Research on the Controller Area Network

Hanxing Chen
College of Computer Science & Technology
Guizhou University
Guiyang 550025, China
e-mail:chx0618@163.com

Jun Tian
Special Equipment Inspection Laboratory
Quality & Technical Supervision Bureau
Guiyang 550002, China
e-mail:13037805229@sina.com

***Abstract*—A controller area network (CAN) is ideally suited to the many high-level industrial protocols embracing CAN and ISO 11898 as their physical layer. Its cost, performance, and upgradeability provide for tremendous flexibility in system design. This paper presents a brief introduction to the CAN operating principles, the implementation of a basic CAN bus using Texas Instrument's CAN transceivers and DSPs, and a discussion of the robust error detection and fault confinement mechanisms. Some of the properties of the enhanced Controller Area Network (eCAN) are then discussed.**

Keywords-controller area network; enhanced; protocol; message

I. INTRODUCTION

The Controller Area Network (CAN) [1] is a multi-master serial bus that uses broadcast to transmit to all CAN nodes. CAN protocol [2,3] provides advantages over other communication protocols. For example, CAN protocol offers a very good price/performance ratio. It allows moving data with a fast transmission speed (up to 1 Mbit/s) and can be implemented in real-time systems [2]. Furthermore, the data is very reliable and the error detection [4] is sophisticated and robust. CAN is very flexible and offers hot swaps.

The CAN bus was developed as a multi-master, message [2] broadcast system that specifies a maximum signaling rate of 1M bit per second (bps). Unlike a traditional network such as USB or Ethernet, CAN does not send large blocks of data point-to-point from node A to node B under the supervision of a central bus master. In a CAN network many short messages like temperature or RPM are broadcast to the entire network, which allows for data consistency in every node of the system.

II. THE CAN STANDARD

CAN is an International Standardization Organization (ISO) defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The specification calls for signaling rates up to 1 Mbps, high immunity to electrical interference, and an ability to self-diagnose and repair data errors. These features have led to CAN's popularity in a variety of industries including automotive, marine, medical, manufacturing, and aerospace.

The CAN communications protocol, ISO 11898, describes how information is passed between devices on a network, and conforms to the Open Systems Interconnection (OSI) model that is defined in terms of layers. Actual communication between devices connected by the physical medium is defined by the physical layer of the model. The ISO 11898 architecture defines the lowest two layers of the seven layer OSI/ISO model as the data-link layer and physical layer in Figure 1.

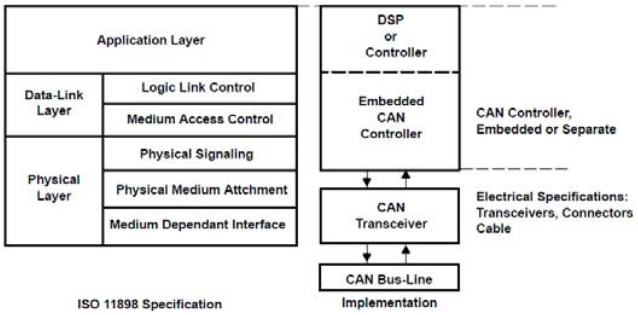

Figure 1. The Layered ISO 11898 Standard Architecture

In Figure 1, the application layer establishes the communication link [5] to an upper-level application specific protocol such as the vendor independent CANopen protocol. This protocol is supported by the international users and manufacturers group, CAN in Automation (CiA).

III. STANDARD CAN OR EXTENDED CAN

The CAN communication protocol is a carrier-sense multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP) [1,2,6]. CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send a message. CD+AMP means that collisions are resolved through a bit-wise arbitration, based upon a preprogrammed priority of each message in the identifier field of a message. The higher priority identifier always wins bus access.

The first version of the CAN standards listed in Table Ⅰ, ISO 11519 (Low-Speed CAN) is for applications up to 125 kbps with a standard 11-bit identifier. The second version, ISO 11898 (1993), also with 11-bit identifiers provides for signaling rates from 125 kbps to 1 Mbps while the more recent ISO 11898 amendment (1995) introduces the extended 29-bit identifier. The ISO 11898 11-bit version is often referred to as Standard CAN Version 2.0A, while the ISO 11898 amendment is referred to as Extended CAN Version 2.0B. The Standard CAN 11-bit identifier field [6] in Figure 2 provides for 2^{11}, or 2048 different message

978-0-7695-3635-4/09 $25.00 © 2009 IEEE
DOI 10.1109/ICNDS.2009.142

identifiers, while the Extended CAN 29-bit identifier in Figure 3 provides for 2^{29}, or 537 million identifiers.

TABLE Ⅰ. CAN VERSIONS

NOMENCLATURE	STANDARD	MAX. SIGNALING RATE	IDENTIFIER
Low–Speed CAN	ISO 11519	125 kbps	11-bit
CAN 2.0A	ISO 11898:1993	1 Mbps	11-bit
CAN 2.0B	ISO 11898:1995	1 Mbps	29-bit

A. *Standard CAN*

Figure 2. Standard CAN: 11-Bit Identifier

The meaning of the bit fields of Figure 2 are:

• SOF—The single dominant start of frame (SOF) bit marks the start of a message, and is used to synchronize the nodes on a bus after being idle.

• Identifier—The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.

• RTR—The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way all data being used in a system is uniform.

• IDE—A dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.

• r0—Reserved bit (for possible use by future standard amendment).

• DLC—The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.

• Data—Up to 64 bits of application data may be transmitted.

• CRC—The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.

• ACK—Every node receiving an accurate message overwrites this recessive bit in the original message with a dominate bit, indicating an error-free message has been sent. Should a receiving node detect an error and leave this bit recessive, it discards the message and the sending node repeats the message after rearbitration. In this way each node acknowledges (ACK) the integrity of its data. ACK is 2 bits, one is the acknowledgement bit and the second is a delimiter.

• EOF—This end-of-frame (EOF) 7-bit field marks the end of a CAN frame (message) and disables bit–stuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is stuffed into the data.

• IFS—This 7-bit inter-frame space (IFS) contains the amount of time required by the controller to move a correctly received frame to its proper position in a message buffer area.

B. *Extended CAN*

Figure 3. Extended CAN: 29-Bit Identifier

As shown in Figure 3, the Extended CAN message is the same as the Standard message with the addition of:

• SRR—The substitute remote request (SRR) bit replaces the RTR bit in the standard message location as a placeholder in the extended format.

• IDE—A recessive bit in the identifier extension (IDE) indicates that there are more identifier bits to follow. The 18-bit extension follows IDE.

• r1—Following the RTR and r0 bits, an additional reserve bit has been included ahead of the DLC bit.

IV. A CAN MESSAGE

A. *Arbitration*

A fundamental CAN characteristic shown in Figure 4 is the opposite logic state between the bus, and the driver input and receiver output. Normally a logic high is associated with a one, and a logic low is associated with a zero—but not so on a CAN bus. This is why it is desirable to have the driver input and receiver output pins of a CAN transceiver passively pulled high internally, as in the SN65HVD230. In the absence of any input, the device automatically defaults to a recessive bus state on all input and output pins.

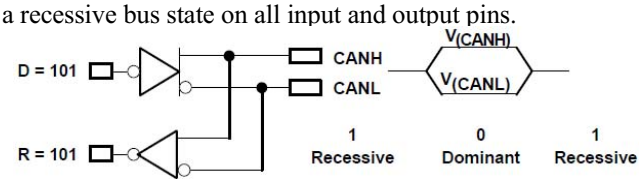

Figure 4. The Inverted Logic of a CAN Bus

Bus access is event-driven and takes place randomly. If two nodes try to occupy the bus simultaneously, access is implemented with a nondestructive, bit-wise arbitration. Nondestructive means that the node winning arbitration just continues on with the message, without the message being destroyed or corrupted by another node.

The allocation of priority to messages in the identifier is a feature of CAN that makes it particularly attractive for use within a real-time control environment. The lower the binary message identifier number, the higher its priority. An identifier consisting entirely of zeros is the highest priority message on a network since it holds the bus dominant the longest. Therefore, if two nodes begin to transmit simultaneously, the node that sends a zero (dominant) while the other nodes send a one (recessive) gets control of the CAN bus and goes on to complete its message. A dominant bit always overwrites a recessive bit on a CAN bus.

Note that a node constantly monitors its own transmission. This is the reason for the transceiver configuration of Figure 4 in which the CANH and CANL output pins of the driver are internally tied to the receiver's input. The propagation delay of a signal in the internal loop

from the driver input to the receiver output is typically used as a qualitative measure of a CAN transceiver. This propagation delay is referred to as the loop time (t_{LOOP} in a TI data sheet), but takes on varied nomenclature from vendor to vendor.

B. Message Types

There are four different message types [2,3], or frames (Figures 2 and 3) that can be transmitted on a CAN bus: the data frame, the remote frame, the error frame, and the overload frame. A message is considered to be error free when the last bit of the ending EOF field of a message is received in the error–free recessive state. A dominant bit in the EOF field causes the transmitter to repeat a transmission.

1) The Data Frame: The data frame is the most common message type, and is made up by the arbitration field, the data field, the CRC field, and the acknowledgement field. The arbitration field determines the priority of a message when two or more nodes are contending for the bus. The arbitration field contains an 11-bit identifier for CAN 2.0A in Figure 2 and the RTR bit, which is dominant for data frames. For CAN 2.0B in Figure 3 it contains the 29-bit identifier and the RTR bit. Next is the data field which contains zero to eight bytes of data, and the CRC field which contains the 16-bit checksum used for error detection. Lastly, there is the acknowledgement field. Any CAN controller that is able to correctly receive a message sends a dominant ACK bit that overwrites the transmitted recessive bit at the end of correct message transmission. The transmitter checks for the presence of the dominant ACK bit and retransmits the message if no acknowledge is detected.

2) The Remote Frame: The intended purpose of the remote frame is to solicit the transmission of data from another node. The remote frame is similar to the data frame, with two important differences. First, this type of message is explicitly marked as a remote frame by a recessive RTR bit in the arbitration field, and secondly, there is no data.

3) The Error Frame: The error frame [4] is a special message that violates the formatting rules of a CAN message. It is transmitted when a node detects an error in a message, and causes all other nodes in the network to send an error frame as well. The original transmitter then automatically retransmits the message. There is an elaborate system of error counters in the CAN controller which ensures that a node cannot tie up a bus by repeatedly transmitting error frames.

4) The Overload Frame: The overload frame is mentioned here for completeness. It is similar to the error frame with regard to the format, and it is transmitted by a node that becomes too busy. It is primarily used to provide for an extra delay between messages.

C. Error Checking and Fault Confinement

The robustness [4] of CAN may be attributed in part to its abundant error checking procedures. The CAN protocol incorporates five methods of error checking: three at the message level and two at the bit level. If a message fails with any one of these error detection methods, it is not accepted and an error frame is generated from the receiving nodes, causing the transmitting node to resend the message until it is received correctly. However, if a faulty node hangs up a bus by continuously repeating an error, its transmit capability is removed by its controller after an error limit is reached.

At the message level are the CRC and the ACK slots displayed in Figures 2 and 3. The 16-bit CRC contains the checksum of the preceding application data for error detection with a 15-bit checksum and 1-bit delimiter. The ACK field is two bits long and consists of the acknowledge bit and an acknowledge delimiter bit. Finally, at the message level there is a form check. This check looks for fields in the message which must always be recessive bits. If a dominant bit is detected, an error is generated. The bits checked are the SOF, EOF, ACK delimiter, and the CRC delimiter bits.

At the bit level each bit transmitted is monitored by the transmitter of the message. If a data bit (not arbitration bit) is written onto the bus and its opposite is read, an error is generated. The only exceptions to this are with the message identifier field which is used for arbitration, and the acknowledge slot which requires a recessive bit to be overwritten by a dominant bit. The final method of error detection is with the bit stuffing rule where after five consecutive bits of the same logic level, if the next bit is not a compliment, an error is generated. Stuffing ensures rising edges available for on-going synchronization of the network, and that a stream of recessive bits are not mistaken for an error frame, or the seven-bit interframe space that signifies the end of a message. Stuffed bits are removed by a receiving node's controller before the data is forwarded to the application.

With this logic, an active error frame consists of six dominant bits—violating the bit stuffing rule. This is interpreted as an error by all of the CAN nodes which then generate their own error frame. This means that an error frame can be from the original six bits to twelve bits long with all the replies. This error frame is then followed by a delimiter field of eight recessive bits and a bus idle period before the corrupted message is retransmitted. It is important to note that the retransmitted message still has to contend for arbitration on the bus.

V. ECAN

The eCAN [7] is a CAN controller with an internal 32–bit architecture.

The eCAN module consists of:

The CAN protocol kernel (CPK)

The message controller comprising:

- The memory management unit (MMU), including the CPU interface and the receive control unit (acceptance filtering), and the timer management unit
- Mailbox RAM enabling the storage of 32 messages
- Control and status registers

After the reception of a valid message by the CPK, the receive control unit of the message controller determines if the received message must be stored into one of the 32 message objects of the mailbox RAM. The receive control unit checks the state, the identifier, and the mask of all message objects to determine the appropriate mailbox location. The received message is stored into the first

mailbox passing the acceptance filtering. If the receive control unit could not find any mailbox to store the received message, the message is discarded.

A message is composed of an 11− or 29−bit identifier, a control field, and up to 8 bytes of data.

When a message must be transmitted, the message controller transfers the message into the transmit buffer of the CPK in order to start the message transmission at the next bus−idle state. When more than one message must be transmitted, the message with the highest priority that is ready to be transmitted is transferred into the CPK by the message controller. If two mailboxes have the same priority, then the mailbox with the higher number is transmitted first.

The timer management unit comprises a time−stamp counter and apposes a time stamp to all messages received or transmitted. It generates an interrupt when a message has not been received or transmitted during an allowed period of time (time−out). The time−stamping feature is available in eCAN mode only.

To initiate a data transfer, the transmission request bit has to be set in the corresponding control register. The entire transmission procedure and possible error handling are then performed without any CPU involvement. If a mailbox has been configured to receive messages, the CPU easily reads its data registers using CPU read instructions. The mailbox may be configured to interrupt the CPU after every successful message transmission or reception.

VI. CONCLUSIONS

CAN is a multi-master serial bus that allows an efficient transmission of data between different nodes. CAN is a flexible, reliable, robust and standardized protocol with real-time capabilities.

Fault confinement is also a major benefit of CAN. Faulty nodes are automatically dropped from the bus, which prevents any single node from bringing a network down, and assures that bandwidth is always available for critical message transmission. This error containment also allows nodes to be added to a bus while the system is in operation, otherwise known as hot-plugging.

The eCAN is a CAN controller with an internal 32−bit architecture. It uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time−stamping feature, the eCAN module provides a versatile and robust serial communication interface.

ACKNOWLEDGMENT

The authors would like to thank Feng Zhou, Dr. Yun Wu and Dr. Shigong Long for their contributions.

REFERENCES

[1] M. Farsi, K. Ratcliff, and M. Barbosa, “An overview of controller area network”, Computing & Control Engineering Journal, pp.113-120, 10(3), Jun.1999.

[2] K.M. Zuberi and K.G. Shin, “Design and implementation of efficient message scheduling for controller area network“, IEEE Transactions on Computers, pp.182-188, 49(2), Feb. 2000.

[3] P. Pedreiras and L. Almeida, “EDF message scheduling on controller area network”, Computing & Control Engineering Journal, pp.163-170, 13(4), Aug. 2002.

[4] B. Gaujal and N. Navet, “Fault confinement mechanisms on CAN: analysis and improvements”, IEEE Transactions on Vehicular Technology, pp.1103-1113, 54(3), May 2005.

[5] M. Bago, S. Marijan, and N. Peric, “Modeling Controller Area Network Communication”, The 5th IEEE International Conference on Industrial Informatics, Vol. 1,pp.485-490, 23-27 June 2007.

[6] Y. Yellambalase and C. Minsu, “Automatic Node Discovery in CAN (Controller Area Network) Controllers using Reserved Identifier Bits”, The IEEE Instrumentation and Measurement Technology Conference Proceedings, pp.1-3, 1-3 May 2007.

[7] G. Cena and A. Valenzano, “FastCAN: a high-performance enhanced CAN-like network”, IEEE Transactions on Industrial Electronics, pp.951-963, 47(4), Aug. 2000.