ABSTRACT-

CAN is a serial bus protocol to connect individual systems and sensors as an alternative to conventional multi-wire looms. It allows automotive components to communicate on a single or dual-wire networked data bus up to 1Mbps. This paper describes and gives an overall idea to design an autonomous all terrain vehicles (ATV) using CAN bus. It also gives importance on the operation and advantages of the CAN protocol in automobiles. The ATV used for design automotive automobile and gives a brief explanation regarding the different parts of the ATV. The high microcontroller configured microcontroller is used as a CAN controller which creates a single two wire bus through which electronic control units (ECU) in the automobiles communicate with each other to have control over the environment.

Key Words- CAN bus, ECU, ATV, microcontroller.

I. INTRODUCTION

Now a day’s automakers are making vehicle more comfortable and convenience by integrating more electric components using different automation which leads to incorporate enormous amount of wiring in the vehicle before releasing CAN bus. CAN Bus vehicle enter in to the market in 1986. By reducing the vehicles wiring by 2km, the vehicles overall weight was significantly reduced by at least 50kg.

CAN-bus line usually interconnects to a CAN controller between line and host at the node. It gives the input and gets output between the physical and data link layers at the host node. The CAN controller has a BIU (bus interface unit consisting of buffer and driver), protocol controller, status-cumcontrol registers, receiver-buffer and message objects. These units connect the host node through the host interface circuit.

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.

Many Author discussed many concept regarding the application of CAN bus, some of them I am trying to mention is that, Richard A. McKinney, Malcolm J. Zapata, James M. Conrad,[1] have discussed about the different components of CAN bus implemented in different automobiles. Wilfried Voss [2] given a beautiful idea about CAN bus. Also the author Renjun Li, Chu Liu and Feng Luo [3] have given how Automotive vehicle can be design using CAN bus protocols.

This work describes and gives an overall idea to design an autonomous all terrain vehicles (ATV) using CAN bus. It also gives importance on the operation and advantages of the CAN protocol in automobiles. The ATV used for design automotive automobile and gives a brief explanation regarding the different parts of the ATV.

II. ARCHITECTURE OF CAN BUS

The layered structure of a CAN node can be represent as

<table>
<thead>
<tr>
<th>Layer</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Layer</td>
<td></td>
</tr>
<tr>
<td>Object layer</td>
<td></td>
</tr>
<tr>
<td>Transfer layer</td>
<td>Fault detection</td>
</tr>
<tr>
<td></td>
<td>Error Detection</td>
</tr>
<tr>
<td></td>
<td>Message Validation</td>
</tr>
<tr>
<td></td>
<td>Acknowledgement etc..</td>
</tr>
<tr>
<td>Physical layer</td>
<td></td>
</tr>
</tbody>
</table>
### Bit Representation.

Transmission Medium.

CAN protocol is a message-based protocol, not an address based protocol. This means that messages are not transmitted from one node to another node based on addresses. Embedded in the CAN message itself is the priority and the contents of the data being transmitted. All nodes in the system receive every message transmitted on the bus (and will acknowledge if the message was properly received). It is up to each node in the system to decide whether the message received should be immediately discarded or kept to be processed. A single message can be destined for one particular node to receive, or many nodes based on the way the network and system are designed.

### CAN MESSAGE FRAME

CAN protocol defines four different types of messages (or Frames). The first and most common type of frame is a Data Frame. This is used when a node transmits information to any/all other nodes in the system. Second is a Remote Frame, which is basically a Data Frame with the RTR bit set to signify it is a Remote Transmit Request (see Figure 2 and Figure 3 for details on Data Frames). The other two frame types are for handling errors. One is called an Error Frame and one is called an Overload Frame. Error Frames are generated by nodes that detect any one of the many protocol errors defined by CAN.

Overload errors are generated by nodes that require more time to process messages already received. Data Frames consist of fields that provide additional information about the message as defined by the CAN specification. Embedded in the Data Frames are Arbitration Fields, Control Fields, Data Fields, CRC Fields, a 2-bit Acknowledge Field and an End of Frame. The Arbitration Field is used to prioritize messages on the bus. Since the CAN protocol defines a logical 0 as the dominant state, the lower the number in the arbitration field, the higher priority the message has on the bus. The arbitration field consists of 12-bits (11 identifier bits and one RTR bit) or 32-bits (29 identifier bits, 1-bit to define the message as an extended data frame, an SRR bit which is unused, and an RTR bit), depending on whether Standard Frames or Extended Frames are being utilized.

The current version of the CAN specification, version 2.0B, defines 29-bit identifiers and calls them Extended Frames. Previous versions of the CAN specification defined 11-bit identifiers which are called Standard Frames.

The Control Field consists of six bits. The MSB is the IDE bit (signifies Extended Frame) which should be dominant for Standard Data Frames. This bit determines if the message is a Standard or Extended Frame. In Extended Frames, this bit is RB1 and it is reserved.

The next bit is RB0 and it is also reserved. The four LSBs are the Data Length Code (DLC) bits. The Data Length Code bits determine how many data bytes are included in the message. It should be noted that a Remote Frame has no data field, regardless of the value of the DLC bits.
The Data Field consists of the number of data bytes described in the Data Length Code of the Control Field.

The CRC Field consists of a 15-bit CRC field and a CRC delimiter, and is used by receiving nodes to determine if transmission errors have occurred.

<table>
<thead>
<tr>
<th>SOF</th>
<th>Identifier</th>
<th>RTR</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>One dominant bit</td>
<td>11 0R 29 BITS</td>
<td>1-Bit</td>
<td>4-Bit</td>
</tr>
<tr>
<td>DATA</td>
<td>CRC</td>
<td>ACK</td>
<td>EOF</td>
</tr>
<tr>
<td>0-8 bytes</td>
<td>15- bits</td>
<td>2- bits</td>
<td>7 recursive bits</td>
</tr>
</tbody>
</table>

Fig-2, CAN Message Frame

IV. ERROR DETECTION

CRC Error
A 15-bit Cyclic Redundancy Check (CRC) value is calculated by the transmitting node and this 15-bit value is transmitted in the CRC field. All nodes on the network receive this message, calculate a CRC and verify that the CRC values match. If the values do not match, a CRC error occurs and an Error Frame is generated. Since at least one node did not properly receive the message, it is then resent after a proper intermission time.

Acknowledge Error
In the Acknowledge Field of a message, the transmitting node checks if the Acknowledge Slot (which it has sent as a recessive bit) contains a dominant bit. This dominant bit would acknowledge that at least one node correctly received the message. If this bit is recessive, then no node received the message properly. An Acknowledge Error has occurred. An Error Frame is then generated and the original message will be repeated after a proper intermission time.

Form Error
If any node detects a dominant bit in one of the following four segments of the message: End of Frame, Interframe Space, Acknowledge Delimiter or CRC Delimiter, the CAN protocol defines this to be a form violation and a Form Error is generated. The original message is then resent after a proper intermission time. (see Figure 2 and/or Figure 3 for where these segments lie in a CAN message).

Bit Error
A Bit Error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit when monitoring the actual bus level and comparing it to the bit that it has just sent.

In the case where the transmitter sends a recessive bit and a dominant bit is detected during the Arbitration Field or Acknowledge Slot, no Bit Error is generated because normal arbitration or acknowledgment is occurring. If a Bit Error is detected, an Error Frame is generated and the original message is resent after a proper intermission time.

Stuff Error
CAN protocol uses a Non-Return–to-Zero (NRZ) transmission method. This means that the bit level is placed on the bus for the entire bit time. CAN is also asynchronous, and bit stuffing is used to allow receiving nodes to synchronize by recovering clock information from the data stream. Receiving nodes synchronize on recessive to dominant transitions.

If there are more than five bits of the same polarity in a row, CAN will automatically stuff an opposite polarity bit in the data stream. The receiving node(s) will use it for synchronization, but will ignore the stuff bit for data purposes.

If, between the Start of Frame and the CRC Delimiter, six consecutive bits with the same polarity are detected, then the bit stuffing rule has been violated. A Stuff Error then occurs, an Error Frame is sent, and the message is repeated.

V. SYSTEM 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.3. Where 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. 3. The block diagram shows the microcontroller and several other devices connected to it through the CAN bus.

The CANL and CANH are the two pins from the which create a bus for communication. The microcontroller used above the h-bridge serve the
purpose of generating signals to the h-bridge in correspondence with the data received from the CAN bus.

The radio receiver and the throttle servo are directly connected to the micro controller. The PC sends data on to the CAN bus, signaling the ATV to steer in the corresponding direction. Fig-3, Design of proposed Method.

based upon the data from the LIDAR,GPS and Camera. This-CAN module or a USB-CAN module can be used for the communication of data from PC to Microcontroller.

VI. IMPLEMENTATION & RESULT

To test the CAN bus Different kinds of test modes available each one featuring different forms of testing.

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.

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.

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 advantages of implementing the CAN bus on the ATV would be

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

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.
13) When the sending of data is successful, the sent data status flag will be enabled.
14) Clear the sent data status flag for the next transmission.
15) 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 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.
13) When the sending of data is successful, the sent data status flag will be enabled.
14) Clear the sent data status flag for the next transmission.
15) Clear the transmission enable bit and set it again for the next transmission.
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.
9) Set the required id in the mailbox to receive data only with that particular id.
10) Switch CAN module to halt mode.
11) Enable the mask for message filtering with a particular id.
12) When the data is received the new data status flag will be enabled.
13) As soon as the new data status flag is enabled, save the data into a temporary variable and clear the new data status flag.

VII. CONCLUSION

This paper describes about implementing the CAN bus on automated vehicles. The operation of the CAN protocol has been tested on different 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 wire clumsiness and large number of devices to communicate.

VIII. FUTURE WORK

This can be implement by including GPRS and GSM Technology.

REFERENCES

[10]www.technologyuk.net/telecommunications/industrialnetworks/can.shtml