Fall 2007

Performance improvements of automobile communication protocols in electromagnetic interference environments

Fei Ren

PERFORMANCE IMPROVEMENTS OF AUTOMOBILE COMMUNICATION PROTOCOLS IN ELECTROMAGNETIC INTERFERENCE ENVIRONMENTS

by

FEI REN

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI–ROLLA

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

2007

Approved by

ABSTRACT

Electromagnetic Interference (EMI) is frequently encountered in automobile communication systems due to a large number of inductive nodes used in these systems. This thesis investigates the effects of EMI on two types of automobile communication systems, the Controller Area Network (CAN) and the FlexRay. It also proposes a modified Automatic Repeat reQuest (ARQ) scheme to improve the communication performances in EMI environments.

A CAN hardware testbed was built to study the effects of EMI. The testbed consists of a four-node CAN network and an EMI generator. The CAN communication nodes were implemented by Intel C8051 development boards and the EMI pulses were generated by a magnetic relay commonly used in vehicles. The effects of EMI were measured in several configurations including two and four CAN nodes network, unshielded and shielded bus cables in different lengths, and various data rates. Measurement results were recorded using oscilloscopes and analyzed using Matlab programs. It was found that the coupling of EMI in the bus is typically additive with 3 – 5 μs in duration and causes burst errors. The burst errors usually result in retransmission but occasionally cause communication halt.

The thesis further proposes a modified ARQ scheme for CAN and FlexRay to combat EMI-induced errors. Current CAN and FlexRay use Cyclic Redundancy Check (CRC) codes for error detection and ARQ for retransmission. The Modified ARQ scheme adds an error-correction code to encode the data field and modifies the CRC code to only encode the header field. Therefore, no retransmission is needed when errors only corrupt the data field. This reduces the probability of retransmission by the ratio of the data field length and the frame length. The proposed scheme imposes minimal change of the signaling structure over the original CAN and FlexRay protocols.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Yahong Rosa Zheng, for her guidance and encouragement through out my MS program. She not only taught me the knowledge in communications, but also trained me the skills to conduct my research. I am grateful to her for giving me the opportunity to work with her. In research meetings, she always gave me clear guidance and helped me when I met problems. She also spent a lot of effort to help me learn English writing skills through weekly reports, the conference paper, and the thesis. I am thankful for what she has done for me and I am grateful that I can continue my Ph.D study under her supervision.

I would like to thank my committee members Dr. Jagannathan Sarangapani and Dr. Maggie Cheng for their timely support and invaluable help. I would like to thank them for reviewing my thesis and offering me important suggestions.

I would like to convey my sincere thanks to Caterpillar Inc. for giving me the opportunity to work on the CAN bus project through the CAT University Challenge Program. I would like to thank Dr. Thomas P. Van Doren and Dr. Maciej Zawodniok for their guidance in CAN testbed measurements. I would like to thank my research team member Krishna C. Emani for his help in my research and Sarat Kumar Chitneni for helping with thesis revising. I would like to thank the other research team members, Jian Zhang, Yuan Liu, Tiange Shao, and Xin Liu, for their help and support.

I would like to express special thanks to my parents Mr. Gaohong Ren and Mrs. Chunzhuo Jin for their blessings and good wishes. They give me invaluable support in my study and my live.

I would like to thank every person I met over the period of my graduate program and all those who helped me in this period.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1. INTRODUCTION TO THE PROBLEM

The automobile communication networks studied in this thesis are wired communication networks that interconnect components inside a vehicle. The communication networks take charge of data exchange among sensors and controllers, which control engine operation, the transmission chain, anti-lock brakes, and body control modules (lights, doors, horn). Currently, there are several popular automobile communication protocols including the Local Interconnect Network (LIN), the Controller Area Network (CAN), FlexRay, and the Media Oriented Systems Transport (MOST). The LIN is a low-cost, low-data-rate system usually used as a cheap sub-network of CAN [1]. The CAN is a low-cost, medium data-rate system mostly utilized as the backbone network in vehicles including cars, buses, off-high-way trucks, and boats [2]. FlexRay is a newly developed high data-rate system with safety-critical features [3] and it is expected to replace CAN in the near future [4]. The MOST is a very-high data-rate networking standard intended for interconnecting multimedia components in automobiles and other vehicles [5]. The data rate comparison is summarized in Figure 1.1.

The biggest challenges in automobile communication networks are Electromagnetic Interference (EMI), serviceability, and cost constraints. EMI is primarily caused by a large number of inductive loads used in automobiles. Strong EMI pulses often couple into the communication system through wiring and cause burst errors. EMI from other wireless applications, such as cellular phone systems, citizen band radio, weather radar, etc., also contribute to single-bit or burst errors in automobile networks. It is the major reason of performance degradation which often increases the number of retransmission, reduces network efficiency, and occasionally causes total halt in communications [6]. Using shielded cables helps to reduce the effect of EMI significantly. However, it also reduces the serviceability and increases the cost significantly. Cable shielding needs to be carefully grounded at both ends of the communication link which is often vulnerable in practical systems. Replacing and repairing shielded cables are also more costly, inconvenient, and unreliable. Therefore, shielded cable is considered as the last solution by the automobile industry.

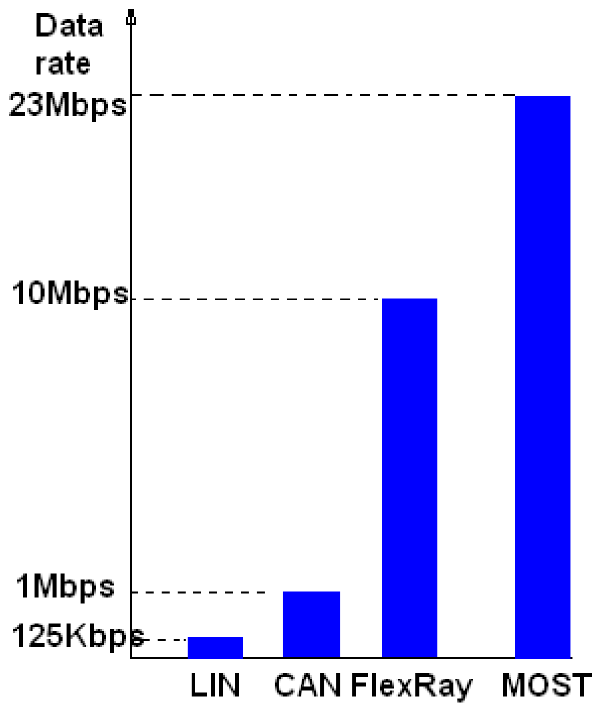


Figure 1.1 The data rates of automobile communication networks

This thesis focuses on the study of EMI effects on the CAN and FlexRay protocols and proposes a modified Automatic Repeat reQuest (ARQ) scheme to improve the communication performances in EMI environments. Current CAN and FlexRay systems employ several mechanisms to combat EMI for reliable communications. In both CAN and FlexRay, the differential bus structure helps to reduce effects of EMI on signal waveforms. The error-detection method implemented by Cyclic Redundancy Check (CRC) codes is employed to detect corrupted frames caused by EMI [7]. Specially for the CAN protocol, bit-stuffing is also used to increase the error detection capability. When an error frame is detected, the stop-and-wait ARQ scheme is employed to retransmit corrupted frames. In FlexRay, the dynamic segment of the communication cycle uses the structure of micro-tick time slots to reduce collision and improve network efficiency. FlexRay also employs a dual-bus structure by using a redundant channel for safety-critical applications.

With the increased level of automation and complexity in cars and off-road machinery, the number of sensors, Electronic Control Module (ECM), and inductive loads has been increased dramatically over the recent years. This means that EMI becomes more serious and requirements for automobile communication networks become more stringent. Therefore, there is a need to investigate the EMI effects on CAN and FlexRay protocols in a realistic environment and research for alternative methods for error mitigation. A fellow MS student in the research team, Mr. Krishna C. Emani, proposed a couple of Hybrid ARQ schemes for the CAN protocol [8]. The Hybrid ARQ (HARQ) schemes employ an outer code to encode each CAN data frame for error correction purposes. The outer code may be transmitted together with the original CAN frame (Type I HARQ) or may be detached from the original frame (Type-II H-ARQ). The Type-II HARQ only transmit the error correction codes when requested by the receiver. It has been shown that the HARQ schemes improve the CAN performance by reducing the number of retransmission at the expenses of increased overhead (for Type-I HARQ) or increased latency (for Type-II HARQ). This thesis proposes a different approach, called the modified ARQ for both CAN and FlexRay, which employs separate coding schemes for the data field and header fields.

1.2. APPROACH AND CONTRIBUTION

This thesis built a hardware testbed for CAN communications and investigated the effects of EMI on CAN and FlexRay. It then proposed a modified Automatic Repeat reQuest (ARQ) scheme to improve the communication performances in EMI environments.

A CAN hardware testbed was built consisting of a four-node CAN network and an EMI generator. The CAN communication nodes were implemented by Intel C8051f040 Mixed Signal ISP Flash MCU board from Silicon Laboratories. The CAN nodes were connected by shielded or unshielded DB9 serial cables with different lengths. The cables are placed closely coupled with the EMI generator cables to study the EMI effects. The EMI pulses were generated by a magnetic relay connected with a 24 V battery and a manual switch. The magnetic relay provided by Caterpillar Inc. was an inductive load used in automobiles and off-road machinery. The effects of EMI were measured in several configurations including two and four CAN nodes network, unshielded and shielded bus cables in different lengths, and various data rates. Measurement results were recorded using oscilloscopes and analyzed using Matlab programs. Measurement results showed that the coupling of EMI in the bus is typically additive and impulsive. Their duration was 3 – 5 μs. When using unshielded cables, EMI usually cause burst errors each time the switch was turned on and off. The number of burst bits depended on the operation of the switch and the data rate. The burst errors usually result in retransmission but occasionally cause communication halt.

The thesis further proposed a modified ARQ scheme for CAN and FlexRay to combat EMI-induced errors. The Modified ARQ scheme adds an error-correction code to encode the data field and modifies the CRC code to only encode the header field. The error-correction codes used in CAN were the (63, k) BCH codes with different ($k = 7, 16, 24$ and 32) depending on the length of the raw data field. For FlexRay, the payload segment is as large as 512 and the error correction codes were chosen to be (n = 511,k), also with a varying k in the range of $112, 184, 304$ and 376 depending on the raw data length. This method is of particular advantage in communicating real-time control signals because the raw data are often very short and hard real-time control is required. With these powerful error correction coding schemes, no retransmission is needed when errors only corrupt the data field. This reduces the probability of retransmission by the ratio of the data field length and the frame length. The modified ARQ also improves the error-detection capability over the original schemes because the header contains less bits to be encoded by the same number of CRC bits. The proposed scheme imposes minimal change of the signaling structure over the original CAN and FlexRay protocols.

Matlab simulations were conducted to evaluate the performance of the modified ARQ schemes for CAN and FlexRay. Burst errors with fixed length were generated and added into CAN and FlexRay data frames. Then Error correction was performed in the receiver. The bit error rate, the frame error rate, and the number of retransmissions were computed. Simulation results show that the modified ARQ scheme reduces the retransmission of CAN frames by 44% and reduces FlexRay retransmission by 66%.

A paper was published and presented at an IEEE reference conference: Fei Ren, Y.R. Zheng, Maciej Zawodniok, and J. Sarangapani, “EFFECTS OF ELECTROMAGNETIC INTERFERENCE ON CONTROL AREA NETWORK PERFORMANCE”, IEEE Region 5 Technical, Professional, and Student Conference (TPSC), 2007.

2. BACKGROUND

This section depicts backgrounds of CAN, FlexRay, the ARQ scheme and BCH codes. It includes four subsections: the CAN protocol, the FlexRay protocol, the ARQ scheme, and the BCH code. The CAN protocol subsection mainly introduces CAN basic features, CAN frame structure, CAN frame transmission principles and CAN error handling mechanisms for EMI. In this thesis, CAN protocol is employed to build hardware test bed. Matlab simulations implement the Modified ARQ scheme on CAN. The FlexRay protocol subsection mainly introduces FlexRay basic features, FlexRay frame structure, FlexRay frame transmission principles and FlexRay error handling mechanisms for EMI. In this thesis, Matlab simulations implement the Modified ARQ scheme on FlexRay. Performances improvement of Modified ARQ FlexRay is compared to that of Modified ARQ CAN. The ARQ scheme subsection introduces the basic principles of the ARQ scheme and the Stop-and-wait ARQ scheme in CAN. The ARQ scheme is employed in current CAN for increasing communication reliability. Proposed Modified ARQ scheme employs the error-correction codes to improve the ARQ scheme. The BCH code subsection introduces basic knowledge of BCH coeds, encoding and decoding of BCH codes and implementation of BCH codes in Matlab simulations. The BCH code is used as the error-correction code in the Modified ARQ scheme. Its error-correction capability can adjust to deal with specified extent errors.

2.1. THE CAN PROTOCOL

2.1.1. CAN Basic Features. CAN is a broadcast, differential serial bus standard, originally developed in the 1980s by Robert Bosch GmbH, for connecting electronic control units (ECU). CAN was specifically designed to be robust in electromagnetically noisy environments and can utilize a differential balanced line like RS-485. It can be even more robust against noise if twisted pair wire is used. Although initially created for automotive purposes (as a vehicle bus), nowadays it is used in many embedded control applications (e.g., industrial) that may be subject to noise [9].

The maximum bit rate of a CAN bus, according to the standard, is 1 Mbps. The CAN bus, with the bit rate reaching to 1 Mbps, is called high-speed CAN bus and applied in vehicle automation. The CAN bus, with the bit rate not exceeding 128 kbps, is called low-speed CAN bus and may applied in area not requiring high bit rates (i.e, temperature control in a building).

Current CAN protocol is standardized in ISO 11898. This standard describes mainly the data link layer - composed of the Logical Link Control (LLC) sublayer and the Media Access Control (MAC) sublayer - and some aspects of the physical layer of the OSI Reference Model. All the other protocol layers are left to the network designer's choice [2].

2.1.2. CAN Frame Structure. The frame is the basic unit for CAN data exchange. The maximum frame's length of CAN is 128 bits. There are two CAN frame standards whose difference is the length of the Identifier in the Arbitration Field of CAN frames. Originally, the CAN standard defined the length of the Identifier in the Arbitration Field to 11 bits. Later on, customer demand forced an extension of the standard. The new format is often called Extended CAN and allows no less than 29 bits in the Identifier. To differentiate between the two frame types, a reserved bit in the Control Field was used.

The two standards are formally called 2.0A (with 11-bit Identifiers only), 2.0B (extended version with the full 29-bit Identifiers or the 11-bit, they can be mixed). New CAN controllers today are usually of the 2.0B type [10]. In this thesis, both hardware experiments and Matlab simulations use CAN 2.0B (29-bit Identifiers).

There are four different frame types on a CAN bus: the data frame, the remote frame, the error frame, and the overload frame. The Data Frame is the most common message type.

The data frame is the most used frame in CAN applications. The frame structure of one CAN data frame is given in Figure 2.1. It comprises four major parts (a few not important fields and functions are omitted): the Arbitration Field, the Data Field, the CRC Field and an Acknowledgement Slot. The Arbitration Field determines the priority of the message when two or more nodes are contending for the bus. It contains: for CAN 2.0A, an 11-bit Identifier and one bit, the RTR bit (Remote Transmission Request), which is dominant for data frames; for CAN 2.0B, a 29-bit Identifier, which also contains two recessive bits: SRR (Substitute remote request) and IDE (Identifier Extension Bit) and the RTR bit. The Data Field contains zero to eight bytes of user data. The CRC Field contains a 15 bits CRC

Start of frame	Arbitration field	Control field	Data field	CRC field	Ack field	End of frame
(1 bits)	(31 bits)	(7 bits)	(64 bits)	(15 bits)	(3 bits)	(7 bits)

Figure 2.1 The CAN data frame structure

calculated on the Start of Frame, the Arbitration Field, the Control Field and the Data field. This CRC is computed using the generator polynomial below:

$$X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1$$

The error-detection capability of the CRC is at most 5 bits single errors or 15 bits burst errors. The Acknowledgement Slot is employed as the acknowledgement bit. Any CAN receiver that has been able to correctly receive the frame sends an Acknowledgement bit at the end of each frame. The CAN transmitter checks for the presence of the Acknowledge bit and retransmits the frame if no acknowledge is detected. But the presence of an Acknowledgement Bit on the bus does not mean that any of the intended addressees has received the message. The only thing known is that one or more nodes on the bus have received it correctly.

The remote frame is just like the data frame, with only two important differences: it is explicitly marked as a remote frame (the RTR bit in the Arbitration Field is recessive), and there is no Data Field. The intended purpose of the remote frame is to solicit the transmission of the corresponding data frame. For example, one node transmits a remote frame with the Arbitration Field set to 123, then another node , if properly initialized, might respond with a data frame with the Arbitration Field also set to 123. remote frames can be used to implement a type of request-response type of bus traffic management. In practice, however, the remote frame is little used. The CAN standard does not prescribe the behavior outlined here. Most CAN controllers can be programmed either to automatically respond to a remote frame, or to notify the local CPU instead.

The error frame is a special frame that violates the framing rules of a CAN frame. It is transmitted when a node detects a fault and will cause all other nodes to detect a fault - so they will send error frames, too. The transmitter will then automatically try to retransmit the frame. There is an elaborate scheme of error counters that ensures that a node can't destroy the bus traffic by repeatedly transmitting error frames. The error frame consists of an Error Flag, which is 6 bits of the same value (thus violating the bit-stuffing rule) and an Error Delimiter, which is 8 recessive bits. The Error Delimiter provides some space in which the other nodes on the bus can send their Error Flags when they detect the first Error Flag.

The overload frame is mentioned here just for completeness. It is very similar to the error frame with regard to the format and it is transmitted by a node that becomes too busy. The overload frame is not used very often, as today's CAN controllers are clever enough not to use it. In fact, the only controller that will generate overload frames is the now obsolete 82526.

2.1.3. CAN Frame Transmission Principles. The frame arbitration (the process in which two or more CAN controllers agree on who is to use the bus) is of great importance for the really available bandwidth for data transmission. Any CAN controller may start a transmission when it has detected an idle bus. This may result in two or more controllers starting a frame (almost) at the same time. The conflict is resolved in the following way. The transmitting nodes monitor the bus while they are sending. If a node detects a dominant level when it is sending a recessive level itself, it will immediately quit the arbitration process and become a receiver instead. The arbitration is performed over the whole Arbitration Field and when that field has been sent, exactly one transmitter is left on the bus. This node continues the transmission as if nothing had happened. The other potential transmitters will try to retransmit their messages when the bus becomes available next time. No time is lost in the arbitration process.

An important condition for this bit-wise arbitration to succeed is that no two nodes may transmit the same Arbitration Field. There is one exception to this rule: if the message contains no data, then any node may transmit that message. Since the bus is wired-and and a Dominant bit is logically 0, it follows that the message with the numerically lowest Arbitration Field will win the arbitration.

A node wins the arbitration and proceeds with the frame transmission. When the time comes for acknowledging. If the frame is correctly received by one or more nodes (maybe not the intended address node), nodes will send a dominant bit during the ACK slot. Then the transmitter will sense the dominant bit, and transmission next frame if any. If no node correctly receive the frame, none will send a dominant bit during the ACK slot, so the transmitter will sense an ACK error, send an error flag, increase its transmit error counter by 8 and start a retransmission.

Another features of CAN frame transmission is that there is no explicit address in the CAN frames. Each CAN controller will pick up all traffic on the bus, and using a combination of hardware filters and software, determine if the message is “interesting” or not. In fact, there is no notion of frame addresses in CAN. Instead, the contents of the frames are identified by an identifier, which is present somewhere in the message. CAN messages are said to be “contents-addressed”.

The contents of the Arbitration Field are used to determine the message’s priority on the bus. All CAN controllers will also use the whole (some will use just a part) of the Arbitration Field as a key in the hardware filtration process. The Standard does not say that the Arbitration Field must be used as a message identifier. It’s nevertheless a very common usage.

2.1.4. CAN Error Handling Mechanisms for EMI. Several error handling mechanisms related to combat EMI is built in the CAN protocol. In the physical layer of CAN, differential signal is employed to combat EMI. Differential signaling is a method of transmitting information electrically by means of two complementary signals sent on two separate wires. It can combat EMI is not actually due to differential signalling itself, but to the common practice of transmitting differential signals on balanced lines . A balanced line reduces the noise on a connection by rejecting common-mode interference. Its two wires are routed in parallel so that they receive the same interference. They also have the same impedance to ground, so the interfering fields or currents induce the same voltage in both wires. Since the receiver responds only to the difference between the wires, it is not influenced by the induced noise voltage [11].

Besides differential signaling, the CRC and Bit stuffing work to detect errors, including those caused by EMI. Each frame features a 15-bit CRC (at most detect 5 bits errors), and any node that detects a different CRC in the frame than what it has calculated itself will detect CRC errors. Bit Stuffing works when five consecutive bits of the same level have been transmitted by a node. It will add a sixth bit of the opposite level to the outgoing bit stream.

The receivers will remove this extra bit. This is done to avoid excessive DC components on the bus, but it also gives the receivers an extra opportunity to detect errors (it may be caused by EMI): if more than five consecutive bits of the same level occurs on the bus, a Stuff Error is signaled [9].

Every CAN node along a bus will try to detect errors within a frame, so that the transmitter can retransmit an erroneous frame. If an error is found, the discovering node will transmit an Error Flag, thus destroying the bus traffic. The other nodes will detect the error caused by the Error Flag (if they haven't already detected the original error) and take appropriate action, i.e. discard the current frame. The CAN node can automatically retransmit erroneous frame when errors have occurred.

Each node maintains two error counters: the Transmit Error Counter and the Receive Error Counter. There are several rules governing how these counters are incremented and/or decremented. In essence, a transmitter detecting a fault increments its Transmit Error Counter faster than the listening nodes will increment their Receive Error Counter. This is because there is a good chance that it is the transmitter who is at fault! When any Error Counter raises over a certain value, the node will first become "error passive", that is, it will not actively destroy the bus traffic when it detects an error, and then "bus off", which means that the node doesn't participate in the bus traffic at all. Using the error counters, a CAN node cannot only detect faults but also perform error confinement [10]. But CAN does not have any mechanisms for error-correction.

2.2. THE FLEXRAY PROTOCOL

2.2.1. FlexRay Basic Features. FlexRay is a new automative network protocol developed by an industry consortium with four founding members (BMW, Daimler-Chrysler, Philips and Freescale) in the year 2000. It has more than 130 members covering the whole globe by the end of 2005 [12]. FlexRay is designed to develop the faster and safer network for transferring data between sensors and controller on an automobile and ensuring communication reliability simultaneously. The first public release of FlexRay specification is in January 2004, and the version is 2.0. Now, the latest version of FlexRay is "FlexRay Communications System Protocol Specification Version 2.1 Revision A", which is released in December 2005. It is widely used as a basis for the implementation of semiconductor devices and tools [13]. The first vehicle employed FlexRay is the “2006 BMW X5”, though FlexRay only was used for the pneumatic damping system. Full use of FlexRay in other high-speed control applications of vehicles such as advanced powertrain, anti-brake system, and by-wire system are expected in 2008.

The three top design objectives of FlexRay standardization are: high bit rate, deterministic communication, and fault-tolerant communication. FlexRay supports high bit rate up to 10 Mbps in one channel. It also supports two channels communication. The second optional channel can be used to increase the total bit rate up to roughly 20 Mbps, or be used as the redundant channel to provide fault toleration. Because two channels are independent in communication wires, additional communication wires need to be provided to support the second one. It means that in the same time, the second channel can transmit totally different data from that of first one.

Deterministic communication means that FlexRay uses a hybrid MAC scheme to avoid uncertain results due to competition of nodes trying to get bus. In nodes, frame transmission can be triggered either dynamically, in response to an event (event-driven), or statically, at predetermined moments in time (time-driven). Therefore, on one hand, there are protocols that schedule frame transmissions statically, based on the progression of time, such as the TTCAN, and Time-Triggered Protocol (TTP). A drawback of such protocols is their lack of flexibility. On the other hand, there are communication protocols where frame transmission scheduling is performed dynamically, such as CAN or Byteflight. Their drawback is less reliability and node transmissions may collide each other. FlexRay employed a hybrid MAC scheme and allows the sharing of the bus among event-driven and time-driven messages [14]. It gets advantages from both and avoids uncertain due to competition of bus. The FlexRay hybrid MAC scheme is described in detail in Section 2.2.3 in this thesis.

The fault-tolerant feature are implement by three aspects: two basic mechanisms responding to errors, bus-guardians in the physical layer and the flexibility of interconnection architecture. Two basic mechanisms of FlexRay to deal with various errors are described in Section 2.2.4. Bus-guardians are optional hardware equipment in each node, or the central node in active star topology. They are used to mainly detect errors that nodes transmitting untimely frames in static segment slots (refers to the charpter 5 in [15]). Bus-guardians do not transmit any data to channels, but only receive. In FlexRay communication cycle, the static segment slot is predefined and used for real-time critical applications, and bus-guardians check for the frame transmission times during this period to prevent communication from errors [16]. Another important fault-tolerant feature of FlexRay is the flexibility of the interconnection architecture. While a simple system with no fault-tolerance can use a bus architecture and only one channel, there are various levels of more redundancy. A dual channel system replicates the physical network and tolerates one faulty channel. At even higher cost, star and multiple-star topologies can increase the usable channel capacity and provide error containment. An overview of architectures in FlexRay along with design criteria for each is given in Figure 2.2 [15]. It should be noted that combinations of star and bus architectures are possible in FlexRay [13].

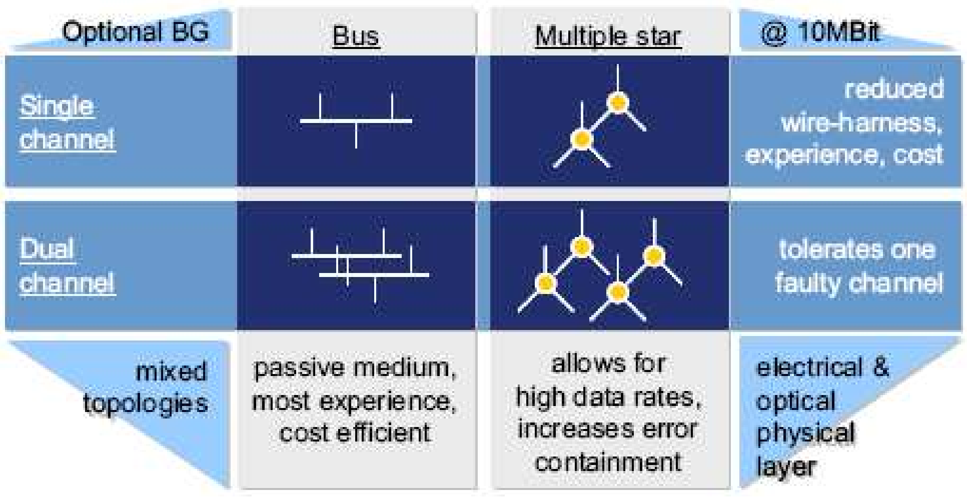

Optional BG
Bus
Multiple star
@ 10MBit
Single channel
reduced wire-harness, experience, cost
Dual channel
tolerates one faulty channel
mixed topologies
passive medium, most experience, cost efficient
allows for high data rates, increases error containment
electrical & optical physical layer

Figure 2.2 The topologies of FlexRay

2.2.2. FlexRay Frame Structure. FlexRay has only one type of frame, the FlexRay data frame, which is easier than four types of frames in CAN. An overview of the FlexRay frame format is given in Figure 2.3 [9]. One FlexRay data frame consists of three segments, a Header segment, a Payload segment and a Trailer segment. The FlexRay header segment consists of 5 bytes. These bytes contain the reserved bit, the payload preamble indicator, the null frame indicator, the sync frame indicator, the startup frame indicator, the frame ID, the payload length, the header CRC, and the cycle count.

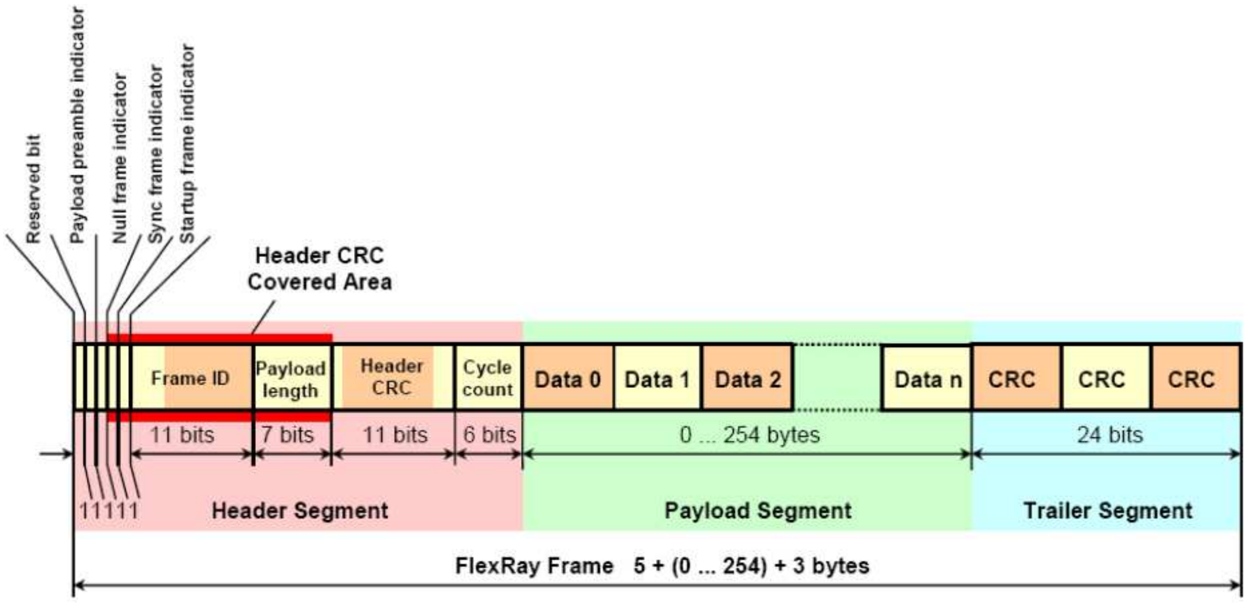


Figure 2.3 The FlexRay data frame structure

These are three main fields in the Header segment: the frame ID, the payload length and the header CRC. The frame ID defines the slot (refers to the chapter 5 in [15]) in which the frame should be transmitted. A frame ID is used no more than once on each channel in a communication cycle. Each frame that may be transmitted in a cluster has a frame ID assigned to it. The frame ID ranges from 1 to 2047. The frame ID 0 is an invalid frame ID42. The payload length field is used to indicate the size of the payload segment. The payload segment size is encoded in this field by setting it to the number of payload data bytes divided by two (e.g., a frame that contains a payload segment consisting of 72 bytes would be sent with the payload length set to 36). The header CRC contains a CRC that is computed over the sync frame indicator, the startup frame indicator, the frame ID, and the payload length. The communication controller (CC) shall not calculate the header CRC for a transmitted frame. The header CRC of transmitted frames is computed offline and provided to the CC by means of configuration (i.e., it is not computed by the transmitting CC). The CC shall calculate the header CRC of a received frame in order to check that the CRC is correct. The CRC is computed in the same manner for all configured channels. The CRC polynomial shall be:

$X^{11} + X^9 + X^8 + X^7 + X^2 + 1$

This 11 bits CRC polynomial generates a (31, 20) BCH code that has a minimum Hamming distance of 6. It is employed to detected errors up to 5 bits.

The FlexRay payload segment contains 0 to 254 bytes (0 to 127 two-byte words) of data. Because the payload length contains the number of two-byte words, the payload segment contains an even number of bytes. The bytes of the payload segment are identified numerically, starting at 0 for the first byte after the header segment and increasing by one with each subsequent byte. The individual bytes are referred to as “Data 0”, “Data 1”, “Data 2”, etc., with “Data 0” being the first byte of the payload segment, “Data 1” being the second byte, etc.

The FlexRay trailer segment contains a single field, a 24 bits CRC for the frame. The Frame CRC field contains a cyclic redundancy check code (CRC) computed over the header segment and the payload segment of the frame. The computation includes all fields in these segments. The CRC is computed using the same generator polynomial on both channels. The CRC polynomial shall be:

$X^{24}+X^{22}+X^{20}+X^{19}+X^{18}+X^{16}+X^{14}+X^{13}+X^{11}+X^{10}+X^{8}+X^{7}+X^{6}+X^{3}+X+1$

The frame CRC has a Hamming distance of six for payload lengths up to and including 248 bytes. For payload lengths greater than 248 bytes the CRC has a Hamming distance of four. Thus, when the length of Payload segment is no more than 248 bytes, the frame CRC can detect 5 bits errors. When the length of Payload segment is more than 248 bytes, the frame CRC can detect 3 bits errors [17].

In contrast to the CAN data frame, the FlexRay data frame has two main features. First, user data of the FlexRay data frame is much longer, and more flexible than that of CAN data frame. In a CAN data frame, the length of user data is at most 64 bits, which is the same as the length of other segments. It means that transmission efficiency is at most 50% for CAN. But in FlexRay data frame, the length of user data can be at most 2032 bits, when the length of other segments is fixed at 64 bits. The transmission efficiency can reach nearly 97%. Second, besides the Trail segment CRC checks errors which occur in the whole FlexRay data frame, the Header CRC field checks errors which occur in the main part of the Header segment. The CAN data frame has only one CRC.

2.2.3. FlexRay Frame Transmission Principles. FlexRay frame transmission in a Flexray network is based on the Time Division Multiple Access (TDMA) method: all communication is organized in communication cycles (FlexRay cycles), and each communication cycle is made up of a defined number of time slots exclusively reserved for frame transmission. Communication in a FlexRay network is characterized by a hybrid communication structure. The overview of FlexRay frame transmission is given in Figure 2.4. Shown in Figure 2.4, each communication cycle of FlexRay comprises a static (synchronous) and a dynamic (asynchronous) segment. The static communication segment is provided for deterministic frame transmission. Bandwidth in the dynamic communication segment, on the other hand, is available for need-based frame transmission. Each communication cycle exhibits two other time segments. The "symbol window" segment serves to check the operation of the bus guard. During the "Network Idle Time - NIT" segment, the FlexRay nodes compute the correction factors required for synchronization of their local clocks. If necessary, an offset correction is performed at the end of the NIT (rate correction is always performed distributed over the entire communication cycle). No data is transported during the NIT.

The static segment is provided for deterministic message transmission that is predestined for the transport of real-time relevant data. This segment is subdivided into a configurable number of static slots (maximum 1023). Assigned to each slot is a FlexRay data frame that is transmitted by a specific FlexRay node. The static slot, FlexRay data frame and FlexRay node are interrelated by the slot number and message identifier (ID) contained in the Header segment of the FlexRay data frame, and the value of the slot counter implemented on each FlexRay node. In each communication cycle, the slot counters are incremented synchronously after each static slot has been executed. The FlexRay data frame associated with the values of the slot counter, and identified by an ID, are transmitted by the relevant FlexRay node according to the slot counter. After FlexRay data frame transmission, the static slot itself is terminated by the CID (Channel Idle Delimiter).

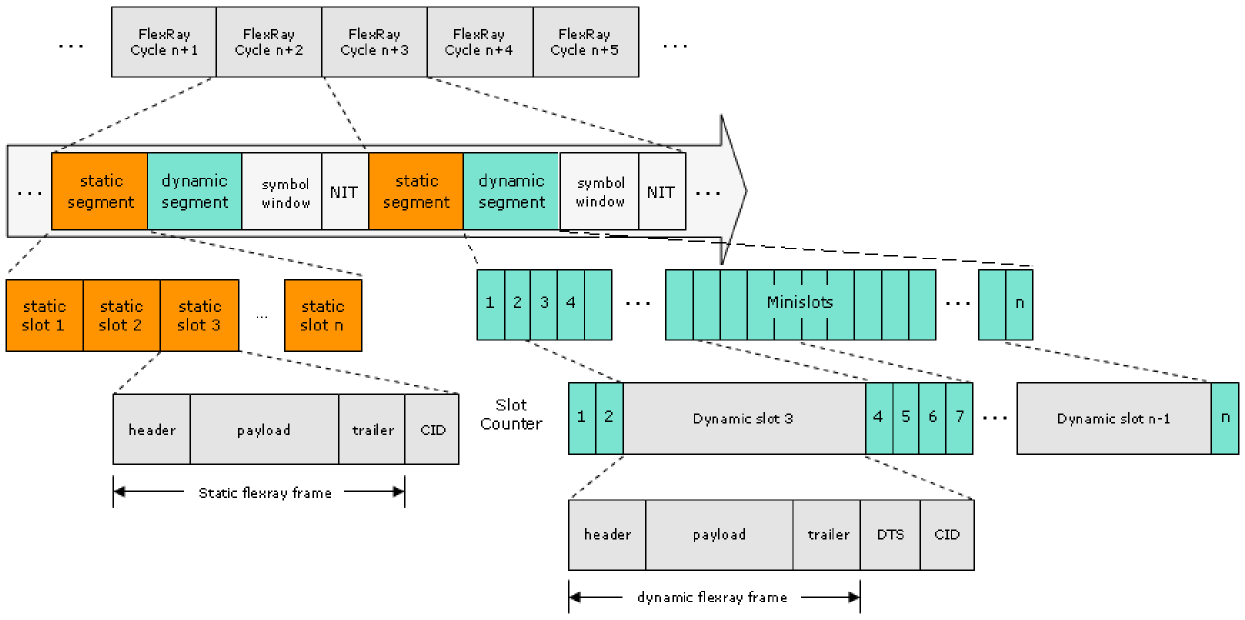


Figure 2.4 FlexRay frame transmission

The optional dynamic communication segment, which always has the same length, is available for transport of sporadically occurring data (e.g. diagnostic data), and it makes a contribution toward economical utilization of bandwidth. Bus access in the dynamic communication segment is based on the mini-slotting or Flexible Division Multiple Access (FTDMA) method. As in the dynamic segment, the slot counters found on the FlexRay nodes are incremented synchronously. However, the FlexRay data frames associated with values of the slot counter are only transmitted if there is a send request. In this case, the relevant FlexRay data frame, marked by an ID, is transmitted by the relevant FlexRay node. The slot counters are stopped for the duration of the frame transmission and are then synchronously incremented. If there is no send request, the slot counters are incremented after the defined time of a minislot.

FlexRay data frames are given implicit priorities by the FTDMA method: The lower the slot counter value, the higher the priority of the associated FlexRay data frame. As in the static segment, FlexRay frame transmission is terminated by the CID (Channel Idle Delimiter). Beforehand, dynamic adaptation to the mini-slotting interval is performed by DTS (Dynamic Trailing Sequence).

The situation of two channels application is similar. Two channels can transmit same data or totally different data simultaneously. Figure 2.5 shows the overview of two channels

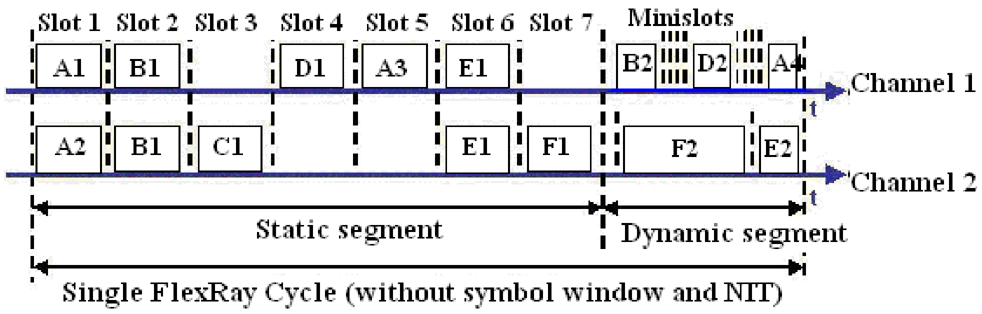


Figure 2.5 FlexRay frame transmission in two channels

frame transmission. Shown in Figure 2.5, A, B, C, D, E and F are nodes connected to the two channels FlexRay network. Squares represent FlexRay data frames. The name of each square represents the node transmitting the FlexRay data frame and the number of data frame to the node. For example, “A2” represents this data frame is the second data frame sent by the node A. Static slots are fixed length and arranged to relevant nodes for frame transmission. Dynamic frames have variable length and consist of several minislots.

FlexRay has a special encoding process before sending data frames into the channel. In FlexRay encoding process, a FlexRay data frame is separated into hundreds of individual bytes packets and added some extra bits. The flowchart of FlexRay encoding process is given in Figure 2.6. After FlexRay encoding process, FlexRay data frames are assembled into a bit stream, which is shown in Figure 2.7.

Compared to CAN frame transmission, FlexRay frame transmission has two main features. Firstly, CAN frame transmission in nodes is event-triggered (dynamic). Uncertain collisions are possible in this architecture. FlexRay frame transmission in nodes is hybrid of time-triggered (static) and event-triggered (dynamic). It can avoid uncertain collisions and also provides advantages of event-triggered. Secondly, there is a special encoding process before FlexRay data frames are transmitted to the channel. In this process, FlexRay data frames are separated into individual bytes, and some bits are added between each byte.

2.2.4. FlexRay Error Handling Mechanisms for EMI. The FlexRay contains two basic mechanisms to errors. For significant errors, the POC (Protocol Operation Control):halt state is immediately entered. The FlexRay also contains a three-state degradation model for

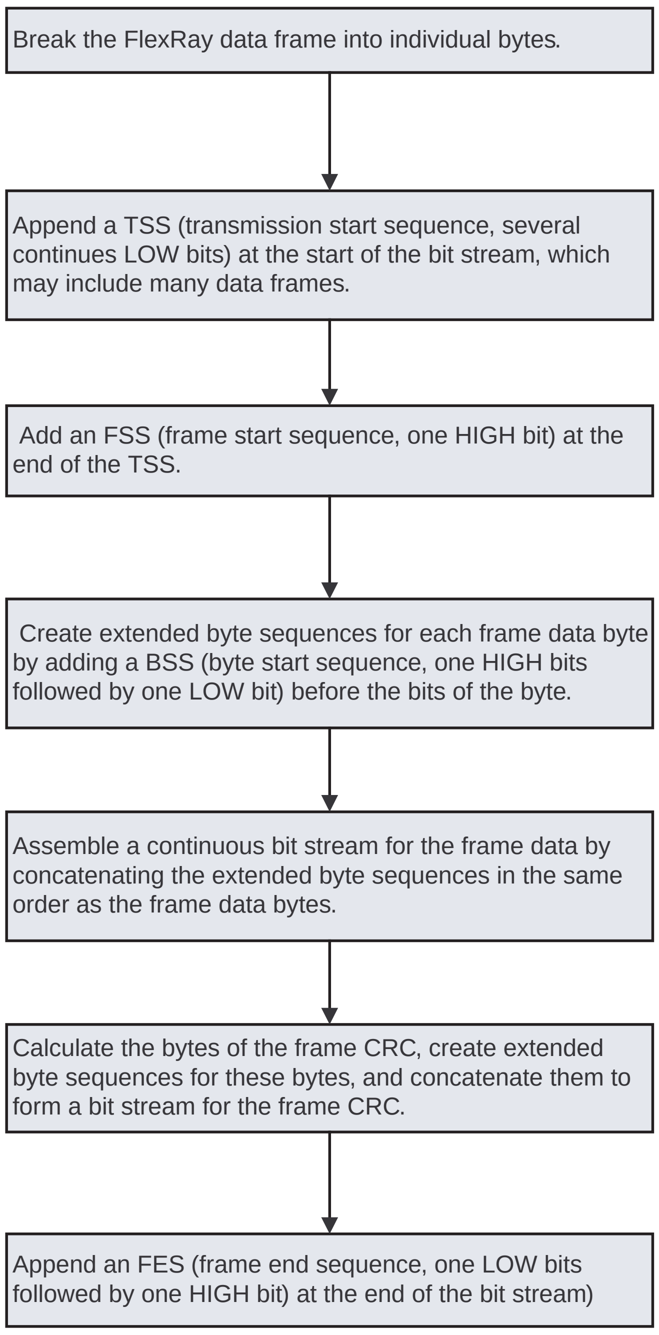

Break the FlexRay data frame into individual bytes.
Append a TSS (transmission start sequence, several continues LOW bits) at the start of the bit stream, which may include many data frames.
Add an FSS (frame start sequence, one HIGH bit) at the end of the TSS.
Create extended byte sequences for each frame data byte by adding a BSS (byte start sequence, one HIGH bits followed by one LOW bit) before the bits of the byte.
Assemble a continuous bit stream for the frame data by concatenating the extended byte sequences in the same order as the frame data bytes.
Calculate the bytes of the frame CRC, create extended byte sequences for these bytes, and concatenate them to form a bit stream for the frame CRC.
Append an FES (frame end sequence, one LOW bits followed by one HIGH bit) at the end of the bit stream)

Figure 2.6 The FlexRay encoding process

errors that can be endured for a limited period of time. In this case entry to the POC:halt state is deferred, at least temporarily, to support possible recovery from a potentially transient condition.

There are three general conditions that trigger entry to the POC:halt state immediately. They are product-specific error conditions (Built-In Self Test errors and sanity checks), error

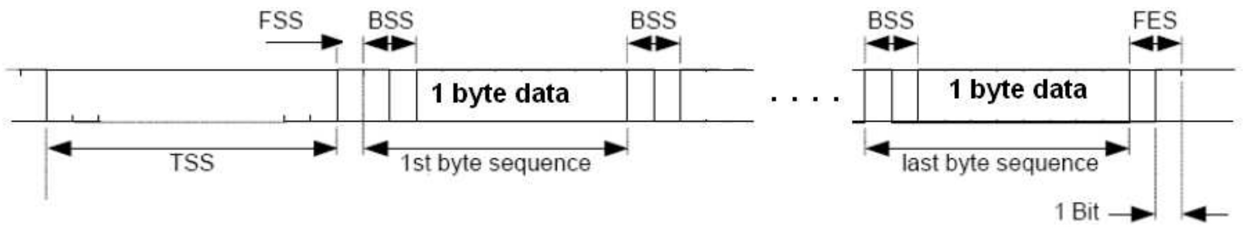


Figure 2.7 The bit stream after the FlexRay encoding process

conditions detected by the host that result in a FREEZE command being sent to the POC via the Controller Host Interface, and fatal error conditions detected by the POC or one of the core mechanisms (Including errors detected by the CRC).

In FlexRay, besides immediate entry to the POC:halt state, there is a three-state error handling mechanism referred to as the degradation model. It is designed to react to certain conditions detected by the clock synchronization mechanism that are indicative of a problem, but that may not require immediate action due to the inherent fault tolerance of the clock synchronization mechanism. This makes it possible to avoid immediate transitions to the POC:halt state while assessing the nature and extent of the errors [15].

Errors caused by EMI are bit or burst errors on FlexRay data frames, which can be detected by the Header CRC or the Frame CRC. They belong to fatal errors and will trigger FlexRay enter to the POC:halt state immediately. In the available documentation about the FlexRay architecture, no mention of any mechanisms similar to the CAN ARQ scheme to retransmit corrupted frames. The problem how to deal with the POC:halt caused by the CRC detecting errors, seems to be left to application programs.

2.3. THE ARQ SCHEME

2.3.1. The Basic Principles of the ARQ Scheme. The Automatic Repeat-reQuest (ARQ) is an error control method for data transmission which uses acknowledgments and timeouts to achieve reliable data transmission. An acknowledgment is a message sent by the receiver to the transmitter to indicate that it has correctly received a data frame. A timeout is a reasonable point in time after the sender sends the data frame; if the sender does not receive an acknowledgment before the timeout, it usually retransmits the frame until it receives an acknowledgment or exceeds a predefined number of retransmissions [18].

Types of ARQ scheme include Stop-and-wait ARQ, Go-Back-N ARQ and Selective Repeat ARQ. The Stop-and-wait ARQ scheme is the most basic ARQ scheme. Its communication is done one frame at one time. After sending each frame, the transmitter waits for the ACK (acknowledgement) signal and doesn't send any further frames until it is received. If the received frame is damaged or lost, the receiver discards it and does not send an ACK. If a certain time, known as the timeout, passes without ACK, the sender sends the frame again. In order to avoid duplicate frames caused by ACK lost or too short timeout. In common, a 1 bit sequence number is defined in the header of the frame. This sequence number alternates (from 0 to 1) in subsequent frames. When the receiver sends an ACK, it includes the sequence number of the next packet it expects. This way, the receiver can detect duplicated frames by checking if the frame sequence numbers alternate. If two subsequent frames have the same sequence number, they are duplicates, and the second frame is discarded. Similarly, if two subsequent ACKs reference the same sequence number, they are acknowledging the same frame. Go-Back-N ARQ and Selective Repeat ARQ schemes employ buffers to improve communication efficiency. They are widely used in computer network protocols.

A variation of ARQ is Hybrid Automatic Repeat reQuest (HARQ), which has better performance, particularly over wireless channels, at the cost of increased implementation complexity. The HARQ scheme employs not only the error-detection code, but also the error-correction code to control errors in transmission. Implementations and evaluations of HARQ modified CAN are finished by my research partner, Krishna chaitanya suryavenkata Emani [19].

2.3.2. The Stop-and-wait ARQ Scheme in CAN. The error-handling scheme employed in CAN is based on the Stop-and-wait ARQ scheme and given in Figure 2.8. Shown in Figure 2.8, the CAN transmitter sends a data frame (Frame 1) with specified Identifier (11 bits for CAN 2.0A, 29 bits for CAN 2.0B). If one or more receivers correctly receive the data frame, they send the ACK to the transmitter by setting the ACK slot in the data frame (Frame 1) to dominant “0”. The transmitter can detect dominant “0” in the ACK slot and send the next frame (Frame 2). If no receiver receives the data frame, no ACK is sent by receivers. The transmitter cannot detect dominant "0" in the ACK slot and retransmit the frame (Frame 1). If errors are detected by any receiver, the receiver send the error frame, which can be received by other nodes (including transmitter) in bus. After receiving the error frame, the transmitter retransmit the corrupted data frame. The retransmission process repeats until the transmitter receives the ACK.

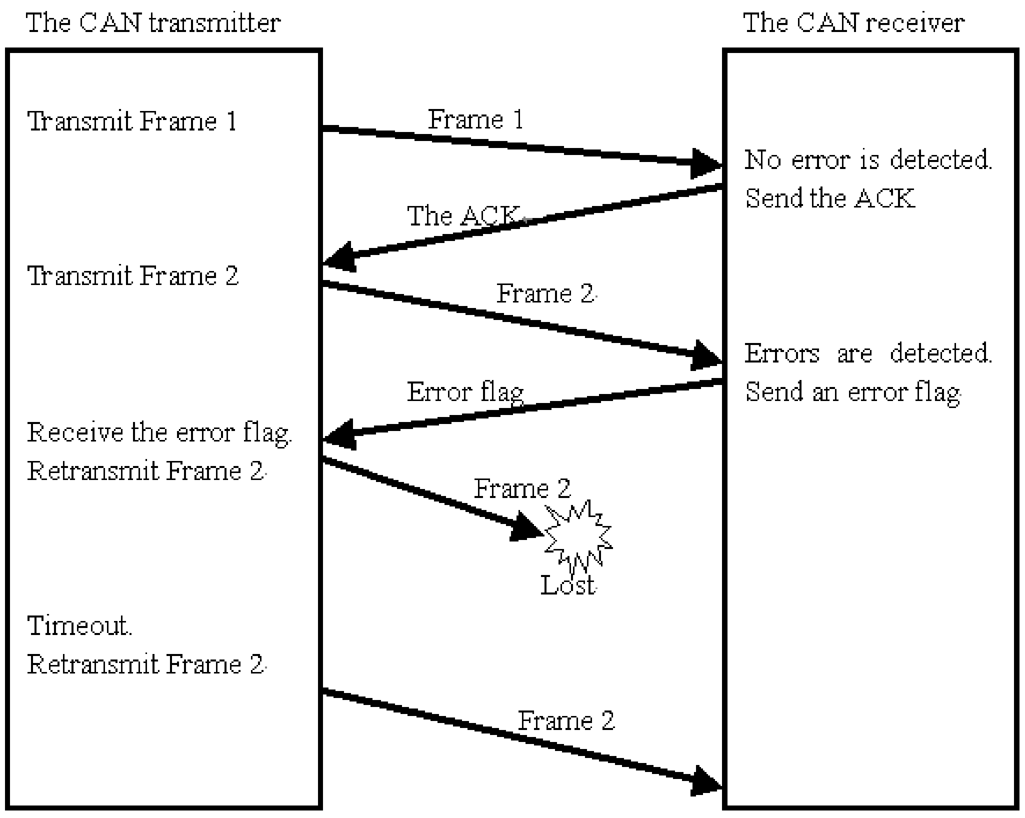


Figure 2.8 The Stop-and-wait ARQ in CAN

Basically, the Stop-and-wait ARQ scheme of CAN employs error-detection and retransmission to guarantee reliability of transmission.

2.4. THE BCH CODE

2.4.1. Basic Knowledge of BCH Codes. In general, error-correction coding is the technique used to correct the errors in the received data. It introduces systematic redundancy in the transmitting data in order to combat the error [20]. The BCH codes are among the most important block codes that are much studied in coding theory. Because they can achieve significant coding gain, and the complexity of their decoders is such that they are implementable even at high speeds. The BCH codes are linear cyclic codes that are always defined by their code generator polynomial.

The block length n (encoding output length) for BCH codes is always $n = 2^m - 1$ for $m \geq 3$, and the number of errors can be corrected is bounded by $t < (2^m - 1)/2$. Specific values for t and k (encoding input length) can be found using algebraic techniques for determining code polynomials. Table 2.1 and Table 2.2 give values for n, k and t for BCH codes employed in this thesis.

Table 2.1 Relationship between n, k and t in the BCH codes (n=63)

n	k	t
63	36	5
63	30	6
63	24	7
63	18	10
63	16	11
63	10	13
63	7	15

Table 2.2 Relationship between n, k and t in the BCH codes (n=511)

n	k	t
511	376	15
511	304	25
511	184	45
511	112	59

2.4.2. Encoding and Decoding of BCH Codes. The BCH codes are cyclic codes so that encoding is simply accomplished using the general technique for cyclic codes. Given the code generator g(D), the code words in systematic form are found using the following steps:

1. multiply the message polynomial $w(D)$ by D^{n-k}
2. calculate the remainder $\rho(D) = R_{g(D)}[D^{n-k}w(D)]$
3. generate the code polynomial $x(D) = \rho(D) + D^{n-k}w(D)$

Decoding of BCH codes is similar to that of any cyclic code. Because the foundation for all BCH decoding algorithms is the algebraic structure of the codes, the decoders are straightforward once the algebraic fundamentals are established. Decoding of BCH codes consists of three steps:

1. Calculate a syndrome from the received code polynomial. There are 2t components of the syndrome $S_1, S_2, , S_{2t}$. The syndrome of an undistorted code polynomial is zero so that it is a function only of the transmission errors and the code structure. Calculation of each component of the syndrome is done by finding the remainder of division of the received sequence by a polynomial $\phi_i(D)$, which was specified a t the time of the definition of the code. The polynomial long division is accomplished to calculate syndrome components. There are at most different division circuits required for a t error correcting code.

2. Determine the positions of the errors in the received polynomial using a two-step procedure:

   (a) Determine an error location polynomial from the syndrome components found in step 1. An interactive algorithm is available for finding this polynomial. This algorithm is called the Berlekamp algorithm, after its inventor. The complexity of this algorithm is proportional to $2t^2$.

   (b) Find the roots of the polynomial found in step 2a. There roots directly determine the location so f the errors in the received polynomial.

3. Correct the errors in the received polynomial to find the transmitted codeword and thus the transmitted information.

BCH decoders are commercially available and are widely used today [21].

2.4.3. Implementation of BCH Codes in Matlab Simulations. In this thesis, the BCH code is employed by proposed Modified ARQ scheme to correct errors. In simulations, the encoding and decoding process of the BCH codes are finished by Matlab commands “bchenc” and “bchdec”. In utilization, the BCH code parameters , n, k and t, are important for choosing the proper BCH code to counter errors. When n is fixed, decreasing the length of input, k, will increase the error-correction capability of the BCH code, but decrease transmission efficiency due to the shorter k. The converse situation works too. Thus, in applications, the t should be made exactly larger one bit than maximum error bits so that the BCH code can meet error-correction requirements and maximize transmission efficiency simultaneously. The BCH code has flexibility to meet requirements of various error extents.

3. EMI MEASUREMENTS ON THE CAN HARDWARE TESTBED

In order to mitigate the effects of EMI on automobile communication protocols by employing shielded cables and error-correction mechanisms, the performance of protocols in the EMI environment must be measured based on hardware experiments. CAN, as the communication protocol most widely used in the automobile and construction machinery, was chosen as the experimental automobile communication protocol. The CAN hardware testbed was built in the EMI environment to measure signal waveforms and the actual bit rate. Figure. 3.1 is a photo of the CAN testbed. Based on the analysis of measurement results, shielded cables proved to be effective to mitigate the effects of EMI.

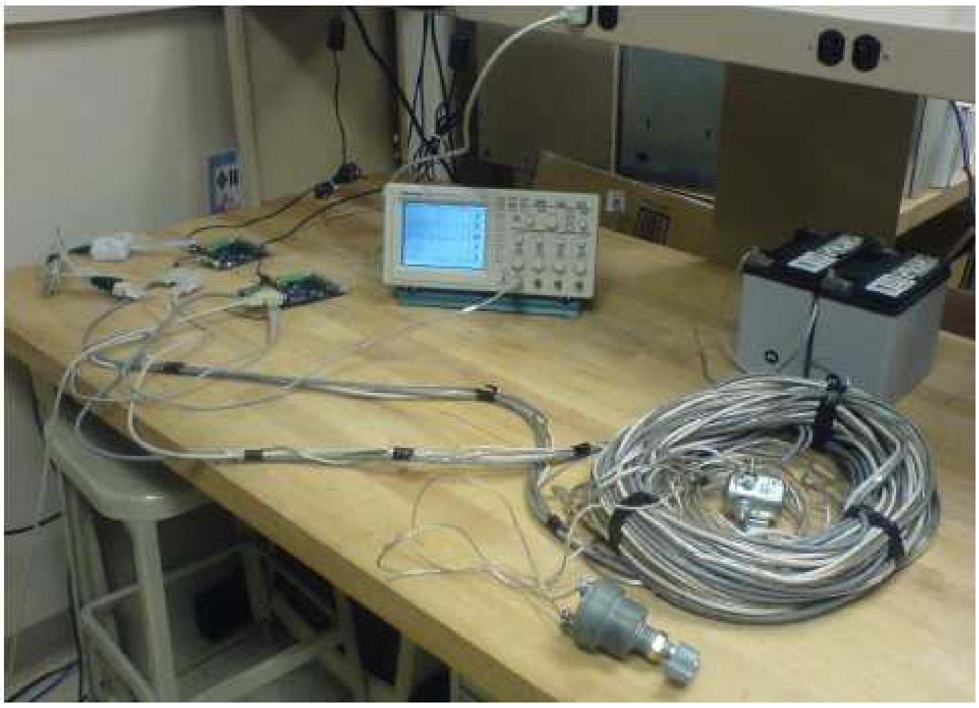


Figure 3.1 The CAN testbed photo

3.1. HARDWARE SETUPS FOR CAN TESTBED

A CAN bus with up to four nodes were implemented by Intel 8051 microprocessors and connection cables. A magnetic relay and a manual switch were used to generate the EMI which was then coupled with the communication bus. Data waveforms were recorded using an oscilloscope. When using the oscilloscope in the measurement, caution was taken to preserve the balance of the differential bus structure of CAN.

The hardware setup is shown in Figure 3.2. Every CAN node was implemented by a C8051 board (C8051f040 Mixed Signal ISP Flash MCU board from Silicon Laboratories). By flushing different control programs into C8051 from the PC, the board can become either a transmit node, receive node, or silence node. Nodes were connected through DB9 serial cables.

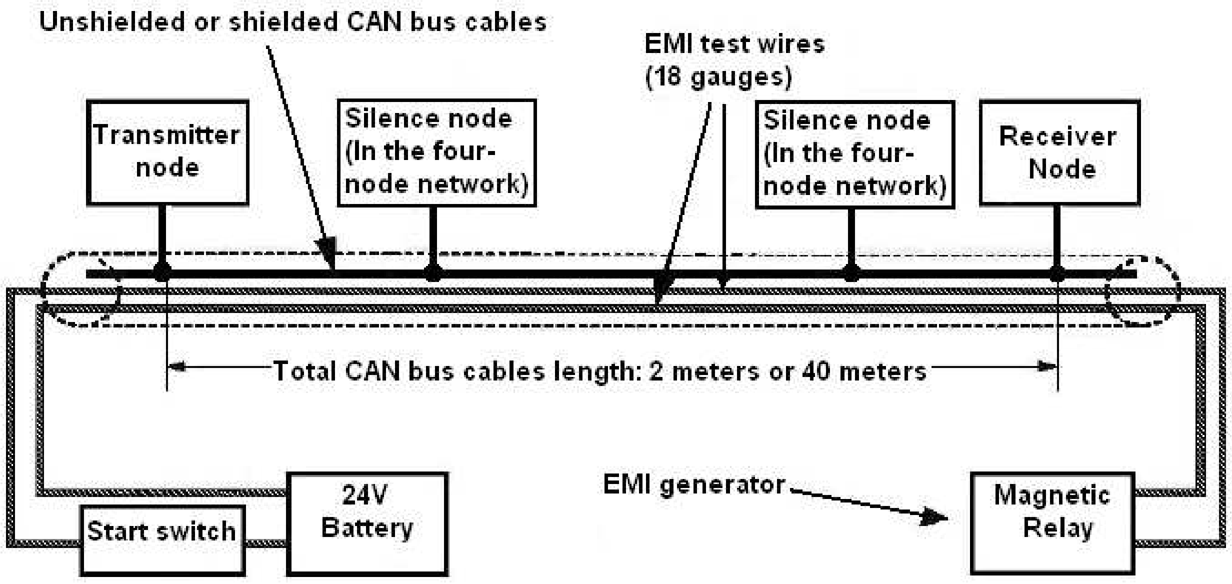

Unshielded or shielded CAN bus cables
EMI test wires (18 gauges)
Transmitter node
Silence node (In the four-node network)
Silence node (In the four-node network)
Receiver Node
Total CAN bus cables length: 2 meters or 40 meters
EMI generator
24V Battery
Start switch
Magnetic Relay

Figure 3.2 The CAN testbed hardware blocks

In order to compare the effects of EMI on different cables, two kinds of cables were used for the bus: shielded and unshielded. The unshielded cables were the traditional CAN bus communication cables, which were four unshielded un-twisted wires. The shielded cables were un-twisted wires with shielded layers connected to the ground of the C8051 boards. In four-node network measurements, Y cables were necessary to connect every node to the CAN bus.

The EMI generator circuit consists of a magnetic relay unit, a 24 V battery, and a switch, serially connected to each other through 18-gauge wires that were coiled to CAN cables. The magnetic relay unit is a large inductive load. The switch controls the EMI generation circuit. Turning on and off the switch make the relay to generate EMI. The EMI pulses couple to the CAN bus and cause reduction in the CAN bus speed and errors in CAN communication.

3.2. SOFTWARE SETUPS FOR CAN TESTBED

This study used software, Silicon Laboratories IDE, to control actions of C8051f040 Mixed Signal ISP Flash MCU boards and make them communicate with each other in a CAN bus environment. In the two-node network, one node was set to the sender node, which sends data, while another was set to the receiver node, which receives data sent by the sender node. In the four-node network, besides the sender and receiver nodes, two more nodes were set in the silence mode to receive CAN bus data. As listeners, they can receive all data from the sender node. The silence Mode can be used to analyze the traffic on the CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, error frames) [22].

The retransmission function of CAN was disable in all hardware experiments. Through setting the value of Bit Timing Register (BTR), the period of one bit, Tq, was controlled. The CAN bus bit rate was computed as 1/Tq, and it can be set at some bit rates up to 1Mbps. When CAN bus works at a certain set bit rate, the actual bit rate, at the receive node, are not exactly the set one. Figure 3.3 shows the curve of the actual bit rates relative to the set bit rates.

In Figure 3.3, the actual bit rates are close to, but lower than the set bit rates. That is because the receive node takes some time to process received data after every data frame arriving. Processes include storing received data, reading certain registers, and recording the number of errors. When such situation occurs, the CAN bus has to wait for processes finishing. The difference between the set bit rate and the actual bit rate may change, and is dependent on different custom control programs.

In physical layer, data transmission of CAN is based on transmission of CAN data frames. Following specified structure, user data is packed into CAN data frames for transmitted in physical layer. According to the CAN frame structure, one data frame has 64 bits of Data Field, which include effective user data, and 44 bits of other data. When the receive node receives one data frame, it actually receives 108 bits data, approximately three fifths of which

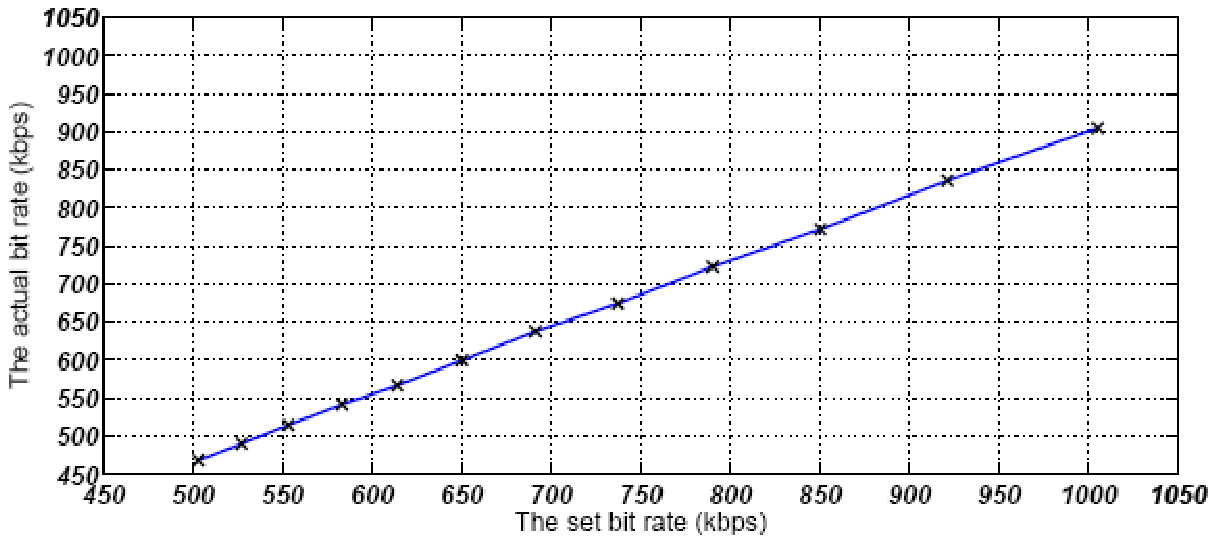


Figure 3.3 The relationship between set bit rates and actual bit rates

are user data. The bit rate of the CAN bus is defined on the basis of all received data, including both user data and other data.

In order to observe CAN data frames waveforms more conveniently and prevent the bit-stuffing, user data, which need to be packed into each CAN data frame, were set to be eight bytes of 55. In the Data Field of the CAN data frame, 55 was stored as 01010101. Thus, the Data Field of each CAN data frame was thirty-two 01 alternating bit streams. Timer 4 of the C8051 board was programmed to time the communication between the transmitter and receiver. The CAN bus communication was stopped automatically in 20 seconds controlled by the timer. The receive node records the total number of received bits and the number of errors. The total number of received bits dividing by 20 seconds is the number of bits of successful received data per second by the receive node, which is seen as the actual bit rate.

When receives frames, if any errors occurs in this frame, the receiver node can record the number of errors, and then request retransmission. The bit error rate can be gained by dividing the number of errors by the total number of received bits.The actual bit rate is an average value in a 20-second experimental time interval. In order to analyze the effect of EMI results from every switch change, the switch were turned on and off at 40 times/20 sec and 60 times/20 sec. In order to analyze the EMI effects on the bit error rate, the bit rates were set at different values when the switch were kept turning on and off at 40 times/20 sec. Every value in results section is an average based on repeating the measurements 50 times.

3.3. MEASUREMENTS AND DATA RECORDING

Precaution is taken to preserve the circuit balance of the differential bus structure of CAN transceivers. The waveforms of CAN data frames were measured by a digital oscilloscope (Tektronix TDS5034B). It is necessary to connect the two probes of the scope to CAN H (Pin 7) and CAN L (Pin 2) pins of the DB9 connector of the receive node, as shown in Figure 3.4.

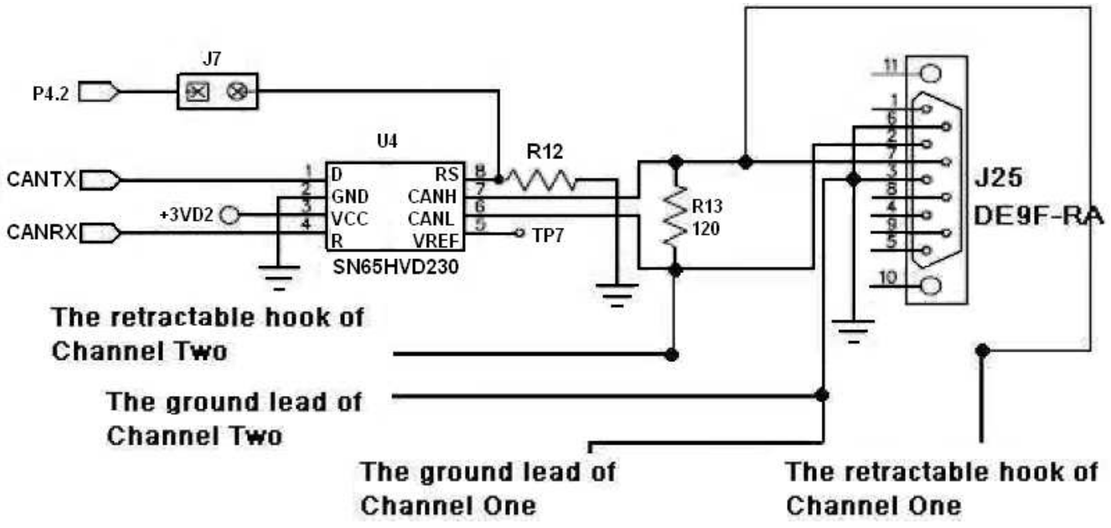

J7
P4.2
U4
R12
CANTX
CANRX
+3VD2
D
GND
VCC
R
RS
CANH
CANL
VREF
TP7
SN65HVD230
R13
120
J25
DE9F-RA
The retractable hook of Channel Two
The ground lead of Channel Two
The ground lead of Channel One
The retractable hook of Channel One

Figure 3.4 Probe connections setup

The cable length of the two probes should be the same so that the impedances and the capacitances of the two probes form a balanced circuit. The equivalent schematic diagram of the measurement circuit is shown in Figure 3.5, where R1 and R2 represent the resistor of the two probes, respectively.

And C1 and C2 were the capacitances of the grounding wires of the two channels. Ground 1 is the ground of the receive node, while ground 2 and ground 3 were the grounds of Channel One and Channel Two, respectively. Using such connection, the nodes ground and two channels grounds were connected through balanced capacitors. A third probe may be used to measure the coupled EMI only. It may also be used as the trigger signal if only the

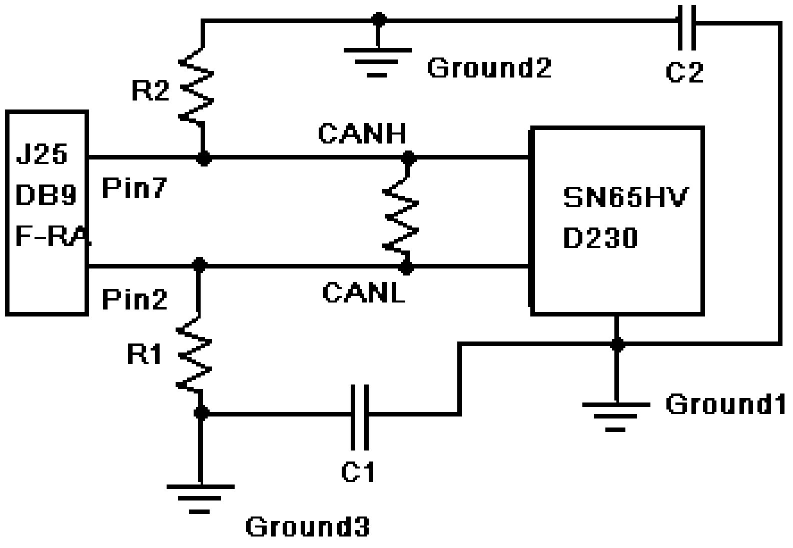


Figure 3.5 Probe connections schematic

corrupted data waveforms were to be recorded. The waveforms of data frames and EMI were recorded by choosing the math mode of the oscilloscope, Ch1 . Ch2.

Note that a common mistake is to use the retractable hook and ground lead of one probe connected to CAN H and CAN L. Since the electrical grounds of the oscilloscope and the microprocessor boards were essentially connected through power outlets, imbalance was created between the signal pins of the CAN differential bus thus causing inaccurate measurements. Other tips gained in the hardware experiment include: keeping EMI generation circuits away from the probes as far as possible so that the EMI was not coupled directly into the scope; ensuring reliable connections experiment setups; and making solid contacts of the probes with the measured pins.

3.4. SIGNAL WAVEFORMS AND ANALYSIS

Positions of the probes of the two channels are connected as shown in Figure 3.4. When no communication occurs on the CAN bus, then the voltage between CANH and CANL is 0 V. After starting CAN bus communication, one can observe CAN data frames. Figure 3.6 shows the waveforms of data frames at the bit rate of 503 kbps.

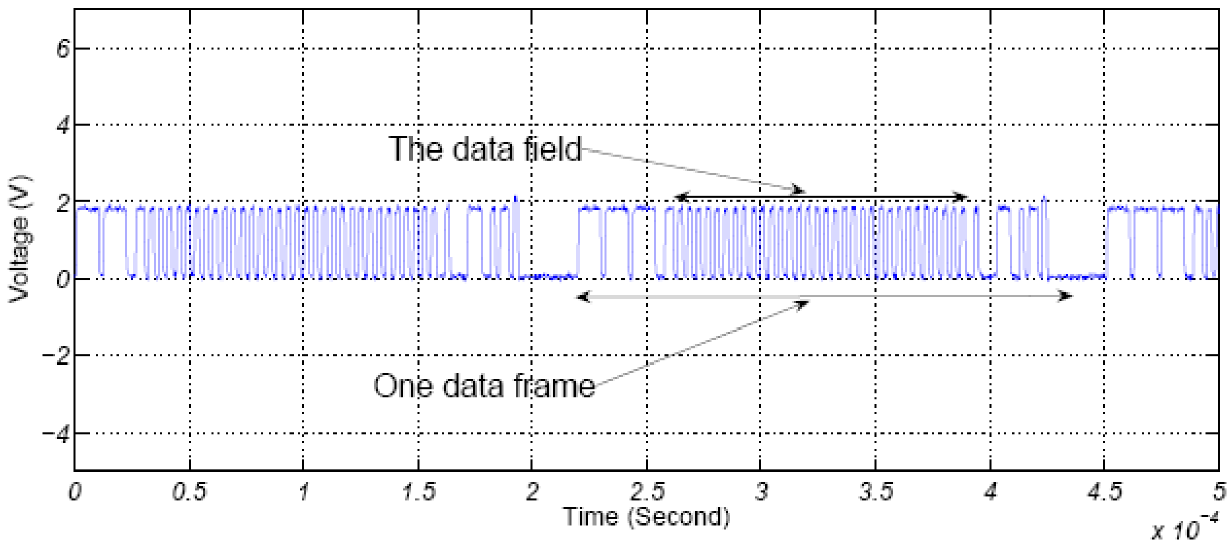


Figure 3.6 Waveformes of CAN data frames at the bit rate of 503 kbps

In Figure 3.6, the amplitude of data frames is 1.8 V. One data frame lasts from 0.22 ms to 0.44 ms, while the data field lasts from 0.26 ms to 0.39 ms. The length of the Data Field and the data frame are 0.13 ms and 0.22 ms, respectively. At the bit rate of 503 kbps, the length of every bit is 1/503000= 0.002 ms. The observed value is roughly equal to the theoretical value, 0.002*64 = 0.128 ms and 0.002*108 = 0.216 ms. The length of repeat portions is 0.232ms. Thus, this is 0.232- 0.216 = 0.016 ms time interval between two data frames.

Keeping the probes positions constant, the CAN bit rate is set to 1 Mbps. Figure 3.7 shows the waveforms of one data frame at this set bit rate.

In Figure 3.7, the amplitude of data frames is the same to that in Figure 3.6. One data frame lasts from 0.08 ms to 0.019 ms, while the Data Field lasts from 0.1 ms to 0.164 ms. When the CAN bus speed is 1 Mbps, every bit lasts approximately 0.001 ms. The Data Field is 64 bits, and the theoretical length, which is 0.064 ms, is equal to what Figure 3.7 shows: 0.164-0.1=0.064 ms. The data frames theoretical length of 0.108 ms is close to what is actually seen: 0.019.0.08 = 0.11 ms. The length of repeat portions is 0.120 ms. Thus, this is 0.120-0.108 = 0.012 ms time interval between two data frames.

Because EMI lasts only a short interval of time, a trigger source is necessary to see the waveforms of EMI. Channel One was set as the trigger source, whose trigger level is 4 V, and

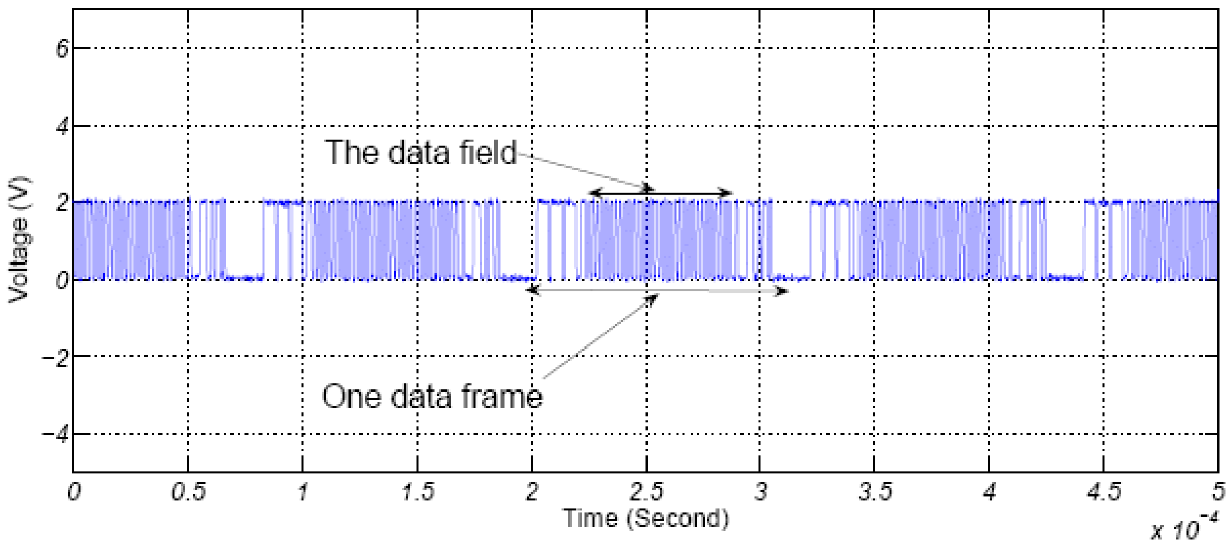


Figure 3.7 Waveformes of CAN data frames at the bit rate of 1 Mbps

the switch was turned on and off to generate EMI. The digital oscilloscope shows and records the waveforms of EMI. Figure 3.8shows the EMI waveform made by one switch change.

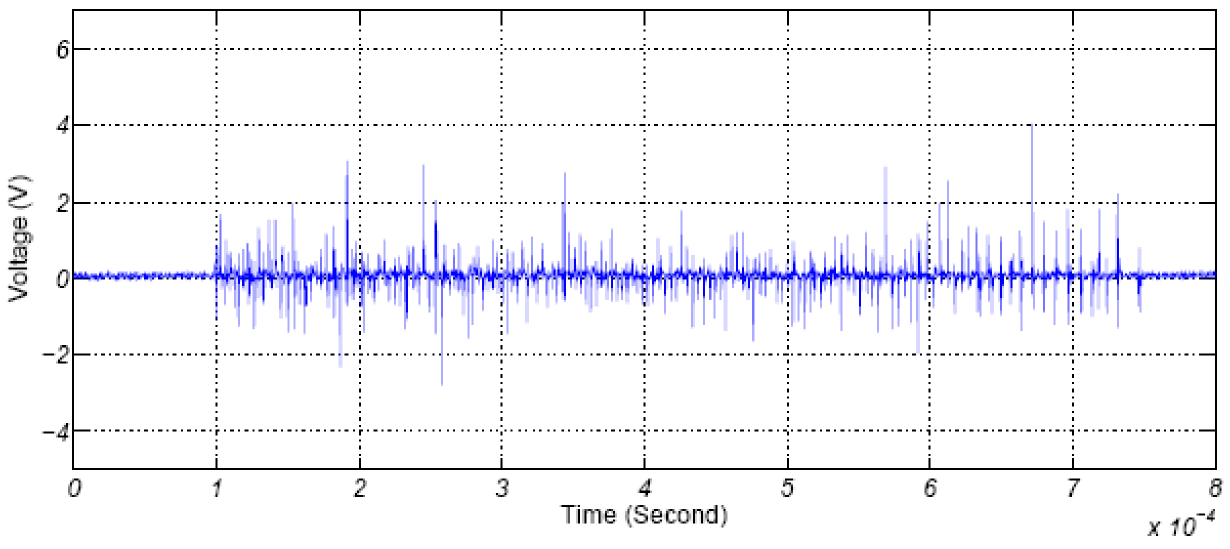


Figure 3.8 Waveformes of Electromagnetic Interference

In Figure 3.8, the voltage peak of EMI is 4 V. The length of EMI is approximately 0.65 ms. The length of EMI generated by one switch change vary in a range. Based on 50 times tests, the average length of EMI is 0.49 ms. Moreover, in tests, turning off the switch in the EMI generation system triggers the oscilloscope to record waveforms of EMI. It makes sense to the principle of EMI occurring.

In order to record waveforms of the data frames mixed EMI. The trigger level of Channel One is at 4 V, and the switch is turned on and off to generate EMI. First, 2-meter unshielded cables are used as the CAN communication cable. Figure 3.9 shows the waveforms of EMI based on an one-time switch change affecting data frames at the bit rate of 1 Mbps.

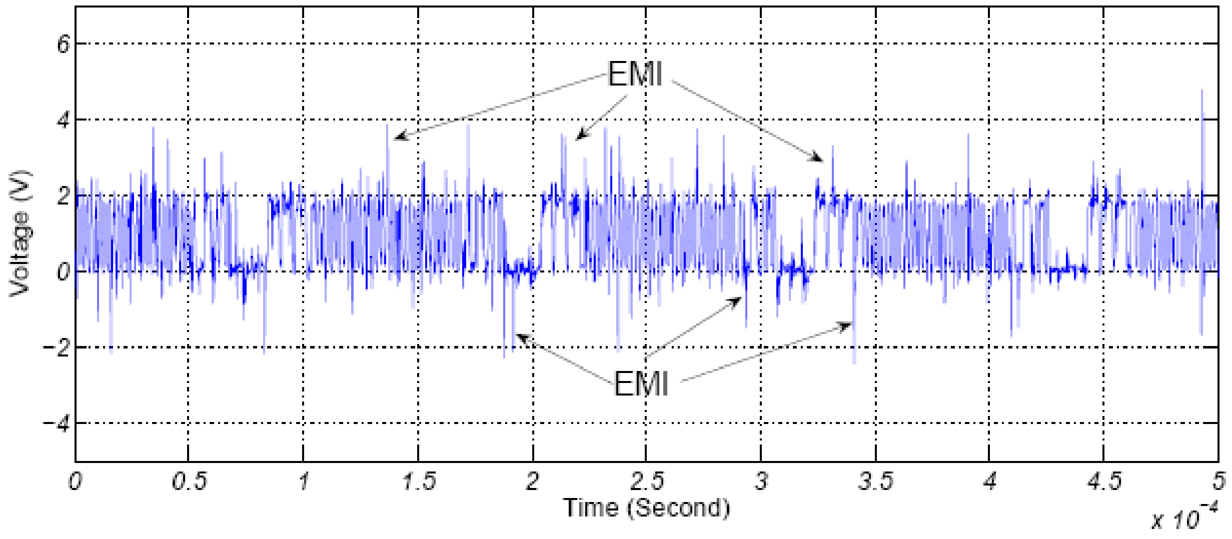


Figure 3.9 Waveformes of CAN data frames with EMI on unshielded cables

In Figure 3.9, data frames, to a large extent, are distorted by EMI on unshielded cables. Based on 50 measurements, on average, there are approximately three EMI pulses per 0.1 ms, whose peak values are in the range from 3 V to 4 V. Some bits in the Data Field even change to wrong voltage levels, which affects the determination and may results in errors. Analyzed by matlab, every data frame affected by EMI, on average, has 6 error bits. These errors can be detected by CAN, and wrong data frames can be retransmitted. However, retransmission causes a reduction of the bit rate.

When shielded cables are used in the system, the waveforms of EMI affecting the data frames are shown in Figure 3.10.

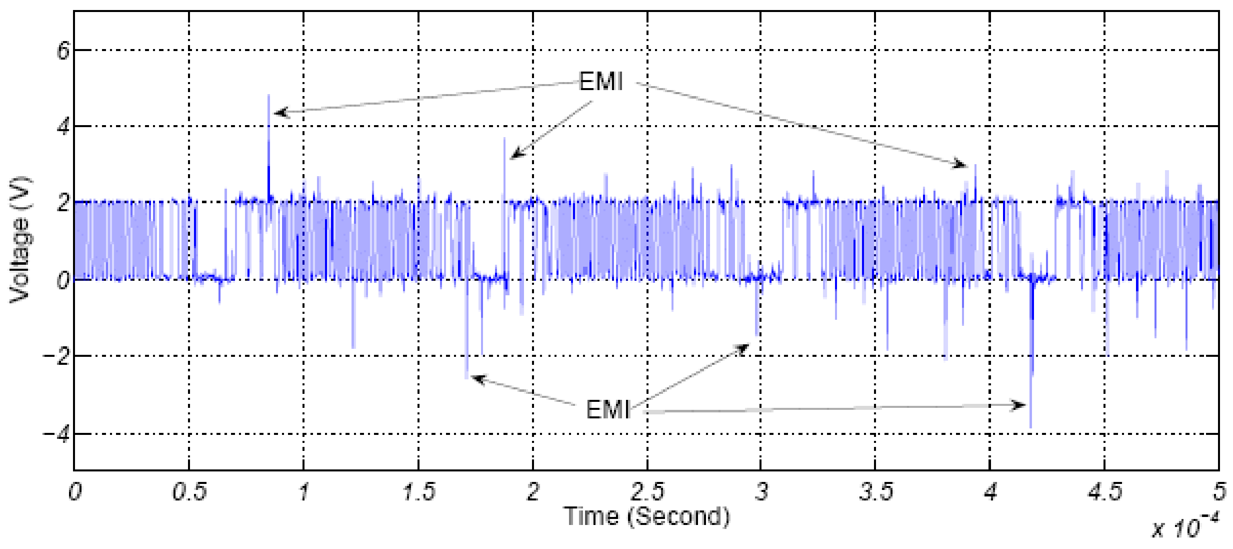


Figure 3.10 Waveformes of CAN data frames with EMI on shielded cables

Compared with Figure 3.9, the waveforms of data frames on shielded cables are more clear and mixed by a lower EMI. Based on 50 measurements, a rare EMI pulse is beyond 3 V on shielded cables. Analyzed by matlab, every data frameaffected by EMI, on average, has less than 1 error bits. They are much less than those in unshielded cables situation. As shown by the performance measurements, in different length cables and set bit rates, such an extent of EMI affects the actual bit rate of CAN bus very little.

In tests, one fact is found that, with a small probability, EMI results in a halt of CAN bus communication. Based on 50 tests that show halts, the average time interval of a halt is 2.3 ms. In the set bit rate of 1 Mbps CAN bus, such a halt reduces 2300 bits, or approximately 21 data frames.

Moreover, tests show that the length of the communication cable affects the shape of the data frame waveforms. Data frames in the short cable have better edges than those in the long cable. The reason is the attenuation of the cables. Long cables have more attenuation than short ones do.

3.5. STATISTICS RESULTS AND ANALYSIS

Following the Figure 3.2, the CAN networks were built. After measurements, the results of unshielded cables are shown in Table 3.1.

Table 3.1 Two-node network using unshielded cables

		The actual bit rate (503 kbps)	The actual bit rate (1 Mbps)
2 meters	No EMI	456539 bps (100 %)	904861 bps (100 %)
	40 times/20 sec	456453 bps (99.98 %)	904222 bps (99.93 %)
	60 times/20 sec	456392 bps (99.97 %)	904032 bps (99.91 %)
40 meters	No EMI	456181 bps (100 %)	902633 bps (100 %)
	40 times/20 sec	456079 bps (99.98 %)	902010 bps (99.93 %)
	60 times/20 sec	456040 bps (99.97 %)	901734 bps (99.90 %)

It shows that in the 2-meter and 40-meter unshielded cables situation, EMI results in the reduction of the actual bit rate. The reduction is not large because it is affected by only one Relay unit. There are hundreds of similar units in automobiles and off-way machines, whose total effects are huge. The percentages in table are obtained by dividing the actual bit rates in EMI environment by those of No EMI situation.

Longer cables results in more reduction of the actual bit rates. In the set bit rate of 503 kbps and no EMI situation, the actual bit rate of using 40-meter cables is 0.37 kbps less than the one of using 2-meter cables. In the set bit rate of 1 Mbps, the reduction becomes 2.23 kbps.

More EMI cause more reduction of the actual bit rates. In the set bit rate of 503 kbps and 2-meter cables situation, when switch change is 2 times/sec, the average reduction of the actual bit rate is 0.096 kbps, which is a 0.02 percent reduction from the actual bit rate of the No EMI situation. Accordingly, when the switch change increases to 3 times/sec, the average reduction of the actual bit rate reduces by 0.146 kbps, which is a 0.03 percent reduction from the actual bit rate of the No EMI situation.

In higher set bit rates, effects of EMI are more obvious. In 2-meter cables and switch change of 2 times/s situation, when the set bit rate is 503 kbps, the actual bit rate can reach 99.98 percent of the actual bit rate in no EMI situation. However, when the set bit rate increases to 1 Mbps, the actual bit rate can reach 99.93 percent of the actual bit rate in no EMI situation. Thus, in unshielded cables situation, longer communication cables, faster set bit rates of the CAN bus and greater numbers of EMI can cause more serious reduction of the actual bit rates.

All unshielded cables were replaced with shielded ones, then the same measurements were taken on both the 2-meter and 40-meter shielded cables. The experiment results are shown in Table 3.2.

Table 3.2 Two-node network using shielded cables

		The actual bit rate (503 kbps)	The actual bit rate (1 Mbps)
2 meters	No EMI	456539 bps (100 %)	904861 bps (100 %)
	40 times/20 sec	456534 bps (100 %)	904859 bps (100 %)
	60 times/20 sec	456530 bps (100 %)	904857 bps (100 %)
40 meters	No EMI	456181 bps (100 %)	902633 bps (100 %)
	40 times/20 sec	456180 bps (100 %)	902630 bps (100 %)
	60 times/20 sec	456180 bps (100 %)	902628 bps (100 %)

When shielded cables were used, the performance of the CAN bus in an EMI environment is obviously improved. In shielded cables situation, there is almost no reduction of the actual bit rate at the set bit rates of 503 kbps and 1 Mbp, even in a 3 times/sec switch change. Moreover, when the length of shielded cables reaches 40 meters, they also effectively protect CAN communication from EMI. The reduction of the actual bit rate, which is caused by EMI, is still zero.

In shielded cables situation, the four-node network were built. In the four-node network, shielded cables also work well in mitigating EMI. Measurement results are shown in Table 3.3.

Table 3.3 Four-node network using shielded cables

	The actual bit rate (503 kbps)	The actual bit rate (1 Mbps)
No EMI	456154 bps (100 %)	902576 bps (100 %)
40 times/20 sec	456150 bps (100 %)	902575 bps (100 %)
60 times/20 sec	456149 bps (100 %)	902573 bps (100 %)

Except for a small reduction in the actual bit rate, the results of the four-node network are similar to those of the two-node network.

From the statistics of shown in the tables, the most significant truth found is that shielded cables can effectively protect CAN bus communication from EMI. At the set bit rate of 503 kbps and 1 Mbps, the shielded cables reduce EMI effects on the actual bit rates to zero. Moreover, an increase of the cables length and the number of nodes was found to reduce the actual bit rates lightly. However, they are not the main factors that affect the actual bit rate. The larger number of EMI, which is reflected by the times of changing switches, results in a greater reduction of the actual bit rate. This rule was found when the CAN cable is unshielded.

In two-node network including 40-meter unshielded or shielded cables, when the retransmission function is disable, the bit error rate of CAN bus in EMI environment was measured at different set bit rates. In the situation of the switch keeping turning on and off at 40 times/20 sec, the value of the bit time register were changed to set CAN bus at distinct set bit rates. In every bit rate, both unshielded and shielded cables situation were measured. By reading values of some registers, the number of errors in 20 seconds communication can be known. Then the bit error rate can be computed. The results are shown in Figure 3.11.

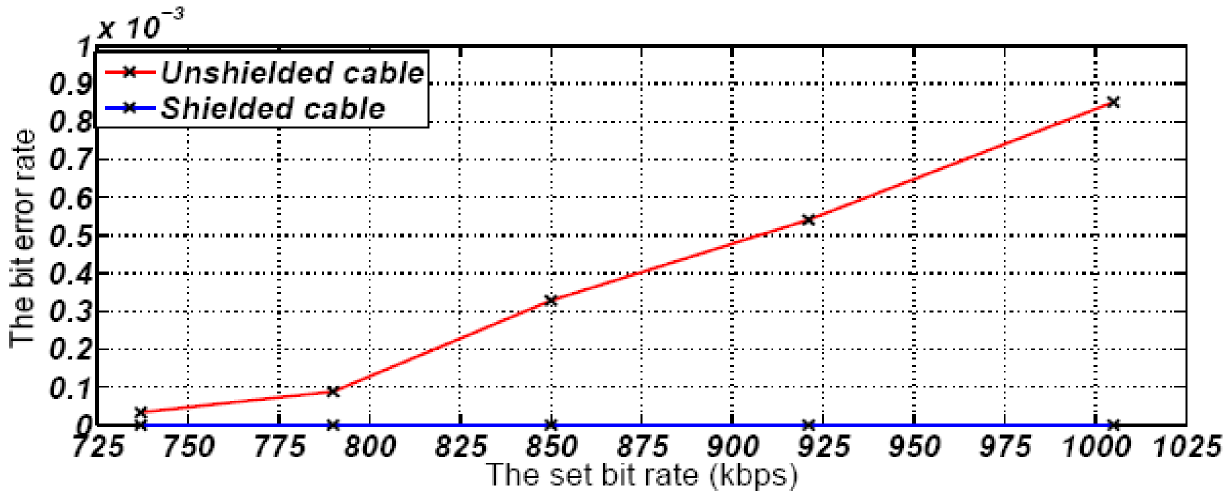


Figure 3.11 The bit error rate of CAN at different bit rates in EMI environments

The red curve represents the situation of the unshielded cable, while the blue one represents the situation of the shielded one. In the unshielded cables situation, the bit error rate increases when CAN bus is at the higher bit rate. In the set bit rate of 1.005 Mbps, the bit error rate reaches approximately 0.00085. In the shielded cable, the bit error rate is perfect zero in every set bit rate.

4. EMI MITIGATION METHODS FOR CAN

Hardware experiments prove that shielded communication cables effectively reduce burst errors caused by EMI. However, in general, a modern car has several kilometers cables, while the large machinery vehicle even has longer cables. Shielded cables in such length have higher cost than traditional unshielded cables. Without replacing unshielded cables, the error-correction code can be utilized to improve performances of CAN combating EMI. In this section, a Modified ARQ scheme employing error-correction codes is proposed to replace the original ARQ scheme in CAN. Matlab simulations implement a serious EMI environment and Modified ARQ CAN communication. Performances of Modified ARQ CAN including the bit error rate, the frame error rate and the number of retransmissions, are computed and analyzed.

4.1. PROPOSED MODIFIED ARQ SCHEME

Referring to results of CAN testbed, CAN with the Stop-and-wait ARQ scheme gets performance degradation in the EMI environment. A Modified ARQ scheme is proposed to combat EMI in CAN. The Modified ARQ scheme employs not only the error-detection codes, but also the error-correction codes to mitigate EMI. In Modified ARQ scheme, the BCH code (the error-correction code) is employed to encode user data and generate encoded user data. Encoded user data are loaded by the Data Field in a CAN data frame. The original 15 bit CRC in CAN data frame is still used to check errors. However, it is computed on the Start of Frame, the Arbitration Field and the Control Field, not on the Data Field anymore. When the data frame is received, the same error-detection and retransmission processes of original CAN are done. If the data frame passes the CRC check, encoded user data in the Data Frame is decoded by the BCH code to get user data.

CAN data frame structure is redrawn in Figure 4.1. The CAN with a Modified ARQ scheme is called the Modified ARQ CAN. Modified ARQ CAN data frame structure is given in Figure 4.2. The Modified ARQ CAN data frame makes modifications in the CRC Field and the Data Field. The CRC in CRC Field does not check the Data Field any more. Errors occurring in the Data Field are taken care by the error-correction code. The Data Field is separated into three parts: user data, BCH parity bits, and 1 bit ("0"). User data is data needed to be delivered. Utilizing specified BCH code to encode user data can get BCH parity bits. The BCH codes, where n=63, are chosen to encode user data. Based on Table 2.1, the length of user data, k, is set to five different values: 7, 10, 16, 24, and 30 bits. The error-correction capabilities of these BCH codes are 15, 13, 11, 7, and 6 bits length burst errors, respectively. Based on the CAN protocol, the length of the Data Field needs to be an integral number of bytes. Thus, 1 bit logical "0" is added to make the Data Field 64 bits (8 bytes).

Start of frame (1 bits)	Arbitration field (31 bits)	Control field (7 bits)	Data field (64 bits)	CRC field (15 bits)	Ack field (3 bits)	End of frame (7 bits)

Figure 4.1 The CAN data frame strucure

Start of frame (1 bits)	Arbitration field (31 bits)	Control field (7 bits)	Data field (64 bits)			CRC field (15 bits)	Ack field (3 bits)	End of frame (7 bits)
			User data (7, 10, 16, 24, or 30 bits)	BCH Parity bits (56, 53, 47, 39, or 33 bits)	1 bit logic "0"			

Figure 4.2 The Modified ARQ CAN data frame strucure

4.2. IMPLEMENTATION OF MODIFIED ARQ SCHEME IN CAN

The Modified ARQ CAN is implemented in Matlab simulations. The communication flowchart is given by Figure 4.3.

As shown in Figure 4.3, the dash line represents the flow of Modified ARQ CAN, while the solid line represents the flow of original CAN. In order to implement Modified ARQ CAN, random data are generated and k bits ('k' in the BCH code) are selected as user data each time. A BCH encoder is used to encode k bits user data and output n bits encoded user data. Encoded user data are added one bit logic "0", and put into the Data Field. Then the

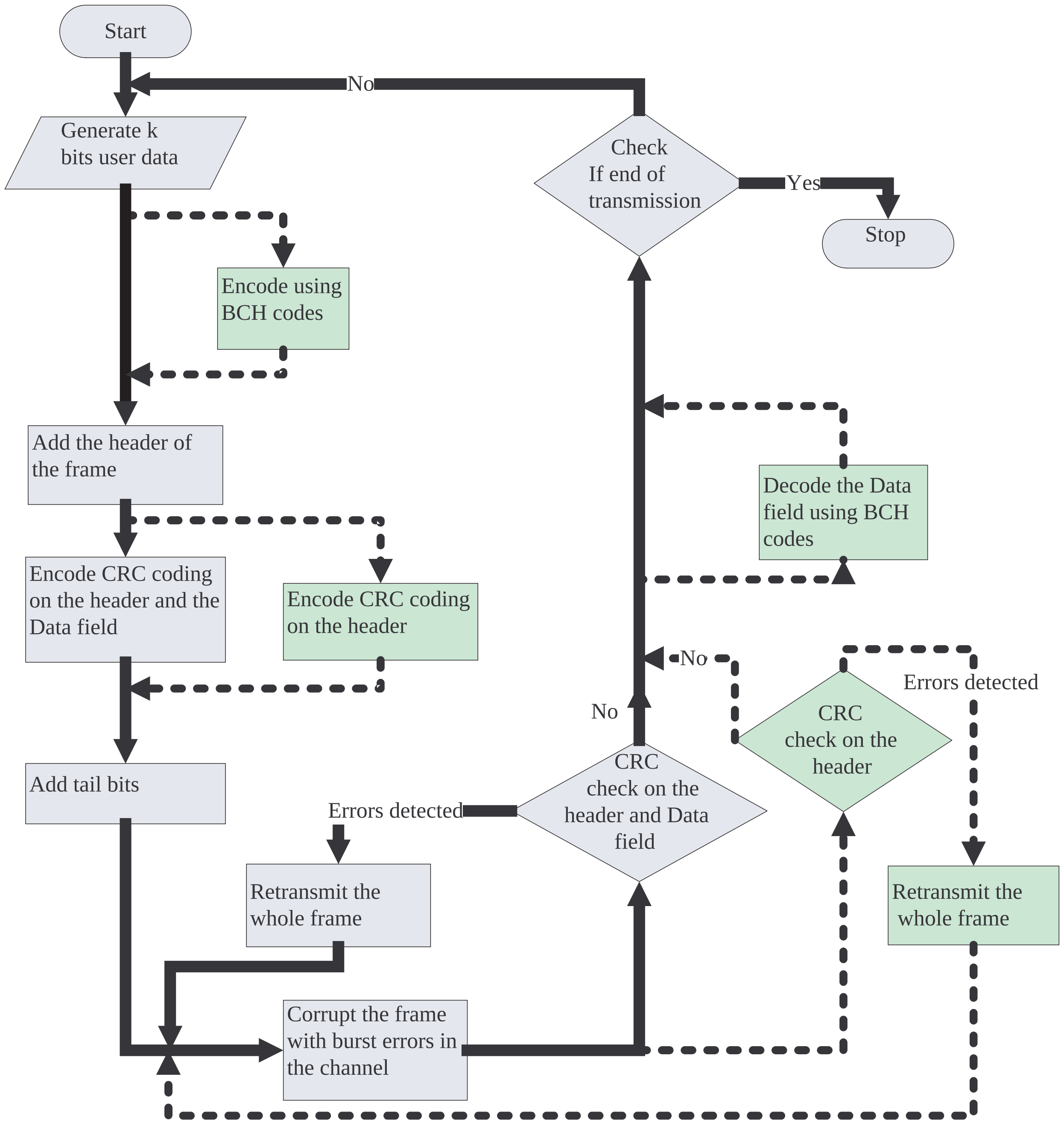

Start
No
Generate k bits user data
Check If end of transmission
Yes
Stop
Encode using BCH codes
Add the header of the frame
Decode the Data field using BCH codes
Encode CRC coding on the header and the Data field
Encode CRC coding on the header
No
Errors detected
No
CRC check on the header
Add tail bits
Errors detected
CRC check on the header and Data field
Retransmit the whole frame
Retransmit the whole frame
Corrupt the frame with burst errors in the channel

Figure 4.3 The flowchart of implementing Modified ARQ CAN

Start of Frame, the Arbitration Field and the Control Field are added to the Data Field. The CRC, which is calculated on the Start of Frame, the Arbitration Field and the Control Field, is appended. Lastly, the ACK Field and the End of frame are added to finish the Modified ARQ CAN data frame.

In transmission channel, some bits in the data frame are corrupted by burst errors caused by EMI. When the data frame reaches the receiver, the CRC is computed and compared to the

CRC received to check errors. Differences between two CRC bits indicate errors are detected in the Start of Frame, the Arbitration Field or the Control Field. Next, the whole data frame is retransmitted. If two CRCs are identical, the data frame is accepted and the Data Field is input to a BCH decoder. The output of the BCH decoder is user data.

4.3. MATLAB SIMULATIONS FOR MODIFIED ARQ CAN

Hardware measurement results on CAN test bed are the basis for configuring Matlab simulation parameters. After analyzing waveforms of CAN data frame with EMI, it is determined that with unshielded cables and the data rate of 1 Mbps, the length of burst errors in each CAN data frame is 5 bits in average. In hardware experiments, only one magnetic relay is utilized to generate EMI and the switch changes at most 3 times per second. In an actual car, the number of magnetic relays can be roughly one hundred. The switch changes faster too. Thus, the actual EMI environment is more serious than that of hardware experiments. Simulations are designed to implement original CAN and Modified ARQ CAN in a serious EMI environment. In simulations, EMI is added to each data frame, and causes several bits burst errors on each data frame. Because the length of CAN data frame is 128 bits and the duration time of each bit is 1 us (the data rate is 1 Mbps), the time interval between two EMI is 128 us. The Signal Noise Ratio (SNR) in simulations is 10 dB. Total 10000 CAN data frames are generated for transmission. Data are transmitted between two nodes: one transmitter and one receiver. In each simulation process, the length of burst error is fixed. When length of burst errors varies from 1 to 15 bits, simulation processes repeat 15 times.

When the retransmission functions of CAN and Modified ARQ CAN are disabled. The bit error rate and the frame error rate are computed. When the retransmission function is enabled, corrupted data frames are retransmitted until correct data frames are received. The bit error rate and the frame error rate of CAN and Modified ARQ CAN both decrease to a low level. But the number of retransmissions for successful delivering of one frame are different in CAN and Modified ARQ CAN. The average number of retransmissions for successful delivering of one frame is computed dividing the number of retransmitted frames by the number of frames need to be delivered. Original CAN retransmits much more corrupted frames than Modified

ARQ CAN do. More time needs to be taken to transmit the same length data, and the actual data rate decreases.

4.4. RESULTS AND ANALYSIS FOR MODIFIED ARQ CAN

The bit error rate of CAN and Modified ARQ CAN with different burst errors are given in Figure 4.4, when retransmission is disabled. The bit error rate of original CAN increases linearly. Utilizing the error-correction code, the bit error rate of the Modified ARQ CAN decrease significantly. When the length of burst errors is 3, 4, 5, and 6 bits, bit error rates of all the Modified ARQ CAN are roughly 50% of those of the original CAN. Moreover, when the length of burst errors increases, bit error rates of Modified ARQ increase at a much slower pace than those of original CAN.

As observed in Figure 4.4, when Modified ARQ CAN employs a BCH code with smaller k, it has a lower bit error rate. The reason can be found in Table 2.1. When n is fixed, the

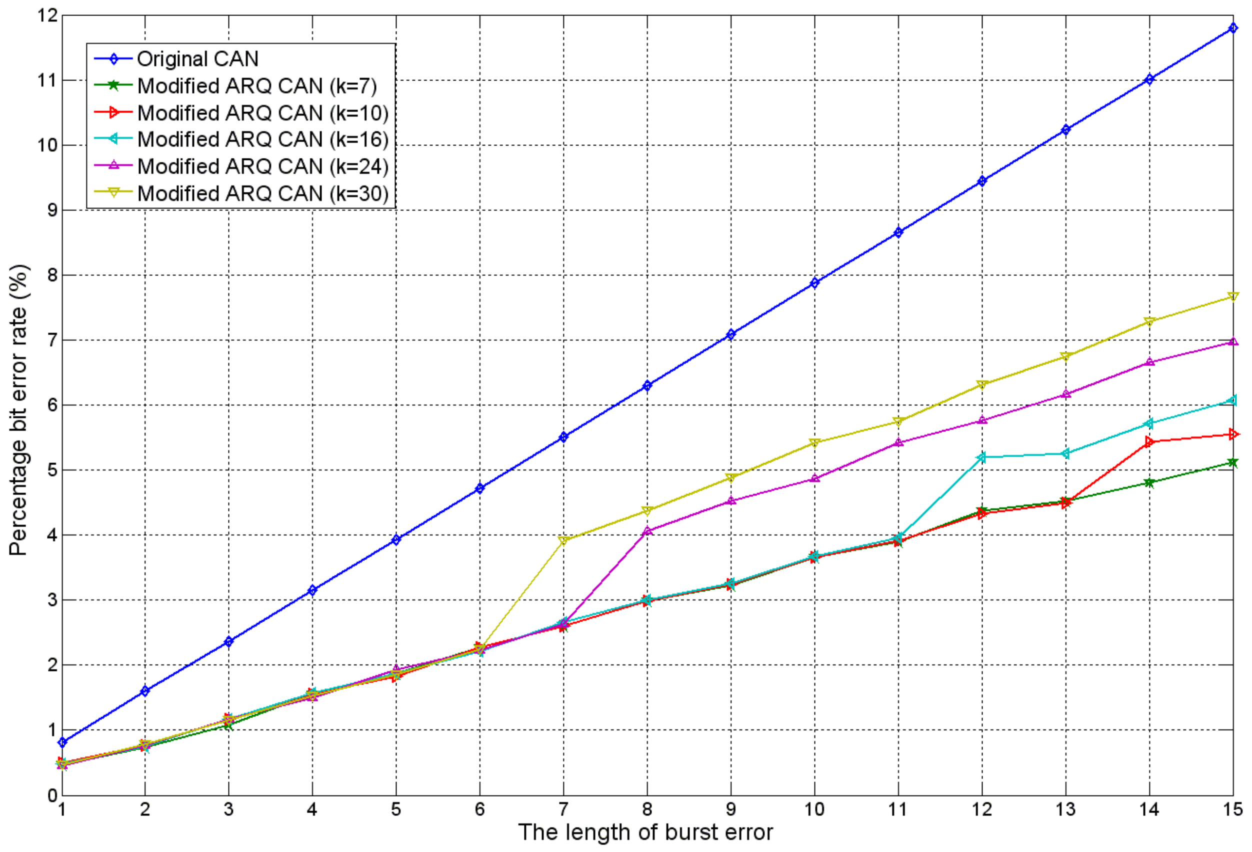


Figure 4.4 The bit error rate of CAN and Modified ARQ CAN

BCH code with smaller k has stronger error-correction capability than the BCH code with bigger k does. Another fact is that when the length of burst errors go beyond error-correction capability of BCH codes, bit error rates drastically change. For example, the BCH code with k=7 has error-correction capability, 6 bits. The BCH code can correct, at most, 6 bits burst errors. Thus, when the length of burst errors change from 6 to 7 bits, the bit error rate jumps from 2.1% to 4%. Moreover, Figure 4.4 shows that when the length of burst errors is 15 bits, the Modified ARQ CAN with the BCH code (n=63, k=7) reduces the bit error rate from 11.9% to 5.1%.

Figure 4.5 shows the frame error rate of CAN and Modified ARQ CAN in various burst errors length when retransmission is disabled. CAN communication is based on frame trans-

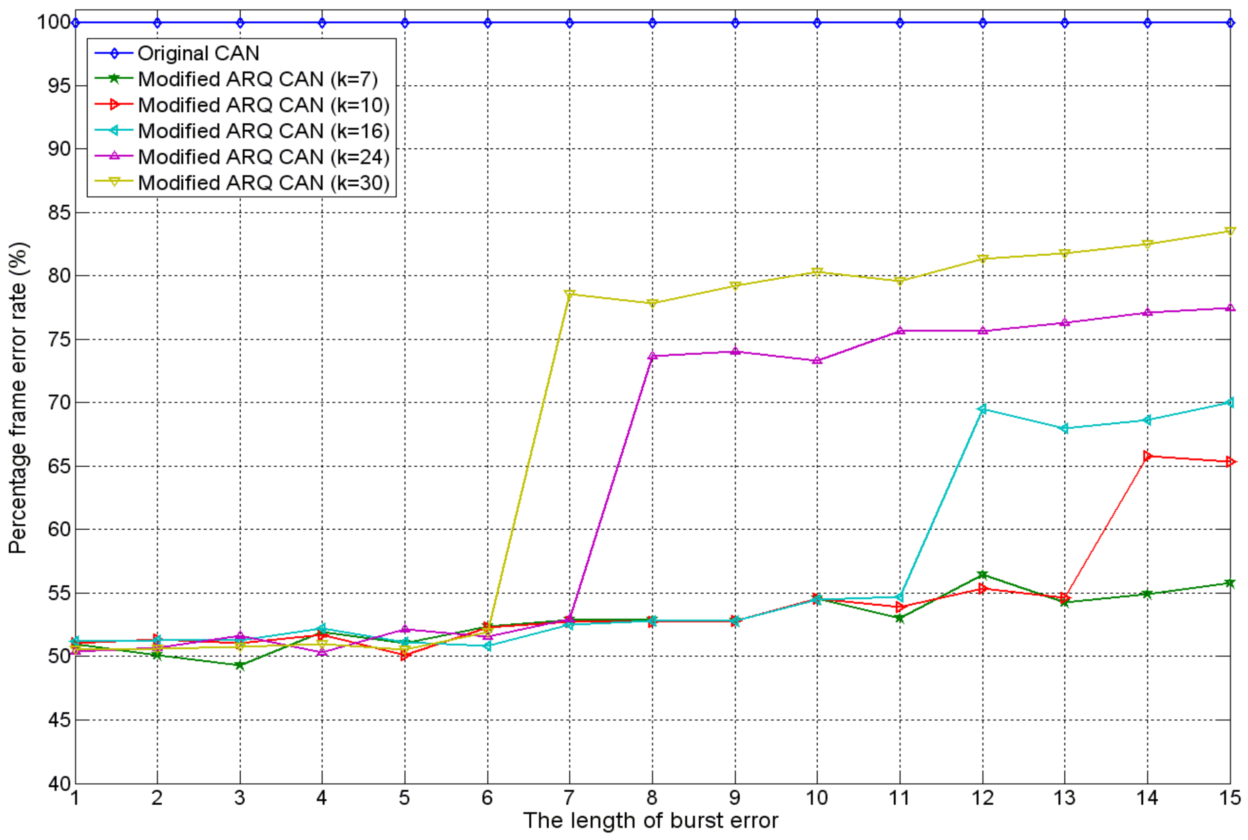


Figure 4.5 The frame error rate of CAN and Modified ARQ CAN

mission. One bit error in a data frame can cause the retransmission of a whole data frame (128 bits). Thus, the frame error rate can reflect CAN performance improvements too.

As shown in Figure 4.5, when burst errors are added to each CAN data frame, errors are detected by all CAN data frames. Thus, original CAN maintains the frame error rate 100%. In Modified ARQ CAN, the frame error rate decreases significantly. When the length of burst errors are 3, 4, 5, and 6 bits, the frame error rates of all Modified ARQ CAN are roughly 52%, half the amount of the original CAN. When Modified ARQ CAN employs a BCH code with smaller k, it can reach a lower frame error rate. When the length of burst errors goes beyond error-correction capability of the BCH codes, the frame error rate drastically increases. When the length of burst errors is 15 bits, the Modified ARQ CAN with the most powerful BCH code (n=63, k=7) can maintains the frame error rate 50%.

The number of retransmissions for successful delivering of one frame is shown in Figure 4.6.

In simulations, EMI is added to each data frame, and causes several bits burst errors on each data frame. In a such serious EMI environment, when retransmission function of CAN

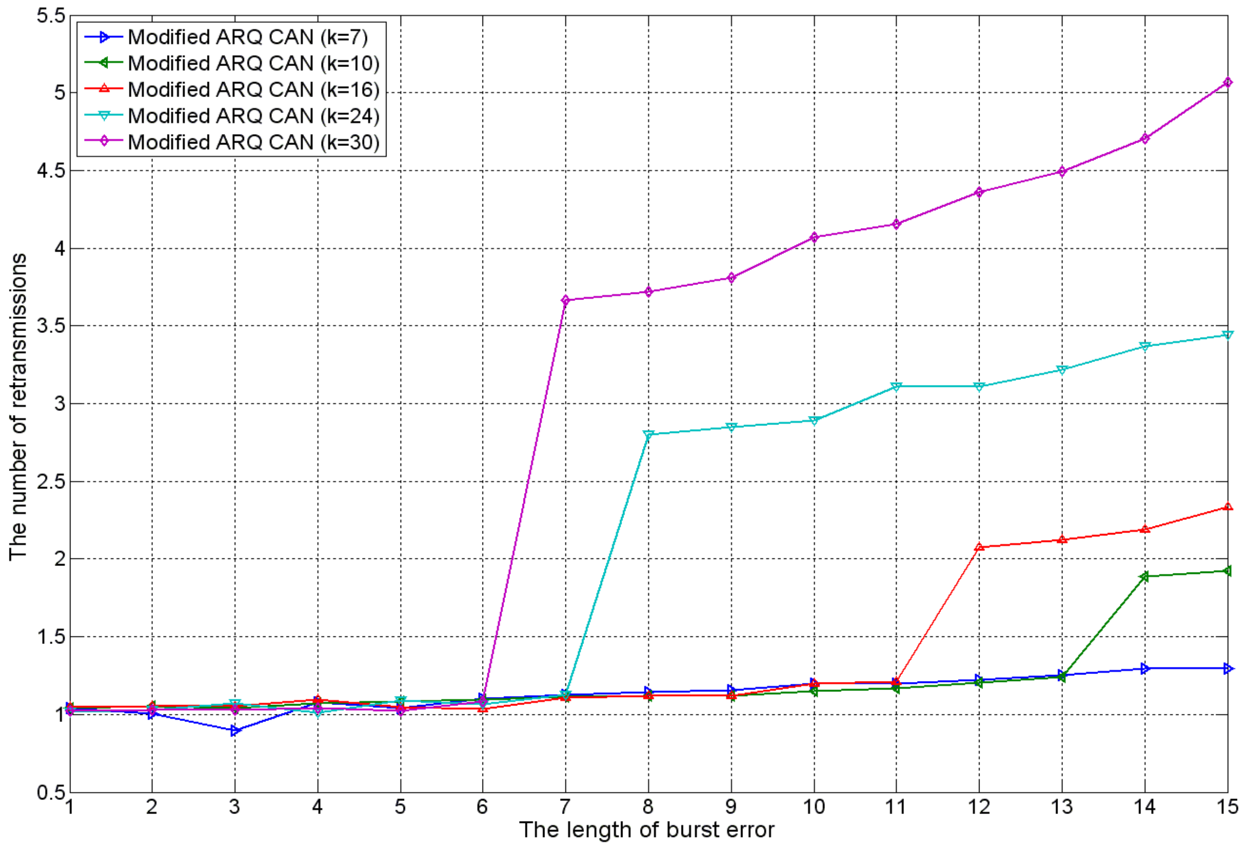


Figure 4.6 The number of retransmissions for successful delivering of one frame in Modified ARQ CAN

and Modified ARQ CAN is enabled, retransmitted data frames are corrupted, CAN cannot successfully deliver data frames. But Modified ARQ CAN can successful deliver data frames. Shown as in Figure 4.6, the Modified ARQ CAN with the BCH code (k=7) retransmits 1 to 1.3 times to successful deliver one data frame. The number of retransmissions drastically changes when the length of burst errors is up to error-correction capability of the BCH code. In actual applications, the length of burst errors caused by EMI can be measured before choosing the proper BCH code. With the proper BCH code, the number of retransmissions for successful delivering of one data frame can be maintained at a low level.

5. EMI MITIGATION METHODS FOR FLEXRAY

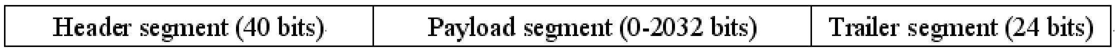
Referred to Section 2.2.4, FlexRay employs two basic mechanisms for responding to significant errors and errors that can be endured for a limited period of time, respectively. For significant errors detected by a node, the “POC: halt” state is immediately entered. For errors that can be endured for a limited period of time, FlexRay also contains a three-state degradation model to defer entry of the “POC:halt” state, at least temporarily, and to support possible recovery from a potentially transient condition.

Bit or burst errors caused by EMI can be detected by CRC and belong to significant errors. After entering “halt” state, the ARQ scheme and Modified ARQ scheme can be implement by application programs.

5.1. IMPLEMENTATION OF MODIFIED ARQ IN FLEXRAY

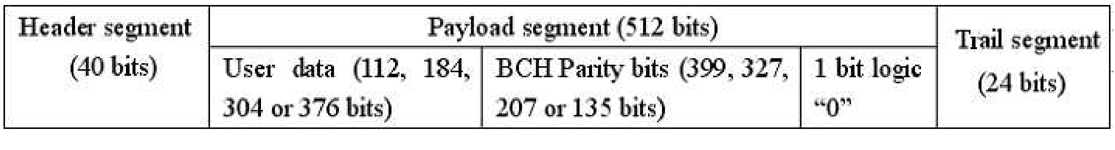
This section describes Matlab simulations on ARQ FlexRay based on the Stop-and-wait ARQ scheme and Modified ARQ FlexRay based on the Modified ARQ scheme.

In order to implement Modified ARQ in FlexRay, new FlexRay data frame structure employing error-correction methods is designed. The simplified FlexRay data frame structure and the Modified ARQ FlexRay data frame structure is given in Figure 5.1 and Figure 5.2.



Header segment (40 bits)	Payload segment (0-2032 bits)	Trailer segment (24 bits)

Figure 5.1 The simplified FlexRay data frame structure



Header segment (40 bits)	Payload segment (512 bits)			Trail segment (24 bits)
	User data (112, 184, 304 or 376 bits)	BCH Parity bits (399, 327, 207 or 135 bits)	1 bit logic “0”	

Figure 5.2 The Modified ARQ FlexRay frame structure

The length of the Payload segment is configured to 512 bits so that the BCH codes (n=511) can be chosen as the error-correction code. Table 2.2 shows parameters of the BCH codes. In the Modified ARQ FlexRay data frame, modifications are similar to these of Modified ARQ CAN. There is no any modification in the Header segment. The Payload segment is separated and consists of three parts: user data, BCH parity bits, and 1 bit logical "0". BCH parity bits are generated by encoding user data with the BCH code. In FlexRay protocol, the length of the Payload segment needs to be an integral number of bytes. Thus, 1 bit logical "0" is added to make the Payload segment 512 bits (64 bytes).The CRC in the Trail segment is computed on the Header segment. In the receiver, it is computed again for checking errors occurring in the Header segment.

The flowchart of implementing Modified ARQ FlexRay is given in Figure 5.3

Shown in Figure 5.3, the solid line represents the flow of implementing ARQ FlexRay, while the dash line represents that of implementing Modified ARQ FlexRay. Modified ARQ FlexRay is implemented by following steps. Random k bits user data are generated for transmission. The BCH code is used to encode k bits user data, and output n bits encoded user data. Encoded user data appends one bit logic "0", and are put into the Payload segment. Then the Header segment is added to the Payload segment. The CRC in the Trail segment is computed on the Header segment and is appended. The Modified ARQ data frames is encoded by FlexRay encoding process before transmitted. In the channel, the data frame is corrupted by burst errors caused by EMI. When the Modified ARQ FlexRay data frame reach the receiver, it is decoded by FlexRay decoding process. The CRC is computed on the received Header segment, and compared to the received CRC. If the two CRCs are different, errors are detected. The data frame is retransmitted. If two CRCs are identical, the data frame is accepted, and the Payload segment is decoded by the BCH code to get user data. The flow of implementing ARQ FlexRay does not have BCH encoding and decoding processes, and the CRC in the Trail segment checks both the Header segment and the Payload segment.

5.2. MATLAB SIMULATIONS FOR MODIFIED ARQ FLEXRAY

Parameter setups of Matlab simulations are based on results of CAN hardware experiments. In Matlab simulations, the SNR is set to 10 dB. 10000 Modified ARQ FlexRay data

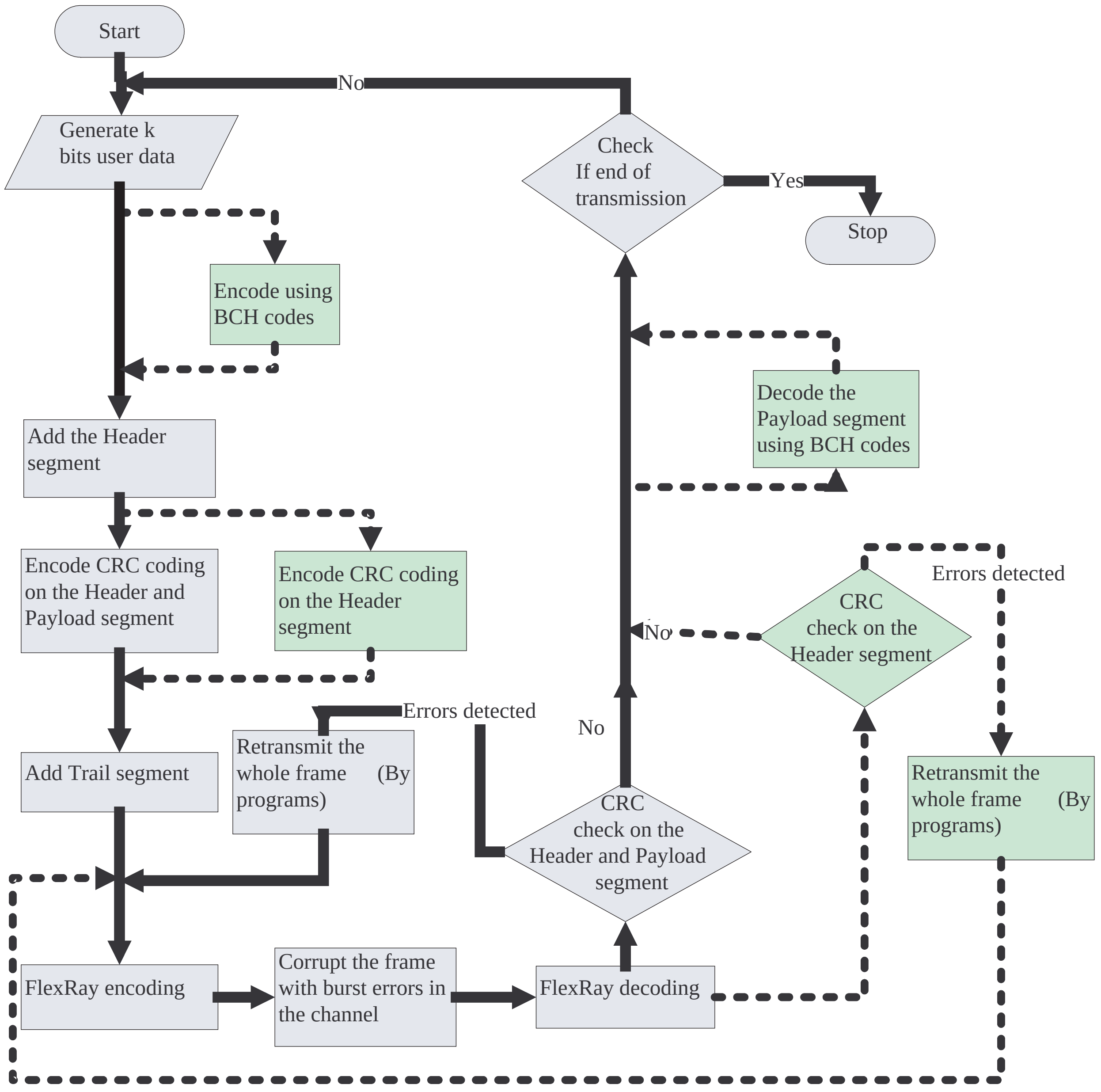

Start
No
Generate k bits user data
Check If end of transmission
Yes
Stop
Encode using BCH codes
Decode the Payload segment using BCH codes
Add the Header segment
Encode CRC coding on the Header and Payload segment
Encode CRC coding on the Header segment
Errors detected
CRC check on the Header segment
No
Errors detected
No
Retransmit the whole frame (By programs)
Add Trail segment
CRC check on the Header and Payload segment
Retransmit the whole frame (By programs)
FlexRay encoding
Corrupt the frame with burst errors in the channel
FlexRay decoding

Figure 5.3 The flowchart of implementing Modified ARQ FlexRay

frames and 10000 ARQ FlexRay data frames are generated and transmitted in EMI environment. The length of Payload segment in all data frames is configured to 512 bits. In each 100 us, one burst error is added to transmission. The length of burst errors is fixed in each simulation, and the bit error rate and the frame error rate are computed after each simulation. The simulation is repeated when the length of burst errors increases. As mentioned in the previous section, the bit rate of FlexRay communication is 10 Mbps, which is ten times faster than that of CAN. The duration time of each bit (0.1 us) in FlexRay is one tenth of that of each CAN bit (1 us). The same extent EMI in FlexRay is assumed to corrupt ten times more bits than it does in CAN. Thus, the maximum length of burst errors simulated increase from 15 bits to 60 bits.

5.3. RESULTS AND ANALYSIS FOR MODIFIED ARQ FLEXRAY

When retransmission is disabled, the bit error rate in ARQ FlexRay and Modified ARQ FlexRay is given in Figure 5.4, and the frame error rate in ARQ FlexRay and Modified ARQ FlexRay are given in Figure 5.5.

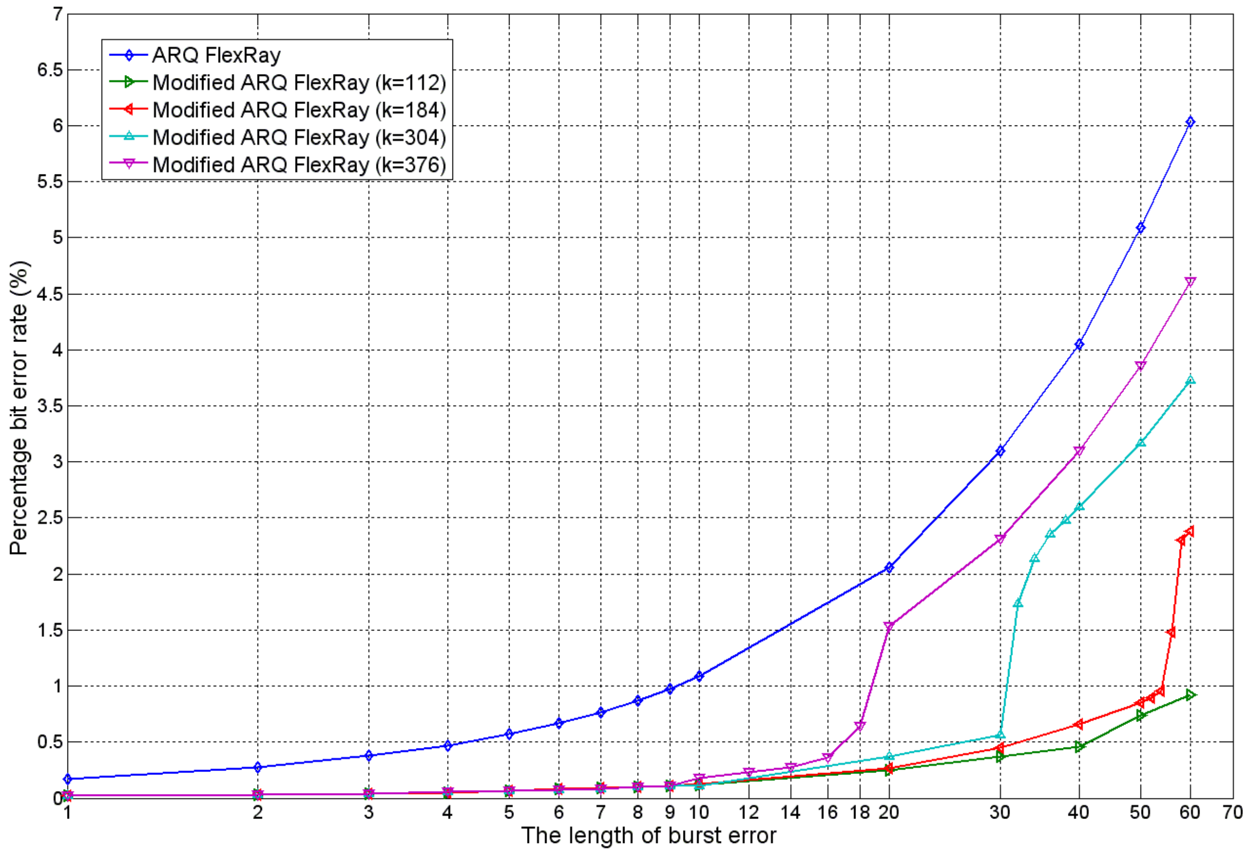


Figure 5.4 The bit error rate of ARQ FlexRay and Modified ARQ FlexRay

Shown in Figure 5.4, the bit error rate of Modified ARQ FlexRay is lower than that of ARQ FlexRay. The bit error rate of Modified ARQ FlexRay drastically increases when the length of burst errors goes beyond error-correction capability of the BCH codes. Modified

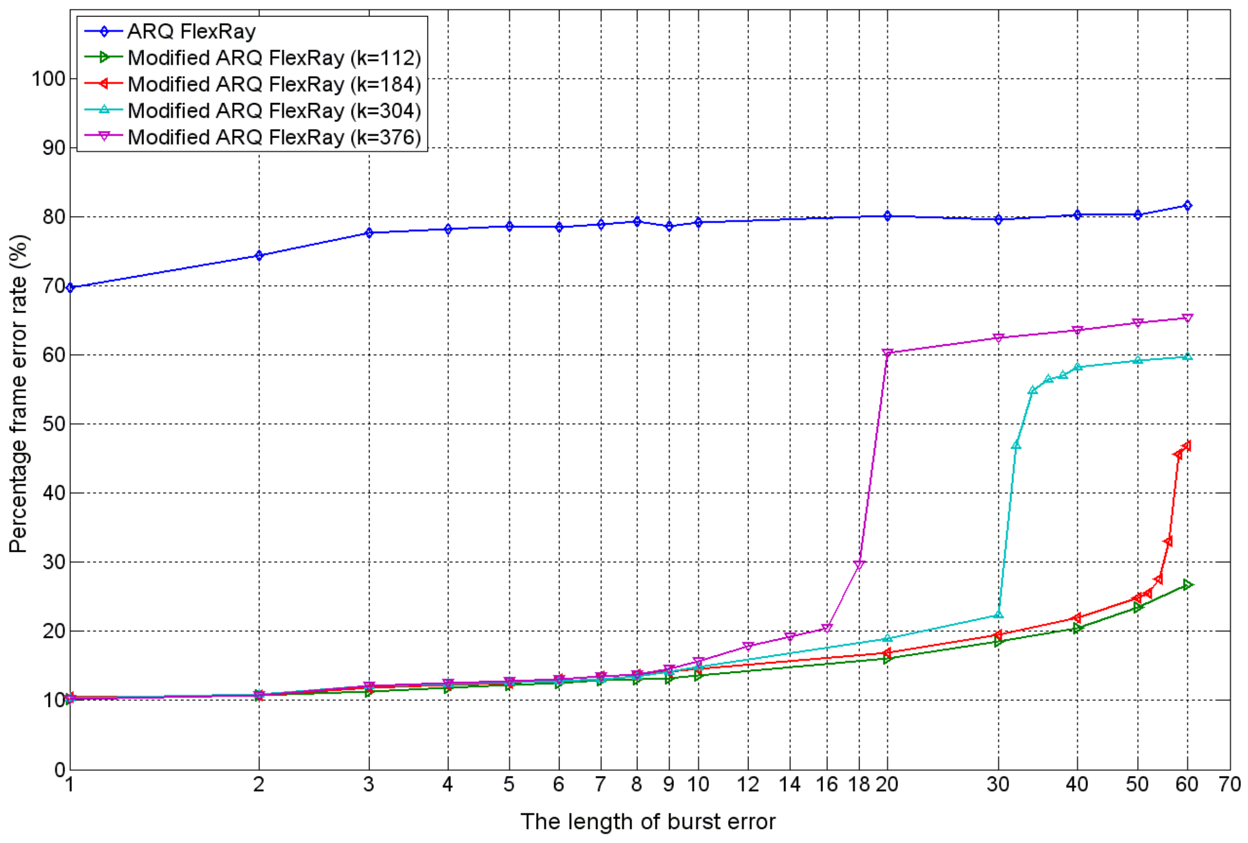


Figure 5.5 The frame error rate of ARQ FlexRay and Modified ARQ FlexRay

ARQ FlexRay with the most powerful BCH code (n=511, k=112) maintains the bit error rate 1% even when there are 60 bits burst errors in each 100 us.

FlexRay communication is based on the frame delivery. Even if one bit in a frame is incorrect, all bits in this frame should be seen as wrong and retransmitted. Thus, the frame error rate in Figure 5.5 also reflects the performance improvement of Modified ARQ FlexRay. Shown in Figure 5.5, the frame error rate of ARQ FlexRay is always more than 70%. The frame error rate indicates that more than 70% ARQ FlexRay data frames have errors. ARQ FlexRay can detect errors and retransmit corrupted frames, but retransmitted frames possibly be corrupted too. It causes a large number of retransmissions for successful delivering one data frame. In Modified ARQ FlexRay, when the length of burst errors is shorter than error-correction capability of the BCH codes, the frame error rate maintains roughly 10%. It indicates that only roughly 10% Modified ARQ data frames have errors and need to be retransmitted.

Comparing Figure 4.5 to Figure 5.5, the Modified ARQ scheme provides more improvements in performances of FlexRay than these of CAN. The main reason is that the FlexRay data frame has longer user data than the CAN data frame has, when the overhead bits of CAN and FlexRay are the same in length. The Modified ARQ scheme employs the BCH codes correct errors which occur in user data, but not correct the overhead bits. Thus, actually, the FlexRay data frame get more protection from the BCH codes. Utilizing Modified ARQ scheme provides more benefits to communication protocols, which have long user data and short overhead bits in frame structure.

When the retransmission is enabled, the The average number of retransmissions for successful delivering of one frame is given in Figure 5.6.

The average number of retransmissions for successful delivering of one frame can be computed dividing the number of retransmitted frames by the number of frames need to

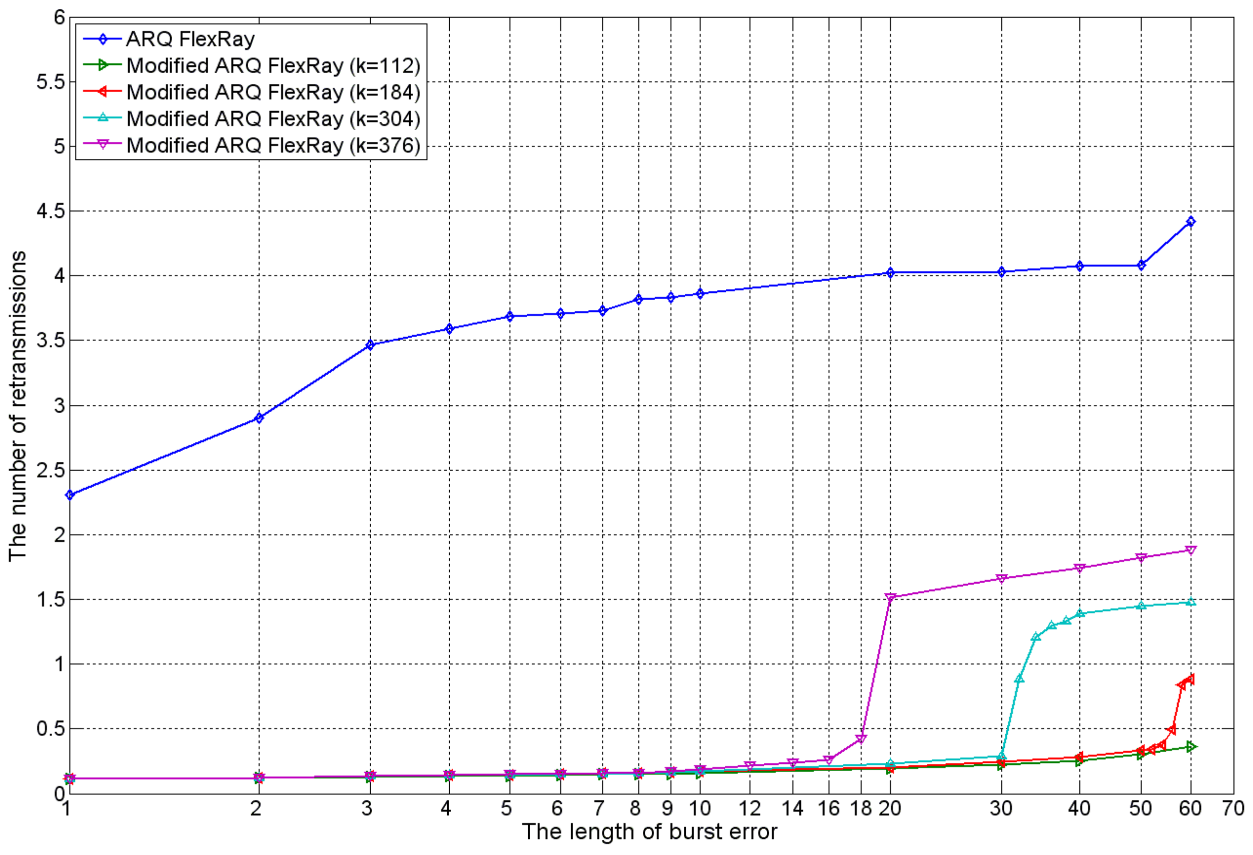


Figure 5.6 The number of retransmissions for successful delivering of one frame in Modified ARQ FlexRay

be delivered, and it possibly is not an integer. Shown in Figure 5.6, when the length of burst error is 60 bits, ARQ FlexRay retransmits 4.4 times for successfully delivering one data frame. Modified ARQ FlexRay with the BCH code (k=112) only retransmits 0.4 times for successfully delivering one frame. With other “k” values, the number of retransmissions drastically increases when the length of burst errors goes beyond error-correction capability of the BCH codes. But Modified ARQ FlexRay maintains much less retransmissions than ARQ FlexRay.

6. CONCLUSIONS

Replacing traditional unshielded parallel communication cables with shielded cables, to a large extent, combat EMI on CAN communication. Shielded cables mitigate the reduction of the data rate and the bit error rate. Measurements show that 40 meters shielded cables reduce the reduction of the data rate from more than 800 bits/second to 5 bits/second. The bit error rate reduces from 0.8510^{-3} to 0. In measurements, the length of shielded cables and the number of nodes affect the data rate a little. Moreover, with unshielded cables, CAN running in 1 Mbps receives more effects of EMI than CAN running in 503 Kbps does.

After employing the Modified ARQ scheme, performances of CAN and FlexRay in EMI environments are improved. When the Modified ARQ scheme utilizing the BCH code, which has error-correction capability beyond the maximum length of burst errors, the Modified ARQ scheme effectively reduces the bit error rate, the frame error rate and the number of retransmissions. In a CAN two-node network, when there are 12 bits burst errors per 100us, the Modified ARQ scheme with the BCH code (n=63, k=7) reduces the bit error rate from 11.8% to 5%, and the frame error rate from 100% to 56%. The number of retransmissions for successful delivering of one data frame reduces from infinite to 1. In a FlexRay two-node network, when there are 60 bits burst errors per 100us, the Modified ARQ scheme with the BCH code (n=511, k=112) reduces the bit error rate from 6% to 0.9%, and the frame error rate from 82% to 27%. The number of retransmissions for successful delivering of one data frame reduces from 4.4 to 0.4. Moreover, the FlexRay data frame has longer user data in the data frame than the CAN data frame does. With the Modified ARQ scheme, more user data in FlexRay are protected by error-correction codes. Thus, FlexRay gets more performance improvements than CAN does from utilizing the Modified ARQ scheme.

BIBLIOGRAPHY

[1] “LIN bus description,” [Online]. Available: http://www.interfacebus.com/Design_Connector_LIN_Bus.html. Oct 2007.

[2] Wikipedia, “Controller area network,” [Online]. Available: http://en.wikipedia.org/wiki/Controller Area Network. Oct 2007.

[3] C. Temple, “Flexray - past present future perfect,” [Online]. Available: http://www.flexray.com/publications/1_FlexRay.pdf. Oct 2006.

[4] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, “Trends in automotive communication systems,” *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1204–1223, 2005.

[5] “Automotive buses,” [Online]. Available: http://www.interfacebus.com/Design_Connector_Automotive.html. Oct 2007.

[6] W. Xing, H. Chen, and H. Ding, “The application of controller area network on vehicle,” in *International Vacuum Electronics Conference (IVEC)*, pp. 455–458, 1999.

[7] H. Kopetz, “Fault containment and error detection in the time-triggered architecture,” in *Autonomous Decentralized Systems, 2003. ISADS 2003. The Sixth International Symposium on*, pp. 139–146, 2003.

[8] K. C. S. Emani, *“Appication of Hybrid ARQ to Controller Area Networks”*. MS thesis: University of Missouri - Rolla, 2007.

[9] The Bosch IC Design Center, “CAN specification version 2.0,” [Online]. Available: http://www.semiconductors.bosch.de/pdf/can2spec.pdf. Oct 2006.

[10] Kvaser company, “The CAN protocol tour,” [Online]. Available: http://www.kvaser.com/can/protocol/index.htm. Oct 2007.

[11] Wikipedia, “Differential signaling,” [Online]. Available: http://en.wikipedia.org/wiki/Differential_signaling. Oct 2007.

[12] Wikipedia, “FlexRay,” [Online]. Available: http://en.wikipedia.org/wiki/Flexray. Oct 2007.

[13] R. Makowitz and C. Temple, “Flexray - a communication network for automotive control systems,” in *2006 IEEE International Workshop on Factory Communication Systems*, pp. 207–212, 2006.

[14] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, “Timing analysis of the flexray communication protocol,” in *18th Euromicro Conference on Real-Time Systems, 2006*, p. 11 pp., 2006.

[15] The FlexRay Consortium, “FlexRay communications system protocol specification version 2.1 revision a,” [Online]. Available: http://www.flexray.com/. Oct 2006.

[16] The FlexRay Consortium, “FlexRay communication system preliminary node-local bus guardian specification version 2.0.9,” [Online]. Available: http://www.flexray.com/. Oct 2007.

[17] Vector company, "FlexRay," [Online]. Available: http://www.vectorworldwide.com/vi_flexray_kommunikationsprinzip_en,,223.html. Oct 2007.

[18] Wikipedia, "Automative repeat request," [Online]. Available: http://en.wikipedia.org/wiki/Arq. Oct 2007.

[19] K. C. Emani, K. Kam, M. Zawodniok, Y. R. Zheng, and J. Sarangapani, "Improvement of CAN bus performance by using error-correction codes," in *IEEE Region 5 Technical, Professional, and Student Conference (TPSC)*, (Fayetteville, AK), 2007.

[20] Wikipedia, "The BCH code," [Online]. Available: http://en.wikipedia.org/wiki/BCH_code. Oct 2007.

[21] E. R. Ziemer and L. R. Peterson, *Introduction to Digital Communication.* Englewood Cliffs, New Jersey 07632: Prentice Hall, 2000.

[22] "C-CAN user's manual revision 1.2," The Bosch IC Design Center, Reutlingen, Germany. Robert Bosch GmbH. [Online]. Available: http://www.semiconductors.bosch.de/pdf/Users_Manual_ C_CAN.pdf. Oct 2006.

VITA

Fei Ren was born on May 8th, 1983, in Chongqing, China. He received his Bachelor's degree in Electronics Engineering from the University of Electronic Science and Technology of China, Chengdu, China in 2005. He received his MS degree from the University of Missouri-Rolla in December 2007, then pursued a Ph.D degree also at the University of Missouri-Rolla.