Development of CAN Bus Application Layer Protocol for Beam Control System*

Wang Bangji, Liu Qingxiang, Yu Yi, Zhao Liu, Zhang Zhengquan, Li Xiangqiang, Zhang Jianqiong

Southwest Jiaotong University, Chengdu, Sichuan, 610031, China
bangjiw@yeah.net

***Abstract*—A special Controller Area Network (CAN) bus application layer protocol is designed for the high reliable and high real-time control network of the beam control system. In this paper, messages on the network are classified and the identifier coding, data coding, network management mechanism and physical interface are defined. Then network load analysis for the real-time control network is introduced. At last a test platform based on CAN bus for the beam control system is built to validate the design. Experiment results indicate that the developed CAN control network for the beam control system is very reliable and has good real-time performance and low network load rate.**

Keywords-Beam Control System; CAN Bus; Application Layer Protocol; Network Load Analysis

I. INTRODUCTION

The beam control system can control antenna beam to desired spatial orientation. Due to the ability of antenna beam's rapid scanning, the beam control system needs a high reliable and high real-time control network [1]. CAN bus is a serial communications protocol which efficiently supports distributed real-time control with a very high level of reliability. So CAN bus can meet control requirements of the beam control system. But CAN protocol only comprise physical layer and data link layer, not application layer [2-4]. Therefore, the application layer protocol can be self-defined, or a standard protocol of international organization such as CANOpen or SAE J1939 is chosen for the specific system. The two standard protocols don't fit for the high reliable and high real-time control network because of their complex structures. A simple and practical CAN application layer protocol which can meet control requirements of the beam control system is designed, based on the study of CAN 2.0B standard and the reference of CANOpen specifications [4-5].

II. ARCHITECTURE OF CAN CONTROL NETWORK

The CAN control network of the beam control system, illustrated in figure 1, comprises a master computer and a number of slaves. The master computer, an industrial PC, is connected into the CAN control network via a PCI to CAN adapter. The slave which is a DSP2812 system has a CAN bus interface. A slave can control a number of antenna elements.

The master (short for master computer) receives control commands or control parameters via human machine interface, and then packages and sends a broadcast message to all the salves. The master also needs to send a point to point configuration parameters message to the slave. In addition, the master has other functions such as the network management, fault diagnosis and information display.

After receiving control command, the slave performs corresponding action, and then reports the result to the master. After the slave periodically receiving control parameters, objective parameters are calculated. The slave controls its internal antenna elements to achieve the objective parameters. The slave also monitors real-time status of its internal elements and reports to the master if there are faults. Data sharing among slaves does not be required. Therefore, if a slave fails other slaves still work.

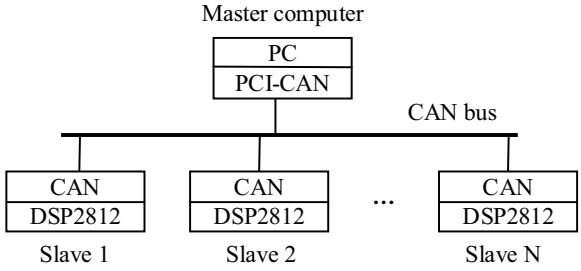

Figure 1. Architecture of CAN control network for the beam control system

III. APPLICATION LAYER PROTOCOL OF CAN CONTROL NETWORK

CAN 2.0B standard contains two message formats: standard frame of 11 bit identifier and extended frame of 29 bit identifier. For every message format, there are four different frame types: data frame, remote frame, error frame and overload frame. The former two are most common. The CAN extended data frame format is illustrated in figure 2.

CAN Extended Data Frame												
	Arbitration Field					Control Field		Data Field	CRC Delimiter			
SOF	Identifier	SRR	IDE	Identifier Ext	RTR	R	DLC	Data Field	CRC	▼	ACK Field	EOF
1	11	1	1	18	1	2	4	0-64	15	1	2	7

Figure 2. CAN extended data frame format

*This work is supported by the Fundamental Research Funds for the Central Universities (SWJTU09ZT38).

978-0-7695-4353-6/11 $26.00 © 2011 IEEE
DOI 10.1109/ICICTA.2011.415

The design goal of the CAN application layer protocol is to maximize excellent performance of CAN bus, make communication more standardized and reliable, improve real-time performance and reduce network load rate [2]. This protocol using extended frame of 29 bit identifier includes the communication content, identifier coding, data coding, network management mechanism and physical interface. The basic idea of the protocol is expounded from the following five aspects.

A. Communication content

The protocol provides all the communication contents of the CAN control network. Based on control strategy of the beam control system, the protocol provides all the communication contents, including the control commands, control parameters, configuration parameters, status information and other required messages, for the master and every slave. In addition, the extended space is reserved for future use.

B. Identifier coding

The CAN permits any node to transmit a message on the network when the bus is idle. Collisions are avoided due to the arbitration process that occurs while the identifier is transmitted (using a non-destructive arbitration scheme). This permits high priority messages to get through with low latency times because there is equal access on the network for any node, but when multiple nodes are simultaneously attempting to transmit, the highest priority message prevails. In the protocol, 29 bit identifier consists of six fields: priority, parameter group, reserved bits, destination address, source address and internal element number. The allocation scheme of the 29 bit identifier is illustrated in figure 3.

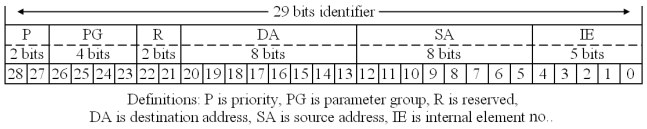

Figure 3. Allocation scheme of the 29 bit identifier

1) Priority (P)

The first two bits of the 29 bit identifier are only used to optimize message latency for transmission onto the bus and should be globally masked off by the receiver. The priority of any message can be set from highest, 0 (00_2), to lowest, 3 (11_2). The default value is 1 (01_2) for control oriented messages and 2 (10_2) for all other messages. The priority can be raised or lowered in the future as bus traffic changes or new contents are assigned.

2) Parameter group (PG)

The next four bits are used to identify the function of the message. This field, in conjunction with the two bits of the priority field, provides for up to 64 parameter groups. It is enough for the CAN control network.

3) Reserved Bits (R)

The next 2 bits are currently reserved for future use. All messages should set the reserved bits to ZERO on transmit.

4) Destination address (DA)

The destination address field is 8 bits long. This field defines the specific address (from 0 to 254) to which the message is being sent. Any other device should ignore this message. The global destination address (255) requires all the devices to listen and respond accordingly as message recipients.

5) Source address (SA)

The source address field is also 8 bits long. Every node on the network should have an exclusive address (from 0 to 254) so that the source address assures that the CAN identifier is unique, as required by CAN.

6) Internal element number (IE)

The last five bits of the 29 bit identifier are used to identify the internal element of the slave, to which the message relates.

C. Data coding

The CAN using a short frame structure, 8 types or less data are enough for the CAN control network. This short frame structure doesn't take the bus a long time for every transmission, ensuring real-time communication. In the process of data packing, the relevant data can be packaged into one frame in order to ensure efficiency of data transmission and make full use of bus bandwidth.

In this protocol, the control commands and status information are represented by bit sequences of length 8, while the control parameters and configuration parameters are represented by single precision floating-point number which is bit sequences of length 32. The coding of its value follows the *IEEE 754-1985 Standard* for single precision floating-point number [6]. The bit sequence $b = b_{31}b_{30}...b_0$ is assigned the value:

$$value(b) = (-1)^S 2^{E-127}(1+F)$$

Where $S = b_{31}$ is the sign bit, $E = b_{30}2^7 + \cdots + b_{23}2^0$ ($0 \leqslant E \leqslant 255$) is the un-biased exponent, and $F = 2^{-23}(b_{22}2^{22} + \cdots + b_1 2^1 + b_0 2^0)$ is the fractional part of the number.

D. Network management mechanism

In order to ensure the normal communication, the master needs to monitor and manage all the salves, and resume communication when exceptions. The master controls the states of the slave through state machine mechanism. The protocol provides all the states and the switch conditions among them of the state machine. The primary functions of the network management mechanism are those of network initialization and network error management.

1) Network initialization

After power on or hard reset, the slave autonomously enters the state of the network initialization. In the state, the master sends an ID-detect message to all the slaves. If the slave responds the ID-detect message within a particular period of time, the master will establish communication links with the slave, and initialize the network parameters. The process of the network initialization is illustrated in figure 4.

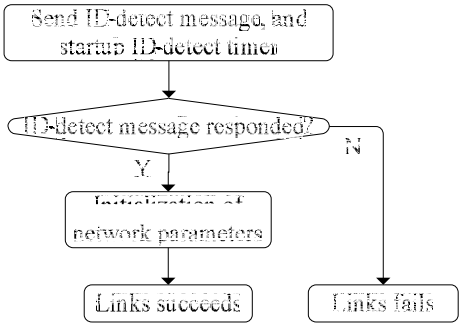

Figure 4. Process of the network initialization

2) Network error management

The network error management exists to provide a means of detecting whether the slave is on the network. In the stare, the master periodically sends a polling message to all the slaves. If the slave doesn't respond the polling message within a particular period of time, the master will consider that the slave is out of the network, and delete communication links with the slave. The slave will again enter the state of the network initialization to resume communication. The process of the network error management is illustrated in figure 5.

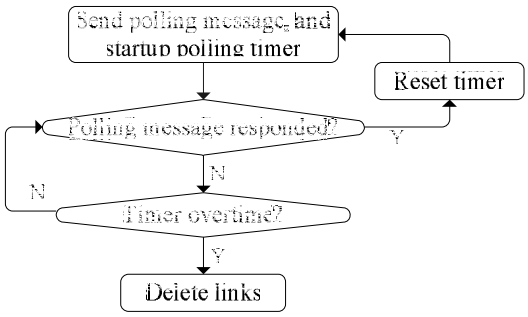

Figure 5. Process of the network error management

E. Physical interface

The CAN has powerful error correction by differential transceiver. The protocol uses shielded twisted pair as the transmission medium, and a 3-pin T-connector to interconnect the bus-lines section and CAN device. In order to minimize reflections, a 120 Ω resistor is used to terminate the bus-lines at both network ends. In addition, a RC filter circuit is used between shield ground and transceiver ground in order to increase EMC performance and to protect the transceiver chip.

IV. NETWORK LOAD ANALYSIS

Before introducing the analysis some terms are defined. The τ_{bit} is time taken to transmit a bit on CAN. The maximum transmission time (including stuff bits) is denoted as T_C. L_D is data length in bytes (from 0 to 8). T_C of a message with L_D types of data is calculated as [7-8]:

$$T_C = (8L_D + g + 10 + \frac{g + 8L_D - 1}{4})\tau_{bit}, g \in \{34,54\}$$

Where g is the number of format bit for standard and extended frame formats in CAN protocol.

The B_{load} is the network load rate on CAN bus. It can be represented by:

$$B_{load} = \frac{(n_{period} + n_{event})T_c}{1s}$$

Where n_{period} and n_{event} are the number of periodical messages and event messages within 1s respectively.

In general, CAN network load rate is limited to 30% or less in order to avoid loss of frames. The theoretical value of CAN network load rate can be calculated through the above two formulas. The values are 14.2% and 9.7% in the self-test phase and work phase respectively.

V. EXPERIMENTS

To validate the correctness of the designed CAN application layer protocol, a test platform for the beam control system via CAN bus is build. The platform mainly consists of one master, sixty-fourth slaves and a CAN analyzer. The CAN analyzer is used to measure and analyze response time of a message and network load rate on CAN bus, and also view error information on CAN bus. Time resolution of the CAN analyzer is 0.1ms. The bus baud rate in the experiment is 250kbps and control period of the beam control system is 50ms.

In a single control period, the master sends a broadcast control parameters message to all the salves and the slave who has faults responds a status information message to the master after the delay of the source address times 0.2ms. Although the control period of the periodical messages is 50ms, transmission time interval of any two consecutive frames is not identical. The transmission time interval is denoted as t_{cycle}, and is illustrated in figure 6. Statistical analysis of the t_{cycle} can give the response time jitter of the periodical messages.

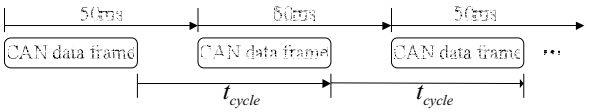

Figure 6. Transmission time interval of any two consecutive frames

The CAN analyzer records 7319 periodical messages for six minutes. Statistical distribution of the t_{cycle} is illustrated in figure 7. The t_{cycle} concentrates in the vicinity of 50ms, and the minimum and maximum of the t_{cycle} are 49.6ms and 50.5ms respectively. The standard deviation of the t_{cycle} is 0.14ms, and the response time jitter of the periodical messages is 0.9ms.

The curve of the CAN network load rate recorded by the CAN analyzer is illustrated in figure 8. The network load rates are about 15% and 10% during the self-test phase and work phase respectively. This is because there is more messages transmission on the network in the self-test phase. They are consistent with the theoretical values.

Summarily, the CAN control network has good real-time performance and low network load rate. The work of consecutive hours indicates that CAN control network is

very reliable.

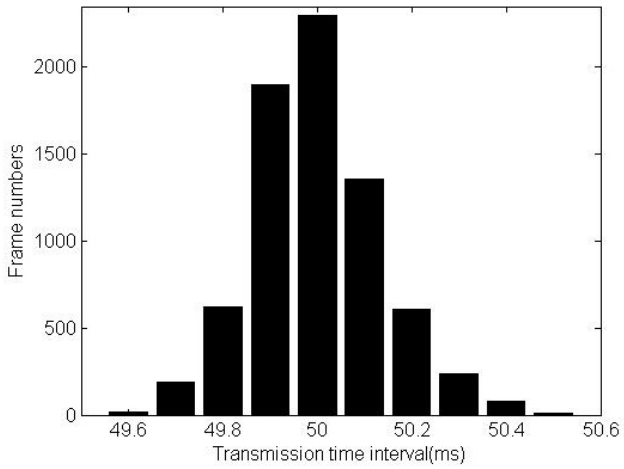

Figure 7. Statistical distribution of the t_{cycle}

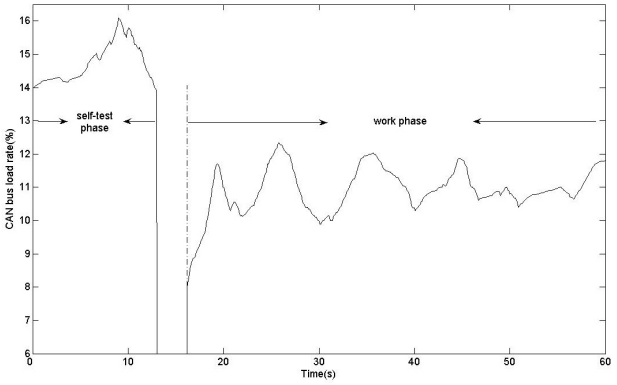

Figure 8. CAN network load rate

VI. CONCLUSIONS

The designed CAN application layer protocol for the beam control system has the following characteristics:

1) Based on the study of CAN 2.0B standard and the reference of CANOpen specifications, the designed CAN application layer protocol can meet control requirements of the beam control system and is simple, practical and easily extended.

2) The experiment results indicate that the CAN control network is very reliable and has good real-time performance and low network load rate.

3) The CAN control network has been applied to achieve a good concerted control in the beam control system.

REFERENCES

[1] Zhang Guangyi, Zhao Yujie. Phased Array Radar Technology [M]. Beijing: Publishing House of Electronics Industry, 2006. (in Chinese)

[2] Xie Hui, Zhou Nenghui, Xiao Shu. Development of the CAN Bus System for XL Pure Electric Cars [J]. Automotive Engineering, 2005, 27(6):660-664. (in Chinese)

[3] Dong Ke, Li Keqiang, Feng Nenglian. CAN and its Application for Hybrid Electric Vehicles [J]. Journal of Tsinghua University, 2003, 43(8):1130-1133. (in Chinese)

[4] CIA 301, CANOpen application layer and communication profile [S].

[5] CIA 102, CAN physical layer specification for industrial applications [S].

[6] IEEE 754:1985, Standard for binary floating point arithmetic[S].

[7] K.W.Tindell, H.Hansson, A.J.Wellings. Analysing Real-Time Communications: Controller Area Network (CAN) [J]. Real-Time Systems Symposium. 1994. 259-263.

[8] Wanke Cao, Chen Lin, Wei Zhou. A Real-Time Planning-based Scheduling Policy with CAN for Automotive Communication Systems [J]. 2009 World Congress on Computer Science and Information Engineering.