Design of CAN-based System for Distributed Control

Jin-Woo Park, Dong-Gyu Noh, Jang-Myung Lee
Electronics Engineering, Pusan National University
Jang-Jung Dong, Kum-Jung Ku, Pusan, 609-735, Korea
Tel: +82-51-510-1696, Fax: +82-51-515-5190
E-mail: jwpark@mail.pusan.ac.kr

Abstract: In this paper, we propose the design method of distributed control system using Controller Area Network (CAN). CAN is an advanced serial communication protocol for distributed real-time control systems. It is a contention-based multi-master network whose timeliness properties come from its collision resolution algorithm, which gives a high schedulable utilization and guaranteed bus access latency. With proposed method using CAN, we apply to robot controller. The effectiveness of proposed method is demonstrated by the simulation and experiment.

1. Introduction

In motion control of robot or motors, one does not usually consider communication constraints. It is often presumed that a host computer can send torque, position and position commands to motors within a short control period (e.g. several milliseconds). Assuming this condition, many effective feedback controllers have been proposed. However, there are situations where communication between the host and the robot or motors is limited [1].

Generally, interface protocol for Communication between the host and other controllers, still widely used, is asynchronous serial communication (RS232C). However, the fastest available communication speed using RS232C is 115.2Kbps and when motors are configured using a daisy-chain communication bus, the communication speed decreases as the number of motors increase. Therefore, it can't be applied for real-time control.

In order to solve above communication constraints, Control system have distributed structure and reduces data transmitted through the channel and stored in the module. Instead if doing motion control exclusively in a central processor, It is delegated some parts of motion control to distributed motor modules. The modules include a micro-controller for position, velocity and torque control. In this control structure, control states of a motor such as position, velocity and torque, are not controlled by the central processor but by the module itself [2].

One of method for implementation, there is Controller Area Network (CAN) [3][4]. CAN is suitable for real-time control system. CAN is a serial bus system especially suited for networking "intelligent" devices as well as sensors and actuators within a system or sub-system. All CAN nodes are able to transmit data and several CAN nodes can request the bus simultaneously. The maximum transmission rate is specified as 1M bit/s. One of the outstanding features of the CAN protocol is its high transmission reliability. The CAN controller registers a stations error and evaluates it statistically in order to take appropriate measure. Therefore it efficiently can be transmitted and received data for real-time data exchange.

In this paper, we propose the design method of CAN-based system for distributed control and apply CAN to distributed control for motors. This paper is organized as follows. In the section 2, we analyze CAN. Section 3 deals with how to design of distributed system by using CAN. In the section 4, Proposed method is applied to Robot controller. Experiment and simulation results present the advantage of proposed method. Section 5 presents conclusions drawn from this work.

2. Controller Area Network (CAN)

CAN is a broad bus with a multi-master architecture. The transmission medium is usually a twisted pair cable. The network maximum length depends of data rate. Typical bounds are: 40m @ 1Mbps, 1000m @ 50kbps. Bit transmission takes possible representations: recessive, which only appears on the bus when all the nodes send recessive bits; dominant, which needs only to be sent by one node to appear on the bus. This means that a dominant bit sent by one node can overwrite recessive bits sent by other nodes. This feature is exploited for bus arbitration. A given bit-stream is transmitted using the NRZ code. Data transfers are subject to a bit stuffing technique that prevents more than five consecutive bits of identical polarity to be transmitted, through automatic insertion of a complementary bit.

Accordingly with CAN terminology, data to be transferred is encapsulated within "communication objects": a unique identifier is assigned to each communication object. The Controller Area Network is a carrier sense multi-access with collision resolve network: nodes delay transmission if the bus-line is in use; when a bus idle condition is detected, any node may start transmitting; bus access conflicts are resolved through comparison of communication objects identifiers and work as follows [5]:

- While transmitting a communication object identifier, each node monitors the serial bus-line;
- If the transmitted bit is recessive and a dominant bit is monitored, the node gives up from transmitting and starts to receive incoming data;
- The node transmitting the object with the highest identifier will go through gets the bus

That means: arbitration is non-destructive, since transmission of the highest identifier object undergoes without delay; bus access is prioritized, allowing transmission of

more urgent data to takeover less urgent one. Automatic retransmission of a communication object is provided after a loss in an arbitration process.

CAN defines four types of message packets: data frame, remote frame, error frame, and overload frame. A data frame is used to send some data to other devices, and it can have up to 8 bytes of data. A remote frame has the same structure of the data frame, except the data field. The remote frame has no data, and it is used to request data from a remote device. An error frame is used to notify all devices on the network when an error is detected, while an overload frame is transmitted when a device is not ready to receive another frame.

The integrity of data and remote frames is checked by a cyclic redundancy code (CRC) 15 bit sequence particularly well suited to check the integrity of frames with a total bit length less that 127, thus providing a good error detection coverage [6].

In summary it can be said that CAN implements a traffic-dependent bus allocation system that permits, by means of a high useful data rate at the lowest possible bus data in terms of the bus busy rate for all station. The efficiency of the bus arbitration procedure is increased by the fact that the bus is utilized only those stations with pending transmission requests.

3. Design of Distributed System

In a distributed control system a number of nodes are connected through a shared communication network. Therefore, resource sharing or scheduling is a topic of major importance. If several time-critical control activities are multiplexed onto a computer system it is not sufficient to provide a fast computer system. Scheduling policies determine which entity (e.g. process or message) that at a given time instant can use a given resource.

Designing a distributed system around CAN requires the identification of the appropriate level of system decentralization and a suitable system organization. When it designs distributed system using CAN, It has to be considered as follows:

- Establishing a global clock for synchronization – CAN doesn't specifies a global clock but, provide a base for implementing clock synchronization [7].
- Several scheduling policies can principally be used at the CAN – However, it has to notice when it decide scheduling policy because CAN has the fixed error handling policy [8][9].
- Allotting Identifier – It is important to base these decisions on application (message) requirements with respect to consistency and real-time behavior. Identifier of the most important data has the lowest value than other data.

The procedure of design for system based on CAN is illustrated in Fig 1. First, we set numbers of node that are applied in the system and then define sample time and allot ID [Identifier] by priority. In case of allotting ID at CAN, the ID of data of importance or urgency has higher priority than other data. That is, high priority data has low ID than lower priority data. Next, it checks whether data can be processed within designed sampling time. If it satisfies its demand then next step is processed, if not, redesign is needed. Last, it applies for prototype system and it compares result of simulation or experiment. If it isn't same then redesign is needed.

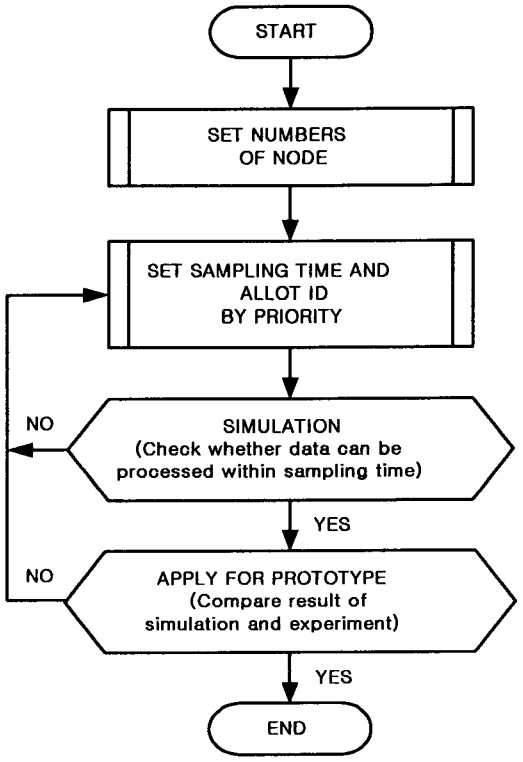


Fig. 1. Flow chart for system design based on CAN

4. Application : Robot Controller

We redesign robot controller of SCORBOT-ER VII in Fig. 2 for application example of distributed system. Originally, SCORBOT-ER VII that is made by Eshed Robotics in 1991 communicated with host computer through serial. It is impossible for implementing our project that is the development of tele-operation surgery system because serial communication between two controllers is too late for real-time data exchange. So, we apply CAN system to robot controller for real-time communication and redesign hardware instead of old robot controller.

4.1 Simulation

For CAN-Communication transmission time is always constant for a message (error free transmission assumed), but for the waiting time until transmission no fixed value can be given. It depends highly on the bus load and the priority of the message. Worst case waiting times can only be given for high-priority message. For lower priority message mean values and statistical techniques have to be used to assure feasibility of the design under the given network load. Therefore, It is considered probability of collision of message and sample time under design step. When above consideration is examined, the CAN simulation is convenient for examination.

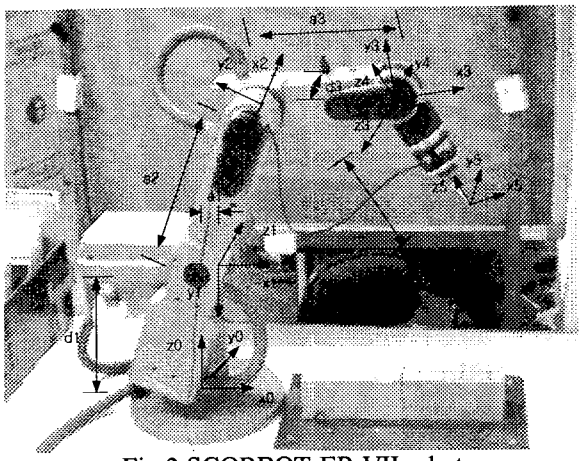
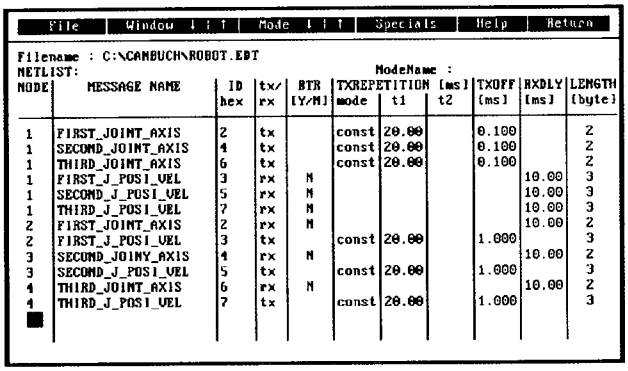


Fig.2 SCORBOT-ER VII robot

There are methods and tools for simulation [10][11]. In this paper, we used CAN simulator of I+ME ACTIA. We simply simulated whether messages transmit correctly within wanted time at 3 axes, and message transmission with noise. First, Input of transmission simulation is shown in Fig. 3 in a syntactical form accepted by CAN simulator.



File | Window | Mode | Specials | Help | Return

Filename : C:\CANBUCH\ROBOT.EBT
NETLIST: NodeName :

NODE	MESSAGE NAME	ID hex	tx/ rx	RTR [Y/N]	TXREPETITION [ms] mode	t1	t2	TXOFF [ms]	RXDLY [ms]	LENGTH [byte]
1	FIRST_JOINT_AXIS	2	tx		const	20.00		0.100		2
1	SECOND_JOINT_AXIS	4	tx		const	20.00		0.100		2
1	THIRD_JOINT_AXIS	6	tx		const	20.00		0.100		2
1	FIRST_J_POSI_VEL	3	rx	N					10.00	3
1	SECOND_J_POSI_VEL	5	rx	N					10.00	3
1	THIRD_J_POSI_VEL	7	rx	N					10.00	3
2	FIRST_JOINT_AXIS	2	rx	N					10.00	2
2	FIRST_J_POSI_VEL	3	tx		const	20.00		1.000		3
3	SECOND_JOINT_AXIS	4	rx	N					10.00	2
3	SECOND_J_POSI_VEL	5	tx		const	20.00		1.000		3
4	THIRD_JOINT_AXIS	6	rx	N					10.00	2
4	THIRD_J_POSI_VEL	7	tx		const	20.00		1.000		3

Fig. 3. Simulator definition of message

At Fig. 3, the simulation network consists of six transmitter and six receiver definitions. The message with symbolic name FIRST_JOINT_AXIS is assigned to network node 1. The assigned CAN identifier is 2 and the message is defined as transmitter(tx).

The field TXREPETITION defines that the message FIRST_JOINT_AXIS will be transmitted once every 20ms. In that case 0.1ms time delay will be required before transmitting for the first time. With each transmission 2byte data information will be transferred. The corresponding receiver(rx) with an identical symbolic name and identifier is defined in network node

Fig. 4 shows simulation result. The right section of figure presents the average bus load , the number of delayed messages. The central part of the screen is filled with a delay profile of all transferred messages. We obtain, for example, the information that 60% of transferred message have been delayed less than 149ms. In the simulation, Number of lost message (LostMess) is none and the maximum delay time (MaxDlyT) is 228us.

Second, we did simulation in case that it exist noise in the bus line. We assumed that noise is occurred every 5ms and also examined bus load and maximum delay time by changing noise length. The number of actual transferred message is 1000. The result is showed in Table 1.

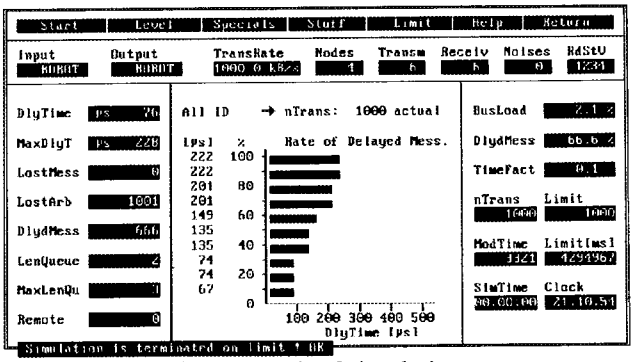


Fig. 8. Result of simulation

Table 1. Transmission under existing noisy environment

Noise repetition [ms]	Noise length [ms]	Bus load [%]	Maximum delay time [us]	Lost Message
5	0.01	4.7	326	0
	0.05	5.2	430	0
	0.1	5.9	480	0
	1	17.5	2,400	0
	2	28.8	4,400	0
	3	39.1	7,600	4

Even in a noisy environment, In the case of CAN, most messages are transmitted within wanted time. But if too much noise (3ms) existed in bus line, lost messages are appeared (LostMess:4). Therefore, it is needed hardware design technique for no occurring a noise.

4.2 Experimental Setup

The Specification of re-designed robot controller is as like Table 2.

Table 2. Robot controller specification

	SPECIFICATION
Number of Motor	5
Sub-controller	87C196CA [Intel]
Transmission data rate	1M bit/s
Control algorithm	Independent Joint Control method
Sampling time	1m sec

Based on proposed design method, robot controller has developed. It consists of 5 sub-controller that is based on micro-controller (80196CA) as like Table 2 [11], and host controller (Windows NT). Host controller calculates trajectory generation of robot and sends calculated value to sub controller. Sub controller receives commands of host computer through CAN and controls each axis of robot and sends position, velocity information to host computer. We control robot using independent joint control method [12]. Sampling time of host computer and sub controller are 20[mSec] and 1[mSec] respectively. We used CAN for Communication between

two controllers. Transmission data rate is 1M bit/s. One of modules of robot controller is as like Fig. 5. Fig. 6 is overall block diagram of distributed control system of robot.

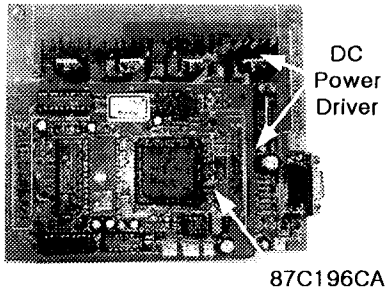


Fig 5. A module of Robot controller

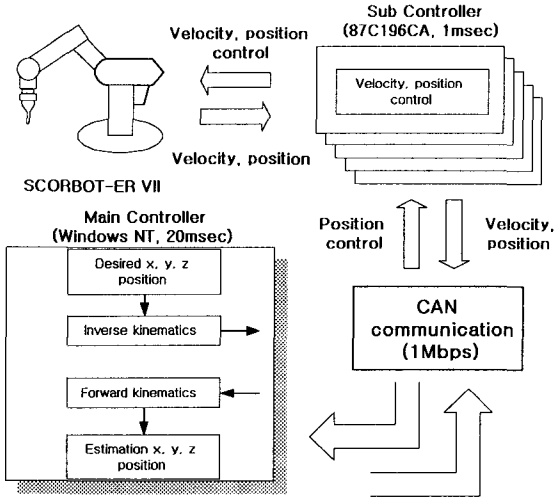
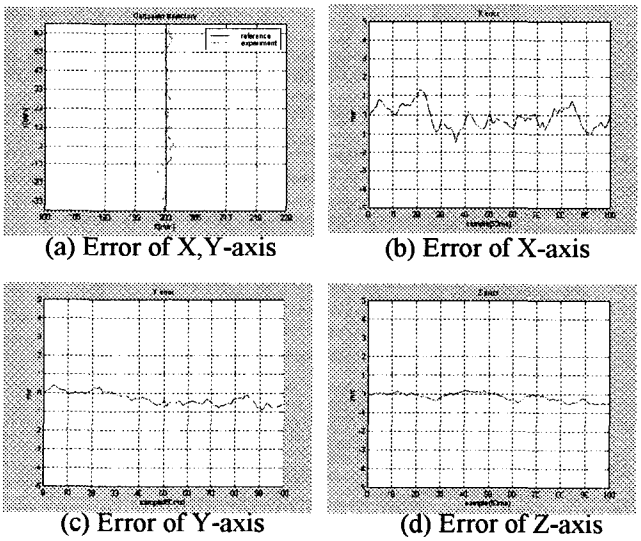


Figure 6. Overall block diagram of distributed control system of robot.

4.3 Results

We executed the experiments that end-effector of robot move toward Y-axis by constant velocity of 1 cm/sec during 5 sec. Initial and final position of Robot are [X : 200, Y : 65, Z : 100 mm], [X : 200, Y : -35, Z : 100 mm] respectively. The results are showed in Figure 7.



(a) Error of X,Y-axis (b) Error of X-axis

(c) Error of Y-axis (d) Error of Z-axis

Fig 3. Result of simulation

Figure 7 shows that the results of experiment converge given trajectory within the error of 2mm.

5. Conclusions

In this paper, we proposed the method of design of CAN based system for distributed control. The own properties of CAN, open structure, flexibility, complete communication protocol offers an optimal platform than other sensor level protocols. To show the efficiency of proposed method, we applied it to robot controller and real experiment and simulation are preformed. The result show that CAN is a good way to perform distributed processing, essential characteristics in distributed system.

We are in the process of adding CAN network to haptic interface system for remote control. After making the system, we will analyze time delay of two distributed system at the public line and propose the method of minimizing time-delay for real-time communication.

References

[1] K.H. Kim, N.J. Ferroer, "A Distributed Control Scheme for Motor Networks with Communication-Constrained Channels," Proc. Of IEEE Int. Conf. on Robotics & Automation, pp. 207-212, May, 1999.

[2] S.Y. Lee, J.W. Lee, D.S. Choi, M.S. Kim, C.W. Lee, "The Distributed Controller Architecture for a Master arm and its Application to Teleoperation with Force Feedback," Proc. Of IEEE Int. Conf. on Robotics & Automation, pp. 207-212, May, 1999.

[3] ISO DIS 11898 – Road vehicles – Interchange of digital information – Controller Area Network (CAN) for high-speed communication, 1992.

[4] CAN Specification version 2.0. Robert Bosch GmbH, 1991.

[5] J.M. Lee, S. Lee, M.H. Lee, K.S. Yoon, "Integrated Wiring System for Construction Equipment," IEEE/ASME Trans. on Mechatronics, vol. 4, no. 2, June, 1999.

[6] Jose Rufino, Paulo Verissimo, "A Study on the Inaccessibility Characteristics of the Controller Area Network," Proc. International CAN Conference 95, pp.7.12-7.22, 1995.

[7] R. Tuominen, T. Virvalo, "Synchronization of Servo system Using CAN," Proc. International CAN Conference 95, pp.9.12-9.20, 1995.

[8] K. M. Zuberi, K.G. Shin, "Scheduling Message on Controller Area Network for Real-Time CIM Applications," IEEE Trans. on Robotics and Automation, vol. 13, no. 2, April 1997.

[9] M.A. Livani, W.J. Jia, "Scheduling Hard and Soft Real-Time Communication in the Controller Area Network (CAN)," 23rd IFAC/IFIP Workshop on Real Time Programming, China, June 1998.

[10] K.W. Tindell, A.Burrns, "Calculationg Controller Are Network (CAN) Message Response Times," Proc. 1994 IFAC workshop on Distributed Computer Control Systems (DCCS), Spain, Sept. 1994.

[11] http://www.ime-actia.com/

[12] Intel, 82527 – Serial Communication CAN Protocol Controller, December 1993.

[13] M.H. Choi, "A Formulation of Joint Disturbance Torque and Its Application for Independent Joint Controlled Robotic Manipulators," IEEE Int. Conf. on Systems, Man, and Cybernetics, Oct. 1998.