Implementation of CAN BUS in an Autonomousall-Terrain Vehicle

K Ganga Amareswarnadh
Computer Science Engineering
Saveetha School Of Engineering, India

Abstract— The main purpose of this effort is to design an autonomous all terrain vehicle which uses a CAN bus. The paper describes the operation and advantages of the CAN protocol in automobiles. The ATV used for this design is a Honda Four Trax Rancher AT and a brief explanation of the parts in the ATV is provided. The Renesas RX62N microcontroller is used as a CAN controller which creates a single two wire bus through which electronic control units (ECU) in the automobiles communicate. The working of the CAN protocol on the RX62N has been explained with the help of pseudo code.

I. Introduction

The recent technology trends in the automobile industry are bringing more comfort in a vehicle by incorporating automation techniques like collision avoidance (which uses lasers to detect the objects around the vehicle and when the vehicle gets closer to any object, the brakes will be applied automatically), advanced safety features, entertainment devices and lot more. As the technology is developing, the use of electronic control units (ECU) in vehicles is increasing rapidly, making the communication between them very complex. Multiplexed communication was eventually developed to decrease the interconnections (cables) and the complexity between the ECUs. But the multiplexed communication has not met the real time communication requirements. In 1980 s, BOSCH, a technology based corporation designed a multi master serial communication protocol called Controller Area Network (CAN) which is robust, real time and also reduces the amount of cables to be used for the interconnections.

The CAN protocol is an asynchronous serial communication protocol which follows ISO 11898 standards and is widely accepted in automobiles due to its real time performance, reliability and compatibility with wide range of devices. The CAN protocol is a two wire, half duplex system which has data rates up to 1Mbps and offers a very high level of security. Its ease of use, robust, low cost and versatile technology made it applicable in other areas of applications where inter processor communication or elimination of excessive wiring is needed. Some of the areas it is widely used are industrial machinery, avionics, medical equipments, building automation etc.

II. Hardware

A. Zapata bot

The ATV selected for this project was a Honda Four Trax Rancher AT, which is also known as the TRX420FPE/EPS. It has a 420cc gasoline engine, an electric power steering assist system, and an electric transmission shift system. The power assist steering system is comprised of a torque sensor and an electric motor attached to the steering column. Additionally, the power assist system is driven by a complex computer and power electronics module. The cost of this ATV was approximately $8000. Since the ATV is a vehicle, it has an alternator that charges a battery to approximately 14 volts. This means that all of our circuits must be able to operate above 14 volts in case there are variations from the intended charging voltage. Any device that is above or below this should have some sort of DC-DC converter. Additionally, anywhere that Vcc from the ATV is considered in this document, it will be considered at 14 volts. [1]

B. Renesas RX62N microcontroller

The Renesas RX62N group has a RX family/RX600 series 32-bit CPU which features high performance and high speed. The RX62N group is equipped with two channels of USB 2.0 , one channel of ethernet, one channel of CAN bus protocol, timers, independent watchdog timer and brown out detectors (power on reset and low level voltage detection). The Fig 1 shows the RX62N board used in this paper.

Fig. 1. Renesas RX62N mircontroller

The CAN module on RX62N group implements one channel of CAN bus protocol according to the specification of ISO 11898-1. This allows communication of messages in both, standard identifier (11 bits) and extended identifier (29 bits) and allows data rates up to 1 Mbps.[2]

III. Operation Of Can Bus

The CAN controller implements only three layers of the ISO/OSI Reference model. It creates a bridge from data link layer to application layer (as shown in Fig. 2) in order to limit the resources and improve the performance. The other layers i.e. Layers 3 to Layers 6 are implemented in higher layer protocols like CANopen, J1939 and DeviceNet. The physical layer and data link layer are integrated on the CAN controller chips and the library functions from the manufactures define the connection between the layers. The application layer deals with the design of the CAN bus for different type of applications.

Fig. 2. ISO/OSI Reference model in CAN protocol [9]

The CAN protocol is a multi master bus access protocol which interfaces unlimited number of nodes in theory and depends on the chip practically. Normal chips available in the market allow upto 110 nodes and extra nodes can be interfaced using repeaters. A maximum of 8 bytes of data at a maximum baud rate of 1Mbps can be sent at a time which would be sufficient for communications in a car. Higher layer protocols like CANopen etc. can be used if more than 8 bytes of data has to be communicated at a time. The Fig. 3 shows a CAN bus attached to three nodes.

Fig. 3. CAN bus with three nodes

The CAN protocol belongs to a class of protocol called carrier sense multiple access/collision avoidance (CSMA/CA).

As all the nodes on the bus listen to any transmitting node, it avoids collision of data. Only one node can access the bus at a time and the bus access is assigned to the nodes using non destructive bus arbitration techniques. In the listen mode, all the nodes receive the same message and a node acts on the message only if it is relevant to the

node. The CAN protocol retransmits the lost or corrupted messages whenever the bus is idle.

The logic levels of the CAN bus are defined as dominant level (TTL = 0V) and recessive level (TTL = 5V). These levels gain importance when bus arbitration comes into picture. When a dominant bit and a recessive bit request for bus access, the bus access will be assigned to the dominant bit.

A. CAN message frame

Fig. 4. CAN message frame

The CAN message frame starts with a start of frame (SOF) which is a dominant bit and is followed by an identifier which can be of 11 bits or 29 bits. The remote transmission request bit is used to indicate whether the message frame is a data frame or remote frame. The control bits are used to indicate the length of the data. The cyclic redundancy check (CRC) is used by the receiver to check errors in the received sequence. The acknowledgement bit (ACK) is sent by the receiver to the transmitter if no errors occur in the transmission and acknowledge bit is followed by end of file (EOF), which is a seven bit frame which denotes the end of the message frame. There are four different message frames in a CAN protocol

- Data frame – used to send data from one node to one or more than node.
- Remote frame – used to request data from a particular source node.
- Error frame – used to send an error signal on to the bus either by the receiver or sender during transmission or reception.
- Overload frame – used to request delay (usually 3 bit times) between two data or remote frames.

Nodes in the CAN bus are not concerned with the configuration of the system which is another reason why interfacing of the nodes is easy. For example, when a message frame is transmitted onto the bus, the receiving node is not concerned with the information of the node which has transmitted to it. The message IDs in the data/remote frame are used for message filtering i.e. to decide whether the data received is relevant or not. This is also used in bus arbitration techniques to acquire bus access. The ID for each message frame will be present in the arbitration field of the data and remote frame.

The ID has two forms, standard ID (11 bits) and extended ID (29 bits). Each message frame should have a different ID.[2]

B. Error Detection

The CAN protocol is highly reliable and error resistant. The following series of procedures are followed to detect errors in transmission of a frame

- Bit monitoring
- Checksum check
- Bit stuffing
- Frame check
- Acknowledgment check

C. Fault Confinement

The CAN protocol performs error handling without ruining the normal operation of the bus. If a node is unable to read the message or detects any error in the message, the entire bus will be notified about the error and the transmitting node retransmits the message frame. Consider a case where a node transmits a high priority message which is corrupted and the receiving node transmits an error frame on the bus, then the transmitting node again sends the corrupted message in response to the error frame. As the message frame of the transmitting node is of high priority, the bus access will not be given to low priority messages which makes most of the bandwidth alloted in transmitting corrupted messages. In this case, the CAN protocol disconnects this type of nodes from the bus. If a node reports error for a certain amount of time or for a predefined count value, the node will be disconnected from the bus and the node would be in bus off state.

D. Interframe space

Interframe space is the delay between transmission of two message frames. It is usually 3 bit times. A receiver can put a request for more delay between the message frames using the overload frame.

IV. DESIGN

The main goal of the design is to distribute the control over the CAN bus. The initial design of the autonomous ATV is as shown in the Fig. 5.

In the above design (Fig. 6), many I/O ports are used for interfacing devices to the microcontroller, which increases the interconnections (wires) and makes the hardware look clumsy. Instead we can substitute all the interconnections by using a single two wire CAN bus. The proposed design with the CAN bus is as shown in the Fig. 6. The block diagram shows the RX62N microcontroller and several other devices connected to it through the CAN bus. The CANL and CANH are the two pins from the RX62N which create a bus for communication. The microcontroller used above the h-bridge serve the purpose of generating signals to the h-bridge in correspondance with the data received from the CAN bus. The radio receiver and the throttle servo are directly connected to the RX62N. The PC sends data on to the CAN bus, signaling the ATV to steer in the corresponding direction

Fig. 5. Design without CAN bus

Fig. 6. Design with the CAN bus

based upon the data from the LIDAR,GPS and Camera. A RS232-CAN module or a USB-CAN module can be used for the communication of data from PC to RX62N.
The advantages of implementing the CAN bus on the ATV would be

- Decreased wire harnesses
- Easy installation of devices on to the bus
- Error detection and fault confinement
- Does not affect the operation of the bus if any node breaks down
- Real time performance
- Robust to noisy environments

V. IMPLEMENTATION OF THE DESIGN ON RENESAS RX62N

The block diagram as shown in Fig. 6 is the design that would be implemented using RX62N microcontroller.

A. Implementing CAN protocol on RX62N

The CAN protocol has been set up between two RX62N microcontrollers. The RX62N provide internal resistance of 120 ohms between CANL and CANH for impedance matching on the bus. The RX62N registers are used to set up the CAN protocol for required specifications. Test modes are available in the RX62N microcontroller which are used to test the CAN bus without connecting the external bus. When a test mode is enabled, the CAN transmit pin (CTx0) is virtually connected to the CAN receive pin (CRx0) and checks for the operation of the CAN protocol. There are three different kinds of test modes available, each one featuring different forms of testing the CAN bus.

1. Listen only mode: In this mode normal data frames and remote frames can be sent but with only recessive bits. This mode is mainly used for baud rate detection
2. Self test mode 0: In self test mode the microcontroller receives its own transmitted messages, stores them in a mailbox and sends an ACK bit. This mode can be used to test the operation of the CAN protocol without connecting the external bus. The CRx0 and CTx0 pins of RX62N should be connected to the transceiver.
3. Self test mode 1: In self test mode the microcontroller receives its own transmitted messages, stores them in a mailbox and sends an ACK bit. This mode can be used to test the operation of the CAN protocol without connecting the external bus. No external connections are needed for the CTx0 and CRx0 pins of RX62N.

The baud rate of the CAN protocol can be set to a maximum of 1 Mbps. It can be varied using the external clock source, settings of the internal clock source and prescaler values set up in the registers. The formula used to calculate the baud rate of CAN bus is

BitRate[Bps] = SegmentlengthFCANCLK

where segment length is used to synchronize data between two nodes as each node may or may not have the same clock frequency.

RX62N microcontroller has totally 32 mailboxes that can be configured in two different modes

Normal mailbox mode: In this all 32 mailboxes can be configured to either transmission or reception mailboxes. • FIFO mailbox mode: In this 24 mailboxes can be configured to either transmission or reception mailboxes. In the remaining 8 mailboxes, first four mailboxes can be configured as transmission and the other four mailboxes are configured as reception mailboxes.

A transmission mailbox carries the data to be transmitted onto the bus and the reception mailbox stores the received data. A status register is available in RX62N which records the status of all the events that occur in a particular node. Bus off state of a node can also be checked by reading this register. The following is the pseudo code for the transmission of data on to the CAN bus using polling

1. Enable CAN module.
2. Enable the ports for transmission and reception.
3. Switch CAN module to reset mode.
4. Select the type of mailbox (normal or FIFO), ID (standard or extended).
5. Set up required clock speed and corresponding baud rate.
6. Select the required test mode if needed.
7. Switch CAN module to halt mode or operation mode.
8. Select a mailbox for transmission and clear the mailbox.
9. Select the length of the data. Set the id and enter the required transmitting data into the mailbox.
10. Clear the transmission enable bit of the CAN bus.
11. Select the type of transmission (one shot or continuous). 12) Set the transmission enable bit of the CAN bus.
12. When the sending of data is successful, the sent data status flag will be enabled.
13. Clear the sent data status flag for the next transmission.
14. Clear the transmission enable bit and set it again for the next transmission.

The following is the pseudo code for receiving of data on the CAN bus using polling

1. Enable CAN module.
2. Enable the ports for transmission and reception.
3. Switch CAN module to reset mode.
4. Select the type of mailbox (normal or FIFO), ID (standard or extended).
5. Set up required clock speed and corresponding baud rate.
6. Select the required test mode if needed.
7. Switch CAN module to halt mode or operation mode. 8) Select a mailbox for receiving and clear the mailbox.
8. Set the required id in the mailbox to receive data only with that particular id.
9. Switch CAN module to halt mode.
10. Enable the mask for message filtering with a particular id.
11. When the data is received the new data status flag will be enabled.
12. As soon as the new data status flag is enabled, save the data into a temporary variable and clear the new data status flag.

VI. CONCLUSION

This paper describes about implementing the CAN bus on automated vehicles. The operation of the CAN protocol has been tested on RX62N microcontroller. As the CAN protocol is compatible with many of the devices it can be implemented in any of the embedded systems for real time transmission of data with less number of interconnections and large number of devices to communicate.

VII. FUTURE WORK

The future work on the ATV would be to make it as an automated vehicle with help of the camera and the LIDAR.

A. PID for steering

Currently the steering time response of the ATV has a large overshoot and steady state error. Proportional Integral Derivative (PID) controllers are able to improve these time response characteristics as well as possibly alleviate wear and tear on the internal components of motors. For more information on PID controllers and control systems, see [4].

B. Emergency-Stop for safety

Although a remote stop system is already implemented that turns off the accelerator if the radio signal is either not

detected or an emergency switch on the remote is thrown, it does not apply the brake in either of these cases. Also, this current stop function for the robot is implemented in software on the Renesas microcontroller along with the control and communication algorithms. Ideally in autonomous mobile robots, an emergency stop function should be implemented separately from any other software and should shut off the engine and apply the brakes.

C. Dead Reckoning for localization

Dead reckoning devices such as Inertial Measurement Units (IMU) and Rotary encoders for the wheels can be used as a backup in case of GPS outages. These devices can also provide somewhat accurate data for short distances and are often sampled at a faster rate than most GPSs.[1]

REFERENCES

[1] Richard A. McKinney, Malcolm J. Zapata, James M. Conrad, Thomas W. Meiswinkel and Siddharth Ahuja, Components of an Autonomous All-Terrain Vehicle, IEEE SoutheastCon, 2010.
[2] Wilfried Voss, A Comprehensive Guide to Controller Area Network, Copperhill Media Corporation, 2005-2008.
[3] Renesas RX62N Hardware Manual Febraury 2010 (Rev. 0.50)
[4] Renjun Li, Chu Liu and Feng Luo, A Design for Automotive CAN Bus Monitoring System,, IEEE Vehicle Power and Propulsion Conference (VPPC), September 3-5, 2008.
[5] Ping Ran, Baoqiang Wang and Wei Wang, The Design of Communication Convertor Based on CAN Bus, IEEE international conference on industrial technology, 2008.
[6] R. Bosch, GmbH, CAN specification 2.0
[7] Benjamin C. Kuo, M. Farid Golnaraghi, Automatic Control Systems, Eight Edition," John Wiley & Sons, Inc. 2003.
[8] http://www.can-cia.org
[9] http://www.copperhillmedia.com/cannewsletter/
[10] http://www.technologyuk.net/telecommunications/industrial networks/ can.shtml

