CAN Protocol Based Assembly Line Management

Mr.M.V.N.R.P.Kumar[1],Ms.B.Hombal[2],Mr. S. A. Bhosale[3], Mr. S.B. Katte[4], Ms. R.B. Bardeskar[5]
Dept. of E&TC, L.N.B.C.I.E.T. Raigaon, Satara (Maharashtra).
Email: mvnrpk@yahoo.com[1],sunilbhosale4025@gmail.com[3], sandeshkatte@gmail.com[4], ritz15bk@gmail.com[5].

Abstract: In the automobile industry, embedded control has grown from stand-alone systems to highly integrated and networked control systems. With the increasing number of distributed microcontrollers and intelligent peripherals used in today's electronic systems, such as vehicle controls, networking protocols between the units have become extremely important. A wide range of these applications are using CAN (Controller Area Network) for network communication. The CAN bus is a multi-master, message broadcast system that specifies a maximum signalling rate of 1M bit per second (bps). Unlike a traditional network such as USB or Ethernet, CAN does not send large blocks of data point-to-point from node A to node B under the supervision of a central bus master. In a CAN network many short messages like temperature or RPM are broadcast to the entire network, which allows for data consistency in every node of the system. [1]

Index Term: Control Area Network (CAN), CANH, CANL, Protocol, CAN bus.

1. INTRODUCTION:

Before a decade technology has reached a level that made it possible to design a machine controlled by a microprocessor which is standalone. These standalone systems are nodes in a serial network, a Controller Area Network, and are coordinated by a master computer. Such machine architecture has several advantages, not only technical but also project managerial. In spite of this the development within this area has made little progress until recent time. The main reason for this has been the lack of a reliable standard protocol suitable for fast hard real-time communication. Such a protocol, designated CAN, was however developed for the automobile industry and is now implemented in chips by the leading chip manufacturers.In the early 1980s, engineers were evaluating existing serial bus systems regarding their possible use in passenger cars. Because none of the available network protocols were able to fulfil the requirements of the automotive engineers. The new bus protocol was mainly supposed to add new functionality – the reduction of wiring harnesses was just a by-product, but not the driving force behind the development of CAN.[1]

This technology is used to control the problems that arrived on the assembly line of the automobile industry. Sometimes the parts of the vehicle are not provided to worker by the storeroom due to this reason the assembly line is stopped to minimize this problem CAN bus technology.There are two different modes for communication .Full CAN and Basic CAN. The basic CAN requires all the parameters to be set in firmware. It uses hardware for message filtering. Basic CAN require that the CPU is interrupted every time a message is received to determine whether it is accepted or not. It consist 4 nodes namely manager node, store node and two nodes of worker that assembles the parts to the chase.

2. LITERATURE SURVEY:

Two varieties of CAN exists today, basic CAN or a higher level Full CAN with acceptance filtering hardware. Can protocol allows for two identifier field lengths: part A specifies 11 bits, which allows 2032 different ID´s out of 2048, while extended CAN (part B) has 29 bits giving over 536 million unique identifiers. The information recorded and processed by each one is often used by one or more others nodes. The CAN protocol describes the method by which information is passed between devices.[1]

It introduces the basic concepts of CAN protocol and demonstrate show CAN bus communication can be implemented using PSoC3 and PSoC 5LP (hereafter referred to as PSoC) and software implementation of control system through Vector CANo/CAN analyzer software for nodes. The PSoC outputs from the CAN component are TX and RX. These outputs need to be level translated in order to obtain the CANH and CANL signals. Now since each node has its own compiler and editor so each of the node is programmed individually on its own editor such if any of the node is removed the system remain unaffected and work efficiently. CANoe/CAN analyzer is the software tool which used for the system design. The system design using CANoe/CAN analyzer involves six steps. [3]

3. MODIFICATION:

Normally the can protocol is used inside the vehicle to control the various parameters of the vehicle and agricultural applications. Here instead of this it is used for automation in the assembly line.

4. TECHNOLOGY:

As all modern communication technologies, CAN-based networks follow the internationally standardized

Open System Interconnect (OSI) reference model as defined in ISO 7498-I. For CAN technology, this model has been adapted. The CAN reference model comprises the CAN physical layer (Layer 1), the CAN data link layer (Layer 2), and the CAN application layer -7 solution include also network functionality. All this communication protocols specify just the communication. In order to verify the implementations standardised conformance test plans are necessary. A manufacturer independent certification based on these test plans minimizes capability problems.

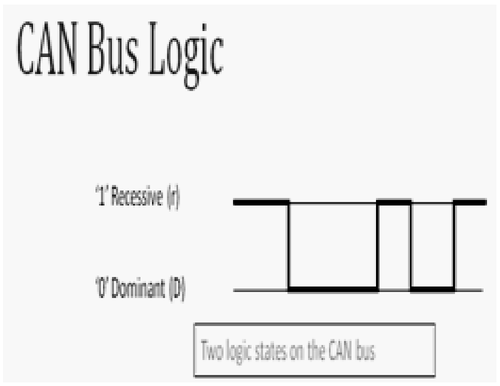


Fig 1.CAN bus logic.

The arbitration field contains the packet of 11 bit destination address and Remote Transmission Request (RTR) bit. Above figure 1 shows the CAN Bus Logic. When this bit is at '1' it means this packet is for the destination address (ie. recessive state). If this bit at 0' (dominant state) it means this packet is a request for the data from a device.Each CAN station is physically connected to the CAN bus through a transceiver. The transeceiver is capable of driving the large current needed for the can bus and has current protection against a defective CAN bus or defective station. [4]

4.1. Standard CAN:

IFS	11 Bit Identifier	RTR	IDF	R0	DLC	0-----8 Byte Data	CRC	ACK	EOF	IFS

Fig 2.Standard CAN: 11 bit identifier

The data link idles in the recessive state, and when any node wishes to initiate a data frame its outputs a dominant bit. The arbitration field which follows is made up mainly of an identifier, which both identifies the message and encodes its priority. The control field includes a data length code, which indicates the number of bytes of data, up to maximum of eight, contained within the following data field. The CRC field contains data used in the Cyclic Redundancy Check as shown in fig.2. The acknowledge field is just two bits long a receiver which successfully receives a message asserts a dominant bit here. The end of frame is made of seven recessive bits.[2]

4.2. Arbitration:

A fundamental CAN characteristic shown in Figure 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. In the absence of any input, the device automatically defaults to a recessive bus state on all input and output pins.[2]

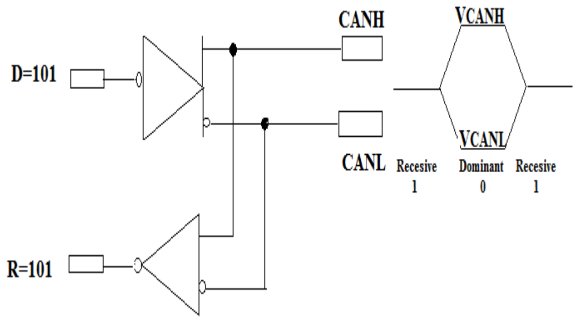


Fig 3.The Inverted Logic of a CAN Bus

Above figure 3 shows the inverted logic of a CAN bus. Bus access is event-driven and takes place randomly. If two nodes try to occupy the bus simultaneously, access is implemented with a non-destructive, bit-wise arbitration. Non-destructive means that the node winning arbitration just continues on with the message, without the message being destroyed or corrupted by another node. [2]

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. 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 a recessive bit on a CAN bus [2].

4.3. Error Checking and Fault Confinement:

The robustness of CAN may be attributed in part to its abundant error-checking procedures. The CAN protocol incorporates five methods of error checking: three at the message level and two at the bit level. If a message fails any one of these error detection methods, it is not accepted and an error frame is generated from the receiving node. This forces the transmitting node to resend the message until it is received correctly. However, if a faulty node hangs up a bus by continuously repeating an error, its transmit capability is removed by its controller after an error limit is reached. [2]

Error checking at the message level is enforced by the CRC and the ACK slots displayed in Fig 2 and Fig 3. The 16-bit CRC contains the checksum of the preceding application data for error detection with a 15-bit checksum and 1-bit delimiter. The ACK field is two bits long and consists of the acknowledge bit and an acknowledge delimiter bit. [2]

Also at the message level is a form check. This check looks for fields in the message which must always be recessive bits. If a dominant bit is detected, an error is generated. The bits checked are the SOF, EOF, ACK delimiter, and the CRC delimiter bits. [2]

At the bit level, each bit transmitted is monitored by the transmitter of the message. If a data bit (not arbitration bit) is written onto the bus and its opposite is read, an error is generated. The only exceptions to this are with the message identifier field which is used for arbitration, and the acknowledge slot which requires a recessive bit to be overwritten by a dominant bit.

The final method of error detection is with the bit-stuffing rule where after five consecutive bits of the same logic level, if the next bit is not a complement, an error is generated. Stuffing ensures that rising edges are available for on-going synchronization of the network. Stuffing also ensures that streams of bits are not mistaken for an error frame, or the seven-bit inter frame space that signifies the end of a message. Stuffed bits are removed by a receiving node's controller before the data is forwarded to the application. With this logic, an active error frame consists of six dominant bits violating the bit stuffing rule. This is interpreted as an error by all of the CAN nodes which then generate their own error frame. This means that an error frame can be from the original six bits to twelve bits long with all the replies. This error frame is then followed by a delimiter field of eight recessive bits and a bus idle period before the corrupted message is retransmitted. It is important to note that the retransmitted message still has to contend for arbitration on the bus. [2]

5. SYSTEM DESCRIPTON:

There are total four nodes are present namely station 1, station 2, store and manager respectively. Station 1 and station 2 are the input stations. Each station having four IR sensors. These two input stations are connected to the store and manager through a CAN bus. The content available at the station one and two are continuously displayed at the output.

5.1. Block Diagram:

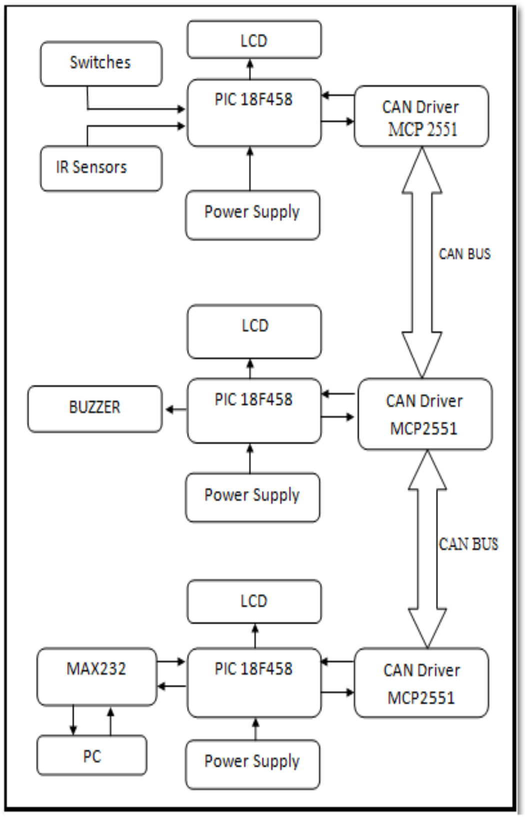


Fig 4.Block Diagram of CAN Automation

5.2. Description:

Total four nodes are there as shown in fig. 4. In which two are worker stations, one is store room and other is manager. Four stations are connected through CAN bus. At the first two stations the mechanical parts are assemble to the chasses. At the single station four parts are assembled by a single worker. The IR sensors are attached to a rack. When the part is picked up from the rack the IR sensor decrements from the available number of the parts. Available information about the parts is given to store room and manager continuously by using the LCD display. When the available parts goes to critical condition (ie. Only 3 parts available) the message is given to both manager and store room through CAN bus. And buzzer gets started at critical condition. Manager observe all the working condition on the PC through VB. whichever part is goes below the critical condition is provided in the standard contain of 10 parts. When the parts are provided by worker the respective switch is pressed to increase the amount of available parts.

6. ASSEMBLY MODEL:

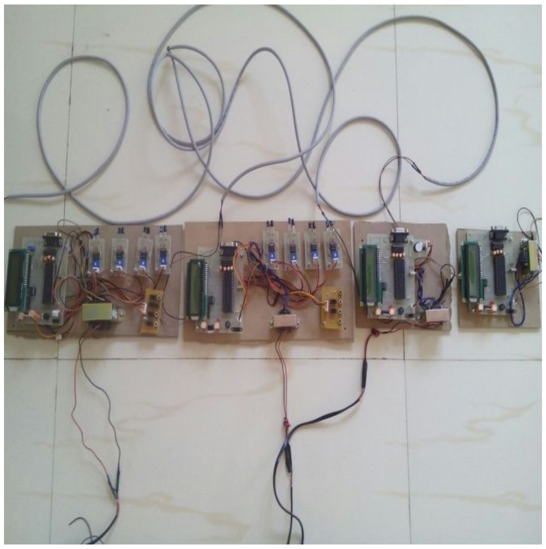
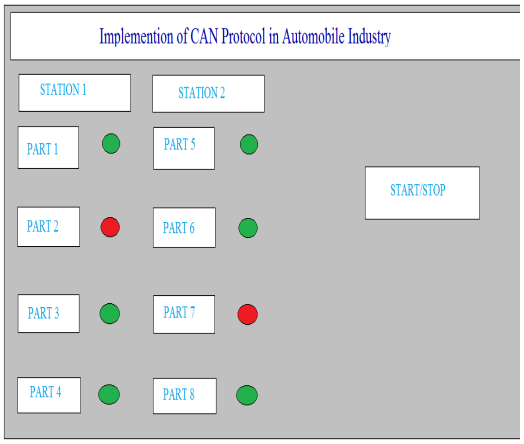


Fig 5.Assembly model

Above fig 5 shows the total hardware of the project. It contains the four nodes (station 1, station 2, store and manager) and CAN Bus.

7. RESULT:



Implemention of CAN Protocol in Automobile Industry

STATION 1 | STATION 2

PART 1 ● | PART 5 ●

PART 2 ● | PART 6 ●

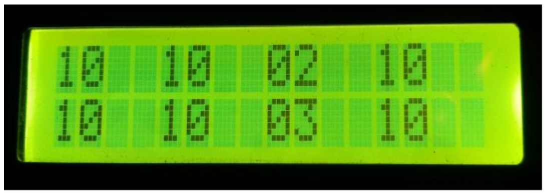
PART 3 ● | PART 7 ●

PART 4 ● | PART 8 ●

START/STOP

Fig 6. Manager station

Above fig. 6 shows the display of monitor which is at the manager station. If the number of components goes below critical condition that is less than 4 then the display indicates red otherwise it will be green.



```
10 10 02 10
10 10 03 10
```

Fig 7.Store room station

Above fig 7 shows number of parts present at each station. First row first Column number indicates that the number of parts available at station 1 of first type. Similarly other number shows the available parts present at the respective stations. If the number of parts at any station goes below critical condition then buzzer will be alarm.

8. ADVANTAGES:

1) CAN is a message based protocol. In CAN, nodes are not assigned specific addresses in CAN Networks. This provides flexibility to add or remove a node to/from the network.
2) CAN protocol ensure that even if one of the nodes fails, others continue to work and communicate properly.[1]
3) CAN network offers system wide data Consistency in CAN networks, corrupted messages are automatically retransmitted as soon as the bus is idle again.[1]
4) CAN incorporate a five level error Checking capability to ensure reliable traffic and data integrity.[1]
5) It minimizes the manual work between worker on the assembly line and the store room.
6) The assembly line doesn't stop because of continuously displaying available parts.

9. APPLICATIONS:

1) Building Automation:Air Conditioning, Access & Light Control.
2) Domestic & Food distribution appliances: Washing machines, dishes cleaner, self- Service bottle distributors.
3) Automotive & Transportation:
   Dash Board, Power train & Car Body.
   Train, bus and truck equipment.
4) Agriculture:
   Harvester, seeding, sowing Machines, tractor control.

10. LIMITATIONS:

1) Requires bit dominance.
2) Propagation delay limits bus length.
3) Unfair access.
4) Poor latency for low priority nodes.

11. FUTURESCOPE:

To obtain larger network sizes in CAN, two solutions can be introduced such as to reduce the bus speed and use of interconnection devices, such as bridges. The data word length can be increased so we can send more information or message.

12. CONCLUSION:

The some problems that arrived on the assembly line of the automobile industry i.e sometimes the parts of the vehicle are not provided by worker from the storeroom due to this reason the assembly line may be stop. This project aims to control this problem by using CAN bus technology.

REFERENCES

[1]International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-2, Issue-5, April 2013 by Divya Sharma and Mayank Gupta.

[2]Application Report: TEXAS INSTRUMENT (SLOA101A–August 2002–Revised July 2008)

[3] Ranjit M, PSoC 3 and PSoC 5LP –getting started with CAN, AN52701, Software Version : PSoC Creator 2.1 SPI.

[4]BOSCH Controller Area Network (CAN) version 2.0 http://www.freescale.com

[5] H. W. Huang, "PIC Microcontroller: An Introduction to Software and Hardware Interfacing," 2005. http://www.delmar.com.

[6] H. Rongen, "Introduction to PIC Microcontroller," For-schungszentrum Jülich Zentrallabor für Elektronik, Jülich, 2009.

[7] N. Gardner, "An introduction to Programming the Mi- crochip PIC in CCS C," Ccs Inc., Christiansburg, 2002.

[8] L. D. Jasio, *et al.*, "PIC Microcontrollers," Elsevier Inc., New York, 2008.

[9] T. Wilmshurst, "Designing Embedded Systems with PIC Microcontrollers Principles and Applications," Elsevier Ltd., New York, 2007.

[10] N. Matic and G. Maneger, "EasyPIC Microcontroller Board User Manual," 2008.

[11] D. Ibrahim, "Advanced PIC Microcontroller Projects in C from USB to RTOS with the PIC18F Series," Elsevier, Amsterdam, 2008.