Research Journal of Applied Sciences, Engineering and Technology 5(2): 445-451, 2013
DOI:10.19026/rjaset.5.4971
ISSN: 2040-7459; E-ISSN: 2040-7467
© 2013 Maxwell Scientific Publication Corp.
Submitted: May 06, 2012 Accepted: June 15, 2012 Published: January 11, 2013

Research Article

Designing a Reliable and Application Specific Controller Area Network Protocol for Intra-Communication of an Embedded System

[1]Seyyed Amir Asghari, [2]Hassan Taheri and [1]Hossein Pedram
[1]Computer Engineering and Information Technology Department,
[2]Electrical Engineering Department, Amirkabir University of Technology, Tehran, Iran

Abstract: By utilizing a CAN (Controller Area Network) bus, point to point wiring is substituted by a shared medium. This is done by adding hardware (embedded systems) to each control unit that uses CAN protocol. The main purpose of this study is communication software protocol design and implementation based on CAN protocol (on CAN) for intra-communication in a micro satellite that can be used for intra-communication of other embedded systems. This protocol is designed in the application layer and is conformed to MAC and PHY layers of CAN protocol.

Keywords: CAN protocol, communication software, embedded system, micro satellite

INTRODUCTION

By utilizing CAN (Controller Area Network) bus, point to point wiring is substituted by a serial bus and the electronic equipment is connected together by a shared medium (Zuberi *et al*., 1997, 2000; Dorbin and Fohler, 2001). This is done by adding hardware (embedded systems) to each control unit which is conformed to CAN protocol. This hardware gives us the rules or protocols needed for sending to and receiving information from bus. In fact, for connecting each unit to the network, we need hardware similar to CAN network nodes. These nodes act as the interface between each electronic unit and bus network (Nolte *et al*., 2003). On these nodes, the entire stack of CAN protocol is implemented. A part of this stack, called Physical layer (PHY), is assumed as the hardware and the other parts, calling MAC and the application layer are assumed as the software and these parts are implemented on this platform. CAN protocol have 4 main and important advantages (Jaman and Hussain, 2007; Marino and Schmalzel, 2007; Bago *et al*., 2007; Murtaza and Khan, 2008):

- CAN protocol as a global standard can support communication into various embedded systems and generally hardware which has the capability of communication by CAN bus and protocol.
- Communication and information transfer workload in each network node of host processor is transferred to an intelligent peripheral communication device, therefore, CAN host controller processor have more time for accomplishing its system task.
- As a shared network, CAN decrease the needed amount of wiring and omits point to point wiring.

As a standard protocol, CAN possess a great demand in market. This creates a motivation for semi-conductors and chips manufacturers to produce goods of reasonable prices.

Access mechanism to bus in CAN protocol is CSMA/CD-A (Carrier Sense Multiple Access with Collision Detection) and the type of nodes in bus is master/slave. This protocol also supports multi-master capability (Ziermann *et al*., 2009; Chen and Tian, 2009).

In this study, a customized reliable and application specific Controller Area Network protocol design for intra-communication of an embedded system such a micro satellite is introduced (standard CAN protocol for these application is very comprehensive and It's usage has many overheads). Some of the requirements of this protocol are as follows:

- The protocol should have the capability of being implemented on all of satellite subsystems.
- The user should be able to send the desired number of bytes.
- The user's interface with protocol should be easy and reliable. A multi-thread replica should be designed for the satellite Command and Data Handling subsystem (C&DH) board. For all of subsystems there should be a single-thread replica.

Corresponding Author: Hassan Taheri, Electrical Engineering Department, AmirKabir University of Technology, Tehran, Iran

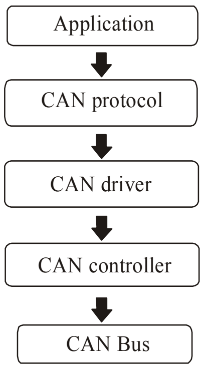


Fig. 1: System protocol total architecture

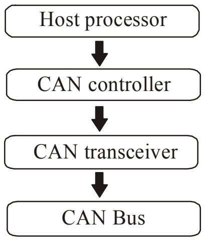


Fig. 2: CAN node structure

One of the other requirements is that the protocol should be able to be implemented on all micro satellite subsystems. The multi-thread replica should only be implemented on C&DH board. In this architecture, there exist two kinds of subsystems. One of them is related to C&DH which utilizes MPC555 micro-controller and uC/OS-II operating system. The other kind is related to other subsystems (the satellite consists of several subsystems) that exploits AT90CAN128 micro-controller. The difference between these two is the sub-layers of the protocol architecture.

The reason for this difference is that C&DH is multi-tasks and utilizes operating system, whereas other subsystems are single-task and are without operating systems. The total architecture of the system protocol is shown in Fig. 1.

CAN bus: CAN bus is the interface shared channel of satellite subsystems. CAN bus is designed to cause interaction of micro-controllers with each other without a central controller.

This feature is useful for some of the embedded systems in which there exists no central processor for controlling the interactions.

For this reason, this bus is useful in embedded systems (like car industries) (Chen and Tian, 2009).

Communication technology: CAN bus is a broadcasting bus. Therefore, all of the nodes connected to the bus, receive the sent messages. Each message contains a header (Zuberi *et al*., 2000; Dorbin and Fohler., 2001).

If each CAN node receives all of the sent messages on bus, there should be a mechanism to determine which packet is accepted and which one is rejected. Each node consists of three components (Fig. 2) (Zeng *et al*., 2009; Aysan *et al*., 2010a; Zhu *et al*., 2010).

Main processor: It determines what message to be sent, which message to be accepted and which one to be rejected. The processor performs this action through bit filters. This filter is placed on CAN packet identification field.

CAN controller: It saves the received bits till all of the messages are received and then it sends a signal to the main processor to inform about packet receiving. On the other hand, if it receives a message from the main processor, it sends this message as a serial through CAN transceiver on bus.

CAN transceiver: At the time of receiving, CAN transceiver converts bus voltages to interpretable bits for controller and at the time of sending, it converts bits to the corresponding voltage level and sends them on the bus. CAN bus can operate with several bit rates. For embedded media that the coverage is not more than 40 m, CAN bus can operate with 1 Mbps. However, if the coverage is more, lower bit rates (e.g., 500, 250 and 125 kbps) should be utilized (Zuberi *et al*., 2000; Dorbin and Fohler, 2001).

Frames: CAN bus can be configured to work with two kinds of different frames; standard frame and extended frame (Aysan *et al*., 2010a, b; Zhu *et al*., 2010; Umamaheswaran and Nagendra, 2009). The only difference is that the standard frame utilizes 11 bit frame identification; however, the extended frame utilizes 29 bits identification. The structure of the standard and the extended frames are shown in Table 1 and 2.

The identification field consists of some controller and application information. As it was mentioned before, mask filter is adjusted on this field.

The RTR (Remote Transmission Request) field is set to 1 if it is determined to send a remote frame. DLC field specifies the entire data byte. CRC field is utilized to check frame sending failure. ACK field determines if the frame is correctly received.

Table 1: Standard frame format

ID	RTR	DLC	Data	CRC	ACK
11 bit	1 bit	4 bit	(8*n) bit	15 bit	1 bit

Table 2: Extended frame format

ID	RTR	DLC	Data	CRC	ACK
29 bit	1 bit	4 bit	(8*n) bit	15 bit	1 bit

Table 3: The extended frame identification format

Priority	Src	Dst	Last seg	NU	Type	Seq no	NU
2	4	4	1	3	8	6	1

THE PROPOSED CAN PROTOCOL FOR A SAMPLE MICRO SATELLITE

In this section, the responding attitude towards the mentioned requirements in the previous section and also the protocol operation in different situations are analyzed.

Message format: In this protocol, the extended frame is utilized. The identification field format of this frame is shown in Table 3.

Priority: This field specifies the priority of CAN frame. 00 and 11 have the most and least frame priority, respectively.

Srs: This field consists of source subsystem address.

Dst: This field consists of destination subsystem address.

Last seg: If this field is 1, it means that the received frame is the last segment of segment sets of a transaction.

NU: This field is not utilized and it can be used in later development or allocating more bits to source and destination addresses.

Type: This single-bit field specifies the kind of frame. Four most significant bits (ACK, NAK, REQ and DATA) specify the frame type and four least significant bits specify the frame sub-types.

In order to decrease the length of Type field, the kind of the frame is determined with regarding to the destination subsystem. For example, if this field is 0×11 and the address is Power subsystem frame, it indicates SET_SWITCH command. However, if the frame's destination address is Attitude Determination and Control Subsystem (ADCS), it indicates SET_ACTIVE_DATA command.

Seg no: Each CAN frame is able to transfer up to 8 bytes, therefore, in case the transmitted packet data length is more than 8 bytes, segmentation operation should be performed. This field shows the segmentation number related to each transaction.

The purpose of this field is assuring to receive all of the segmentations. If each segmentation is failed due to any reason, the receiver is able to specify the occurred failure by the next segmentation number and informs the sender about the occurred failure by sending NAK frame in the transaction to have a second sending. Figure 3, shows this mechanism.

Communication with subsystems: Figure 4 shows satellite communication sub-structure. The entire structure of the satellite is master/slave; i.e., C&DH firstly sends a command to each subsystem and the subsystem after performing the command sends back the suitable response to C&DH.

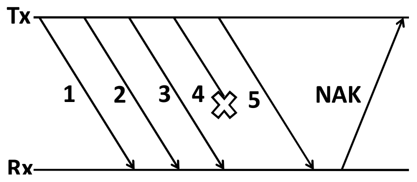


Fig. 3: Segment sequence number application

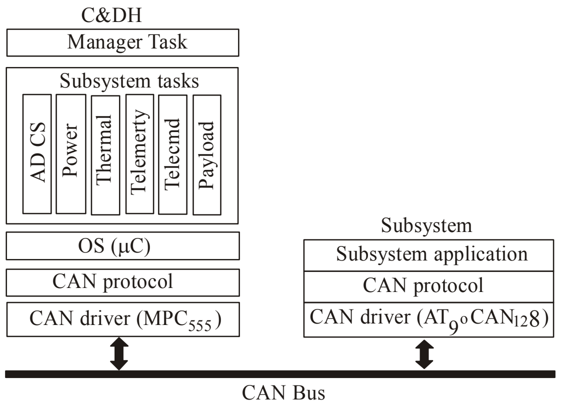
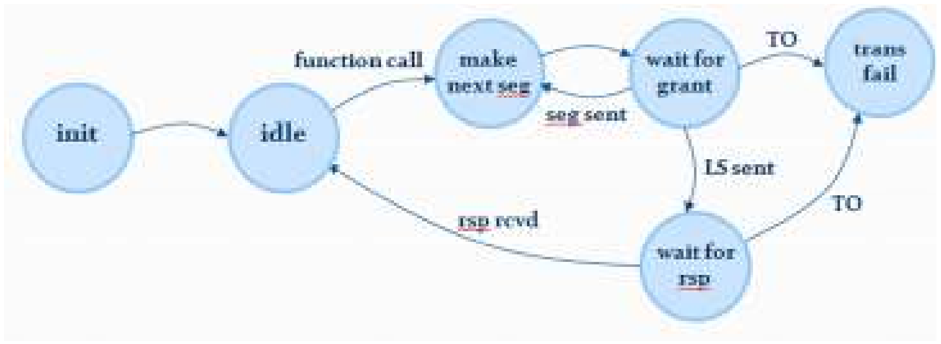


Fig. 4: Satellite communication infrastructure
ADCS: Attitude determination and control subsystem; Telecmd: Telemetry and telecomand; C&DH: command and data handling



seg:	Segment
trans:	Transaction
rsp:	Response
LS:	Last Segment
TO:	Time Out
rcvd:	Received

Fig. 5: State machine of type 1 transaction in C&DH

In this protocol, three kinds of transactions are considered and each of them is described in the following sections.

Sending data to subsystem (type 1): In these transactions, C&DH sends a command as well as the required data for performing the command to subsystem.

For example, SET_SWITCH command, that is connected to power subsystem is one f them.

The state machine of these transactions is shown in Fig. 5. After initial performing, the state machine is firstly placed in idle state. Afterwards, by calling the related function, the first segment which consists of command and data is constructed and waits for the grant to use bus and frame sending.

In case the sending grant is not received before the previous allocated time, timeout failure occurs and transaction will be finished. After sending the next segments (i.e., data), like what was mentioned, the state machine is transferred to wait state for response and will wait to receive response from subsystem. If the response is not received in the specified time, timeout failure happens. Otherwise, the transaction will be finished by receiving the response and the state machine turns back to the idle state.

Data receiving from subsystem (type 2): In this transaction, C&DH demands receiving data by sending a command subsystem. One example of this transaction starts by GET_SENSOR_DATA command.

By function call, the transaction is started and the state machine is transferred to wait state for grant and waits to be allowed to use CAN bus. In case, the grant is not sent in the specified time, the transaction encounters timeout failure.

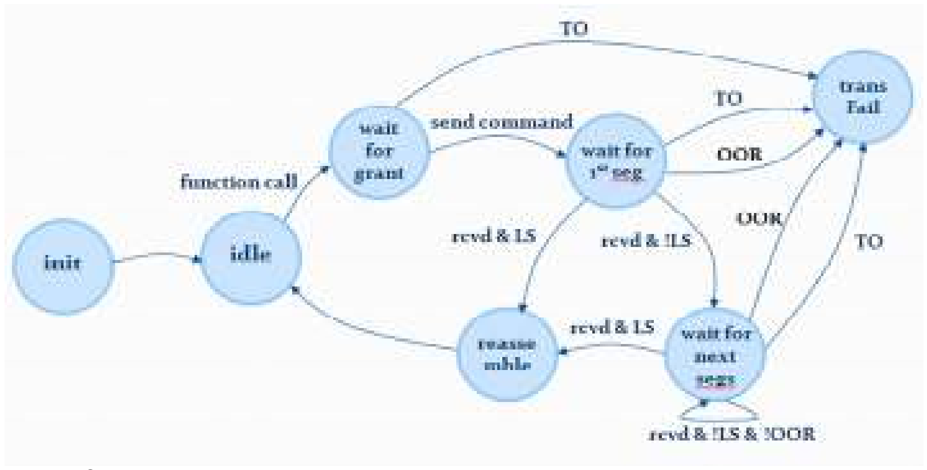
After sending the command, C&DH waits for receiving the first segment. After receiving the first segment, if the LS field is 1, it means that this segment is the last segment of the current transaction and the packet is produced and turned back as the output function. Otherwise, the related task in C&DH waits for receiving the next segments. After receiving each segment, the number of the segment is checked and if it is not correspondent with the desired amount, OOR (Out of Order) failure is produced. Finally after receiving the last segment, the machine turns back to the idle state. This transaction is shown in Fig. 6.

Sending single-command to subsystem (type 3): The purpose of this transaction is sending a command by C&DH to subsystem in a way that there is no need to have data. For example, START_SEND command in Payload subsystem is such a transaction. This transaction is similar to type 1 transaction, by this difference that it is without data for sending. This transaction type is shown in Fig. 7.

EXPERIMENTAL RESULTS

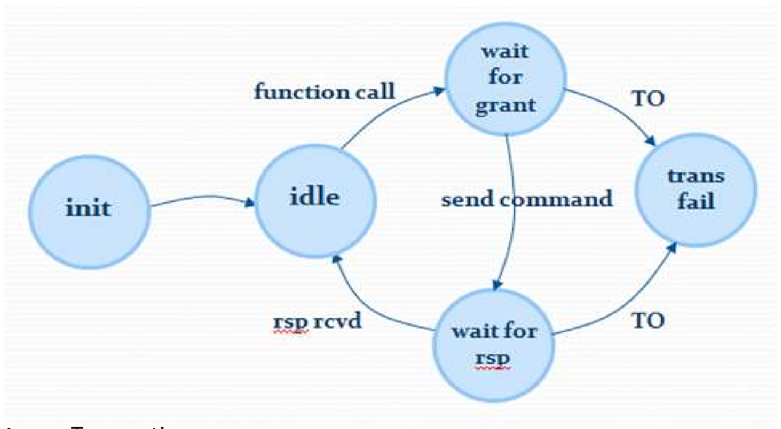
For testing this protocol, the information packet should be transferred through communication bus. In order to do this, firstly, it is needed to test CAN bus driver.

Testing CAN drive: The utilized microprocessors for communication are MPC555 microprocessor (C&DH processor) and AT90CAN128 microprocessor (Subsystem processor). The suitable driver is



seg:	Segment
trans:	Transaction
LS:	Last Segment
TO:	Time Out
rcvd:	Received
OOR:	Out of Order

Fig. 6: State machine of type 2 transaction in C&DH



trans:	Transaction
TO:	Time Out
rcvd:	Received
rsp:	Response

Fig. 7: State machine of type 2 transaction in C&DH

customized for them. In order to do this, PCAN-USB tool is utilized. This tool is an adapter that prepares the ability of connecting the computer to CAN bus by USB. Beside this tool, there exists an interface application software, that CAN bus sending rate can be adjusted by it. Moreover, CAN frames can be produced periodically and with desired field amounts and can be sent onto bus. On the other hand, all of the transferred frames onto bus can be monitored and their contents can be observed by this tool.

The testing procedure is that CAN bus sending rate is adjusted equally between the tested node (that in one state is C&DH node and the other state is a node related to one of the subsystems) and PCAN tool. Then in one step, the tested node is considered to be the sender and PCAN node is the receiver. In the other step, PCAN node is considered to be the sender and the tested node is the receiver. In both steps, the correctness of sending and receiving CAN frame is tested. Therefore, it can be assured about the accuracy of CAN driver's entire operation.

Table 4: Insertion the present protocol in some communication function

Function name	Response normal time (ms)	Send data capacity (byte)	Receive data capacity (byte)	Timeout time (ms)
thermal-get-data	20	4	80	40
adcs-get-data	20	4	60	40
adcs-send-data	5	20	4	10
pw-get-data	4	4	32	8
pw-set-switch	5	8	4	10

Proposed CAN protocol testing: After testing CAN driver's operation, CAN protocol should be tested. For testing the protocol, two subsystem testing nodes are utilized, i.e., C&DH and PCAN tool nodes to monitor traffic on bus. These nodes are connected to CAN bus by DB9 connector and based on CiA standard. Then each mentioned transaction should be tested accurately and under different situations (like creating timeout or out of order situations) to be assured about their operations.

Table 4 shows an example of the results obtained from these operations. As it can be seen, the suggested protocol can fulfill deadlines for different information packets.

CONCLUSION

CAN bus that is formed based on CAN communication protocol, is a useful communication bus especially for embedded systems and it covers great area of application. In specific application, CAN protocol headers can be omitted by a flexible design presentation. Based on this attitude, in this paper, a customized protocol based on CAN protocol is designed and implemented for intra-communication of satellites and it is very light and has a high reliability.

ACKNOWLEDGMENT

This study was supported by the Research Institute for ICT-ITRC.

REFERENCES

Aysan, H., A. Thekkilakattil, R. Dobrin and S. Punnekkat, 2010a. Efficient fault tolerant scheduling on Controller Area Network (CAN). Proceeding of International 15th International conference on Emerging Technologies and Factory Automation, pp: 1-8.

Aysan, H., A. Thekkilakattil, R. Dobrin and S. Punnekkat, 2010b. Efficient fault tolerant scheduling on Controller Area Network (CAN). Proceeding of International 13th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, pp: 1-8.

Bago, M., S. Marijan and N. Perić, 2007. Modeling controller area network communication. Proceeding of 5th International Conference on Industrial Informatics, pp: 485-490.

Chen, H. and J. Tian, 2009. Research on the controller area network. Proceeding of International Conference on Networking and Digital Society (ICNDS 09), pp: 251-254.

Dorbin, R. and G. Fohler, 2001. Implementing off-line message scheduling on Controller Area Network (CAN). Proceeding of the 8th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2001), pp: 241-245.

Jaman, G. and S. Hussain, 2007. Structural monitoring using wireless sensors and controller area network. Proceeding of 5th Annual Conference on Communication Networks and Services Research (CNSR' 07), pp: 26-34.

Marino, A. and J. Schmalzel, 2007. Controller area network for in-vehicle law enforcement applications. Proceeding of Sensors Applications Symposium (SAS 07), pp: 1-5.

Murtaza, A.F. and Z.A. Khan, 2008. Starvation free controller area network using master node. Proceeding of 2th International Conference on Electrical Engineering, pp: 1-6.

Nolte, T., H. Hansson and C. Norstrom, 2003. Probabilistic worst-case response-time analysis for the controller area network. Proceeding of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS' 03), pp: 200-207.

Umamaheswaran, S. and S. Nagendra, 2009. Microcontroller based multi-star simulator using Controller Area Network (CAN). Proceeding of Aerospace and Electronics Conference (NAECON), pp: 139-145.

Zeng, H., M.D. Natale1, P. Giusto and A. Sangiovanni-Vincentelli, 2009. Statistical analysis of controller area network message response times. Proceeding of International Symposium on Industrial Embedded Systems (SIES 09), pp: 1-10.

Zhu, Y., Y. Wang and U. Schaefer, 2010. Study on the communication between FPGA and observer using controller area network and UART. Proceeding of International Conference on Information, Networking and Automation (ICINA), pp: 240-244.

Ziermann, T., S. Wildermann and J. Teich, 2009. CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16x higher data rates. Proceeding of Conference on Design and Automation and Test in Europe (DATE'09), pp: 1088-1093.

Zuberi, K.M. and K.G. Shin, 1997. Scheduling message on controller area network for real-time CIM application. IEEE Trans. Robot. Autom., 13(2): 310-316.

Zuberi, K.M. and K.G. Shin, 2000. Design and implementation of efficient message scheduling for control area network. IEEE Trans. Comput., 49(2): 182-188.