International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 25859-25866

http://www.ripublication.com

Reducing Can Bit-Stuffing Using Selective Xoring

B.Pazhanthanigai Malar[1], B.Vinodhkumar [2] and Dr. J.Ramesh[3]
[1]M.E-Embedded System Technologies, Sri Shakthi Institute of Engineering and Technology, Coimbatore
[2]Associate Professor, Sri Shakthi Institute of Engineering and Technology, Coimbatore
[3] Assistant Professor (senior Grade), PSG College of Technology, Coimbatore

Abstract

This paper presents a new approach for reducing bit-stuffing in Controller Area Network (CAN). CAN uses a bit-stuffing mechanism to prevent six consecutive bits from having the same polarity by inserting a bit of opposite polarity after the fifth bit. The added bits eliminate the forbidden patterns but cause an increase in frame length which worsens the timing accuracy of a network. So the selective XORing method is proposed to reduce the bit-stuffing. The essence of this analysis is to explore on reducing bit-stuffing for the worst case scenarios. The proposed method is validated through Simulink implementation.

Index Terms: CAN, Bit-stuffing, selective XORing, Simulink.

Introduction

Development of the CAN bus started in 1983 at Robert Bosch GmbH. The protocol was officially released in 1986 at the Society of Automotive Engineers(SAE) congress. Bosch published several versions of the CAN specification and the latest is CAN 2.0 published in 1991. This specification has two parts; part A is for the standard format with an 11-bit identifier,commonly called CAN 2.0Aand part B is for the extended format with a 29-bit identifier, called CAN 2.0B. Currently there are hundreds of millions of CAN nodes in use in the world.

CAN 2.0A Data Frame:

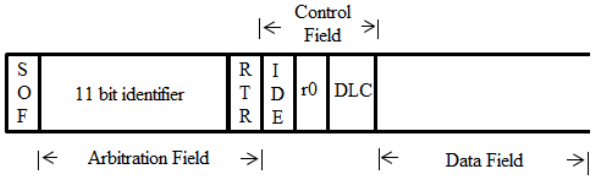


Figure 1: CAN-Standard Format

CAN 2.0B Data Frame:

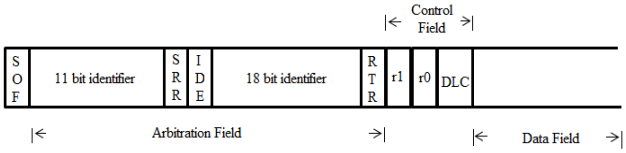


Figure 2: CAN-Extended Format

CAN is a multi-master serial bus standard for connecting Electronic Control Units [ECUs] also known as nodes. Two or more nodes are required on the CAN network to communicate. The complexity of the node can range from a simple I/O device up to an embedded computer with a CAN interface and sophisticated software. The node may also be a gateway allowing a standard computer to communicate over a USB or Ethernet port to the devices on a CAN network.All nodes are connected to each other through a two wire bus. The wires are 120 Ω nominal twisted pair.Message IDs must be unique on a single CAN bus, otherwise two nodes would continue transmission beyond the end of the arbitration field (ID) causing an error.

To ensure enough transitions to maintain synchronization, a bit of opposite polarity is inserted after five consecutive bits of the same polarity. This practice is called bit stuffing.CAN uses a ”Non Return to Zero” (NRZ) coding technique. Since there is no synchronization signalling involved in NRZ a drift in the receiver’s clock can occur when a long sequence of identical bits has been transmitted; this can result in message corruption. To avoid the possibility of this scenario, the CAN communication protocol at the physical level uses a bit-stuffing mechanism which operates as follows: after five consecutive identical bits have been transmitted in a given frame, the sending node adds, an additional bit, of the opposite polarity. When five or more consecutive bits of the same polarity are to be transmitted, a ”stuff” bit of the opposite polarity is inserted by the transmitting hardware, and subsequently removed by the receiver hardware. Six consecutive bits of the same type 111111 or 000000 are considered as an error else an error or overload frame . According to the CAN standard the total number of bits after bit stuffing will be:

Total bits after bit stuffing = $8n+47+\lfloor (34+8n-1)/4 \rfloor$ …. (1)

n- Number of bytes.

Bit stuffing can introduce a delay of up to 24 bit time (worst case).

Today the CAN bus is also used as a fieldbus in general automation environments, primarily due to the low cost of some CAN controllers and processors. CAN is being widely used in passenger cars, buses, factory automation, work machines, agriculture, forestry and mining applications. Medical equipment manufacturer’s use CAN as an embedded network in medical devices. In fact, some hospitals use CAN to manage complete operating rooms. Hospitals control operating room components such as lights, tables, cameras, X-ray machines, and patient beds with CAN-based systems. Lifts and escalators use embedded CAN networks, and hospitals use the CAN protocol to link lift devices, such as panels, controllers, doors, and light barriers, to each other

and control them. CAN also is used in nonindustrial applications such as laboratory equipment, sports cameras, telescopes, automatic doors, and even coffee machines.The applications for CAN are increasing all the time.

Literature Review

Bit stuffing using message manipulation

Using message manipulation method the bitmask value is applied to the data. The data will be de-stuffed in the receiving end with the same bitmask value. While applying this method the bit stuffing is reduced. When applying it for the worst case scenario like the data which is same as the bitmask value the transmitted data will be all 0's.Again it will lead to the bit-stuffing of 12 bits. On the contrary the complement of the bitmask value is sent the transmitted data will be all 1's which will also include the bit stuffing. This method fails to concentrate on worst case scenarios. To overcome this an investigation is done and a new method is proposed.

No. of stuff bits: 12

Figure 3: Bit-stuffing in CAN data using message manipulation

Total number of bits after bit stuffing for the worst case scenario is 135.The largest frame takes 135 bit times to send the data.

Eight-To-Eleven Modulation (EEM)

Eight-to-Eleven Modulation (EEM) is another type of X-to-Y modulation where "X" equals to 8 (the number of bits per byte) and "Y" equals to 11(the number of encoded data bits).After adding the extra stuff bit the total number of stuff bit will be three for a set of 8 bit data. So the total number of bit encoded bit will be 11 bit. In the 11 bit encoded format of the 8 bit input data, where we have one stuff bit at the middle of the input data sequence, one near most significant bit and one near least significant bit.This method includes 3 stuf bits per byte.It increases the length of the frame. To overcome this a new method is introduced.

Selective XORing

In selective XORing method the bitmask is applied for the data field in the selective CAN frames. The bit mask values used are BM1:[55 55 55 55 55 55 55 55] and BM2:[91 91 91 91 91 91 91 91]. The bitmask is applied for the CAN frames other than the worst case scenarios as mentioned above. If the data is same as the bitmask or the complement of the bitmask it will be transmitted without masking. The algorithm explains this method.

BM1: Bitmask1
BM2: Bitmask2

Algorithm for CAN Data transmission

```
if( d==0)
{
d xor BM2
}
    elseif (d==BM2)
{
          d | 1
}
else
{Tx=d}
 elseif(d==BM1)
{
Tx=data}
else
{
d xor BM1
}
if (d==~BMS1)
{
Tx=d
}
else{
            Tx = XORed data
}
```

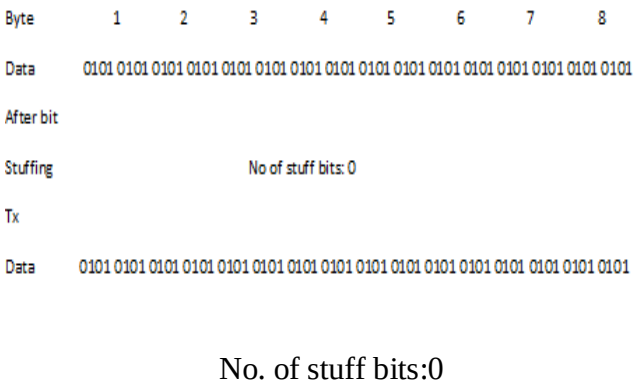
When the transmitted data is all 0's it is XORed with BM2 .If the transmitted data is BM2 it is ORed with1.Otherwise it is XORed with BM1.If the transmitted data is equal to BM1 it is transmitted as such without masking.

Algorithm for CAN DATA reception

```
if(R==BM1)
{
receive data
}
```

```
 elseif(r=C4)
{
X= r|1
X xor BM1
}
else
{
Y= r  xor BM1
If(y==BM2)
{
Y xor BM2
}
elseif(y==0)
{
Y xor 170 and transmit
}
        else {receive the data
}
}
```

In the receiving end also the bit masking should be done to destuff the bits. When the received data is same as BM1 receive the data as such. If the received value is equal to [C4 C4 C4 C4 C4 C4 C4 C4] first the received value is ORed with the data and then it is XORed with BM1.If the received data is equal to BM2 it will be XORed with the BM2 to decode the data. If the value is equal to 0 then it will be XORed with [170 170 170 170 170 170 170 170].Otherwise the data will be received as such.



No. of stuff bits:0

Figure 4: Bit-stuffing in CAN data using selective XORing

Implementation using MATLAB simulink

This method is implemented in Simulink model using MATLAB R2012b.This version consists of vehicle network toolbox. It contains the blocks CAN configuration , CAN transmit, CAN receive, CAN pack and CAN un pack. Using these blocks the algorithm is implemented and it is verified for the worst case scenarios.

The Vehicle Network Toolbox block library is a tool for simulating message traffic ona CAN network, as well for using the CAN bus to send and receive messages. We can use blocks from the block library with blocks from other Simulink libraries to createsophisticated models.

CAN Configuration - Configure the settings of a CAN device.

CAN Transmit -Transmit CAN messages to a CAN bus.

CAN Receive - Receive CAN messages from a CAN bus.

CAN Pack - Pack signals into a CAN message.

CAN Unpack -Unpack signals from a CAN message.

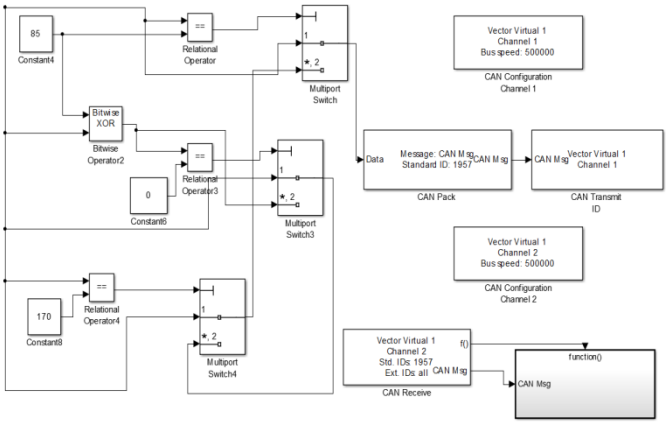


Figure 5; Simulink model for selective XORing

The model is created using these blocks. Then the model is run. Based on the execution the transmitted data is noted.

Results and Discussion

This method is analyzed for the following worst case scenarios. This table shows that without using the selective XORing the number of stuff bits for the worst case scenarios 1 and 2 is 24. The stuff bits using selective XORing are 8 which is reduced from 24.Total number of bits after bit stuffing for the worst case scenario using this method is (64+47+8=119).The largest frame takes 119 bit times to send the data using this method. This is reduced to 119 bit times from 135 bit times.

Table 1: Simulink result for stuff bits with selective XORing Vs.stuff bits without selective XORing

Worst Case Scenarios	Data (hex)	Tx data (hex)	Stuff bits without selective XORing	Stuff bits with selective XORing	Rx data(hex)
WCS1	[55 55 55 55 55 55 55 55]	[55 55 55 55 55 55 55 55]	24	8	[55 55 55 55 55 55 55 55]
WCS2	[AA AA AA AA AA AA AA AA]	[AA AA AA AA AA AA AA AA]	24	8	[AA AA AA AA AA AA AA AA]
WCS3	[1 1 1 1 1 1 1 1]	[54 54 54 54 54 54 54 54]	8	8	[1 1 1 1 1 1 1 1]
WCS4	[0 0 0 0 0 0 0 0]	[C4 C4 C4 C4 C4 C4 C4 C4]	8	8	[0 0 0 0 0 0 0 0]
WCS5	[91 91 91 91 91 91 91 91]	[C6 C6 C6 C6 C6 C6 C6 C6]	8	8	[91 91 91 91 91 91 91 91]
WCS6	[6E 6E 6E 6E 6E 6E 6E 6E]	[3B 3B 3B 3B 3B 3B 3B 3B]	8	8	[6E 6E 6E 6E 6E 6E 6E 6E]

WCS: Worst Case Scenario

The graph depicts the Probability of bit stuffing for different worst case scenarios. The probability of including the stuff bits is reduced from 1 to 0.3 using this method.

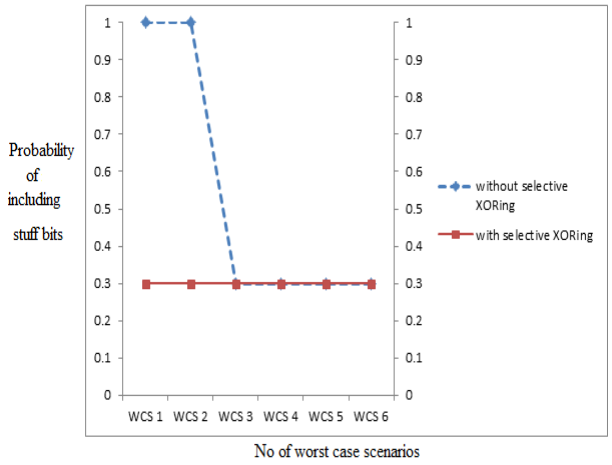


Figure 6: Probability of bit stuffing for worst case scenarios

This result shows that the probability of including the stuff bits using selective XORing is reduced and it will reducethe transmission delay.

Conclusion

In this paper we have presented a new approach to reduce bit stuffing for worst case scenarios. We achieved the accuracy in the Simulink modeling by taking worst case

scenarios into consideration. This allowed us to reduce the frame size used when performing bit stuffing analysis of the CAN bus. It is observed that worst case number of stuff bits using selective XORing is 8 compared to the worst case of 24 bits derived by message manipulation method.On a more detailed level, we will investigate the effects of bit stuffing in the control fields of CAN frames.

References

[1] T. Nolte, H. Hansson, and C. Norstrom. Minimizing CAN Response-Time Analysis Jitter by Message Manipulation. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02), pages 197–206, September

[2] T.Nolte, H. Hansson, and C. Norstrom. Using Bit-Stuffing Distributions in CAN Analysis. IEEE/IEE Real-Time Embedded Workshop (RTES'01), December 2001.

[3] K. W. Tindell, A. Burns, and A. J. Wellings. Calculating Controller Area Network (CAN) Message Response Times. Control Engineering Practice, 3(8):1163–1169, 1995.

[4] Florian Hartwich, Armin Bassemir,"The configuration of the CAN Bit timing",6th international CAN conference,Turin.

[5] J. Xu and D. L. Parnas. Priority scheduling versus preruntime scheduling. Real-Time Systems Journal, 18(1), January 2000.

[6] I. S. O. (ISO). Road Vehicles- Interchange of digital information - Controller Area Network (CAN) for high-speed communication. ISO Standard-11898, Nov 1993

[7] I. Broster and A. Burns. Timely Use of the CAN Protocol in Critical Hard Real-Time Systems With Faults. Proceedings of the Euromicro Conference on Real-Time Systems, June 2001.

[8] A. Burns. Preemptive Priority Based Scheduling: An Appropriate Engineering Approach. Technical Report YCS 214, University of York, 1993.

[9] Bosch (1991), Robert Bosch GmbH "CAN Specification Version 2.0".

[10] Vehicle Network Toolbox,User's guide,Mathworks.