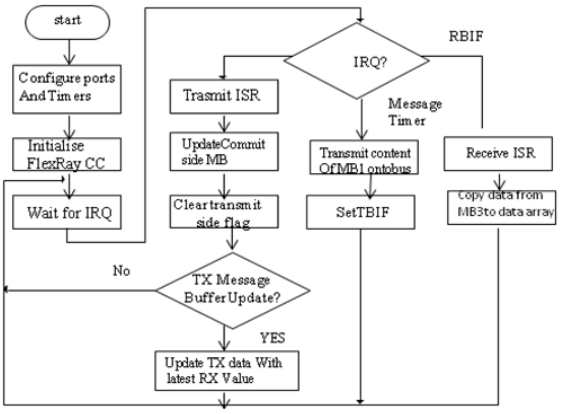

Fig. 8 : Flowchart of FlexRay Node 2 Software Operation

VIII. TESTING OF FLEXRAY IMPLEMENTATION

A software GUI tool called FreeMaster was used to monitor activity on the FlexRay bus. FreeMaster is a GUI developed by Freescale, for use with their processors, to display real time values from various registers within a microprocessor onto the screen of a PC / Laptop.

Free Master has features like Graphical environment, Easy to understand navigation , Simple RS232 connection, Real-time access to embedded-side C variables, Visualization of real-time data in Scope window, Acquisition of fast data changes using on-target Recorder, Built-in support for standard g point, bit fields)variable types (integer, floating), Value interpretation using custom defined text messages, Several built-in transformations for real type variables, Automatic C-application variable extraction from Metrowerks CodeWarrior output files and Remote Communication Server enabling a connection to target board over a network, including the Internet.

To set up an application, a user must first place a driver in the embedded code which allows the GUI access to the registers that are to be monitored. The information is exported from the HCS12X via a SCI (Serial Communication Interface) link to an RS232 transceiver, from where it is then transmitted to the Serial Port of the PC (See Figure 9). The GUI also needs to be configured to take the information from the relevant registers within the HCS12X. Once this is complete the user can manipulate the received data to display it on the screen in numerical, graphical or message based form

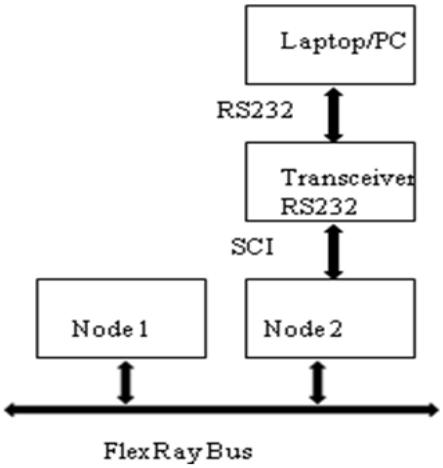

Fig. 9 : FreeMaster being used to monitor variables on Node 2

Fig. 10 : HCS12x Starter Kit configured as Power window node with DC motor

Fig. 11 : HCS12X Starter Kit configured as ECU node

Fig. 12 : Interfacing ECU node and Power window node

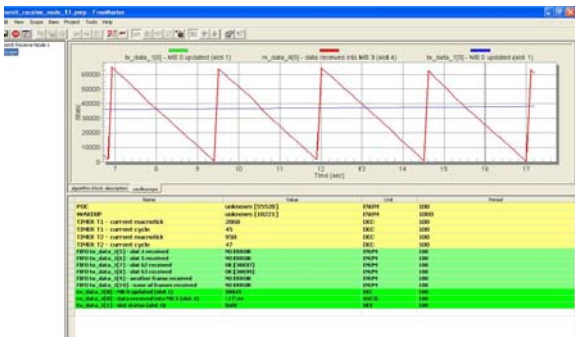

Fig. 13 : Data received on Power window node without switch press at ECU node

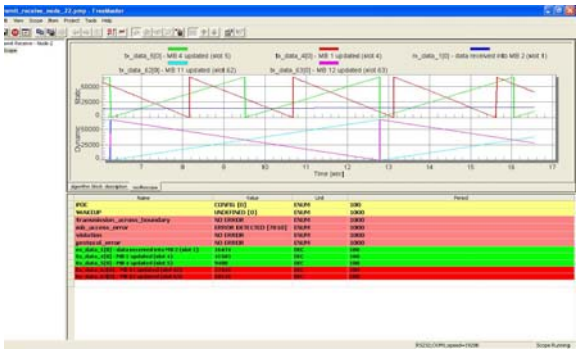

Fig. 14 : Data received on ECU node without switch press at ECU node

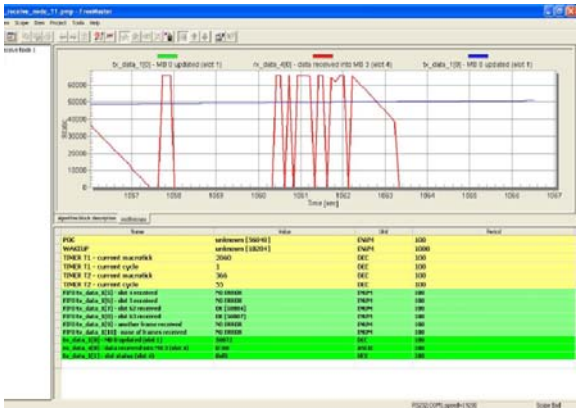

Fig. 15 : Data received on Power window node with switch “UP”press at ECU node

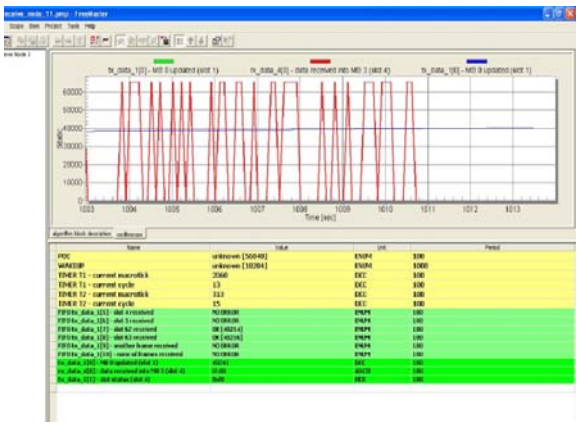

Fig. 16 : Data received on Power window node with switch “DOWN” press at ECU node

CONCLUSION

The aim of this paper is to design and implement a high reliable communication between a FlexRay networks through power window switch and various other nodes. We have successfully implemented power window lifting system with the aim of reducing CPU loading, and maximizing edibility of implementation. The framework was then implemented in a real application. The function of the application was to monitor the data flow on the FlexRay bus and transmit the relevant data in real time to a FreeMaster dashboard via UART at a baud of 115200. The data on the FlexRay bus consisted of information from switch connected to one node on the FlexRay bus and also huge data was pushed onto the bus, to demonstrate the how the ECU recognize and performs the necessary action on the power window application.

After a thorough review of the development boards available for use with this project, the Softec development environment containing a Freescale HSC12X processor was chosen for use in the implementation of the system, as it adequately met all of the key requirements for the system. As mentioned above, the data on the FlexRay bus was represented on a FreeMaster dash panel. We also have used LEDs to display the status of the data transmission between the nodes upon an occurrence of power switch press events. The FreeMaster GUI is developed in such a way that it will draw the waveform to represent the data flow values and also the same is displayed on the GUI with various supported formats like, ASCII, binary, decimal etc. The software based dash board designed intelligently, to display the error message on occurrence of any fault on the flexray bus. Figure: 13 and Figure: 14 show the data transmission and reception on both the Power node and ECU node respectively. Figure: 15 and

Figure: 16 show the transmission of data during switch "UP" press and "DOWN" press respectively. This can be expanded to encompass a number of networks using the framework described, and allowing various topologies to be used

ACKNOWLEDGMENT

On the completion of our paper, we should like to express our deepest gratitude to Freescale for their continuous support though their mail support channel system in resolving the issues with the HCS12X Starter Kit and also to all those kindness and advices have made this work possible. The authors also acknowledge the wonderful support and encouragement from Dr. Vijaya Bhaskar Mantri, starting stage to till end stage of this work. We would like to convey our sincere thanks Sri Satyanarayana Reddy B, correspondent of Aryabhata Instituite of Technology and science, Hyderabad, for constantly motivating and supporting us to achieve this success. Also, we would like to thank our friends Mr.Veerasekhara.D, Ramanjaneya Reddy.K who gave us valuable instructions to complete this paper with error free. The Authors wish acknowledge their sponsor: EmWare Technologies (INDIA) Pvt. Ltd, Hyderabad.

REFERENCES

[1] An Introduction to FlexRay as an Industrial Network , Robert Shaw, Brendan Jackman Automotive Control Group, Waterford Institute of Technology, Waterford, Ireland. E-mail: rshaw@wit.ie, bjackman@wit.ie Website: http://www.wit.ie/automotive G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] CiA (2007) Controller Area Network (CAN) [online], available: http://www.can-cia.org/ [accessed 30 Oct 2007].

[3] FlexRay Consortium (2005) FlexRay Communication System Protocol Specification, Version 2.1 Revision A, Stuttgart: FlexRay Consortium GbR

[4] MC9S12C512 Device User Guide, Freescale Semiconductors, (2010)

[5] MFR4310 Data Sheet, Freescale Semiconductors, (2010)

◈◈◈