International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 11, Issue 4, July – August 2022 ISSN 2278-6856

Implementation of Controller Area Network Transport Protocol for Electric 2 Wheelers

Poorvi Patil[1], Usha Rani K.R[2], Venkatanarasimharao Medam[3]

[1]PG Student, [2]Associate Professor
Department of ECE, R V College of EngineeringBengaluru, India

[3]Chief of Advance EngineeringResearch and Development
Greaves Electric Mobility
Bengaluru, India

***Abstract*—As the number of electronic components are massively increasing in the automobiles, there is a need for increased data size. Rather than modifying the hardware, it is more economical and less complex to incorporate the existing bus structures with increased data limit. In this paper, we a) propose implementing the most familiar Controller Area Network Transfer Protocol (CAN-TP) called ISO 15765-2, or ISO-TP (Transport Layer) for electric vehicles with which the data limit of 8 bytes can be increased up to 4095 bytes of payload per message packet. And, b) The complications of changing the hardware is avoided and existing devices are used without interference in communication. One of the main ideas is to bring in some changes in the protocol. c)The implemented protocol is used to transmit Unified Diagnostics Services (UDS) data and to receive bootloader data in electric vehicles. In this paper, the CAN transfer protocol will be used as a communication protocol for bootloader and UDS data transmission. It is implemented using S32K144, a special purpose microcontroller from NXP for automotive and industrial applications.**

Keywords—Bootloader, CAN, ISO-15765, CAN-TP, ECU, ISO-TP, UDS

I. INTRODUCTION

The automotive industry is one of the most important sectors of the modern economy. Electric two-wheelers have begun to take center stage in the automotive industry. As the highways of the world's cities become increasingly congested, an electric two-wheeler promises to be a more environmentally responsible, sustainable, and practical solution for local commuting. The technology used in the electric cars and bikes are almost similar, with the changes in the number of Electronic Control Units used. The car has more ECUs, and the system is complex compared to two-wheelers. The ECUs are helping the 2 wheelers to become more efficient and more intelligent. Interconnection between different ECUs is essential, which would be very costly and complex through conventional wiring. In contrast, CAN protocol provides an economical and appropriate solution.

CAN is basically designed to replace the conventional wiring used in the old days in the automobile for communication between ECUs. The main problem faced using the standard CAN bus was the data transfer limitation of 8 bytes. Using CAN Transfer Protocol, more than 8-bytes of data can be transmitted; hence this limitation can be overcome without any modification to the hardware. The developed CAN-TP code can be used to transfer data up to 4095 bytes as per the ECU requirements.

The software used for the proposed method is the S32 Design Studio from NXP, and the hardware NXP's S32K144 Evaluation Board (EVB), PCAN View software and PCAN device hardware is used to transmit and receive CAN messages to test the developed protocol. The bit rate is set to 500 Kbits/s. The number of message buffers present in the CAN module are 32, each with a data length of 8 bytes. In this, every message buffer can be used for both receiving and transmitting data. There are different modes of operations for the CAN module are available. An on-board CAN transceiver is present to provide differential transmit and receive capability to the respective peripheral of the microcontroller. In this paper 1) CAN TP is configured for a 32-bit microcontroller. 2) The CAN transfer protocol will be used for bootloader and UDS applications.

II. PROPOSED ARCHITECTURE

A.System Architecture

CAN is a communication protocol used in automobiles for the transfer of data; hence acts as an interconnection medium between different ECUs present in the automobile. The ECUs are referred as CAN nodes. CAN is a messaged-based protocol which means any node can send message through the CAN bus. Based on arbitration, the CAN bus will be allocated to the node. The node with the first 0 bit wins the arbitration during the arbitration, as the arbitration result will be decided by AND logic. Connection of different nodes to CAN bus is shown in figure 1.

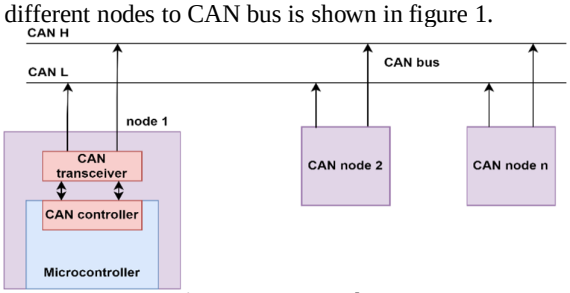

Figure 1 CAN architecture

B. Electronic Control Unit (ECU)

In automobiles ECU is an embedded system which controls different functions. Each ECU is programmed to handle a particular function in an automobile, which improves the overall performance of the system. The data from the sensors can be given as input to the ECU and can be processed as per the requirement and can also store it to use in the future. Based on the different applications, different ECU architectures are developed. Few examples of ECUs in electric automobiles are engine control module, battery management system, UDS, bootloader

C. Controller Area Network (CAN)

The ECUs in automobiles were connected using standard wiring before the development of CAN protocol that led to increased complexity and cost which resulted in reducing the overall performance of the vehicle. All these problems are overcome using the CAN protocol designed by Robert Bosch. CAN needs only 2 wires i.e., CAN High and CAN Low buses for entire communication in a vehicle. The CAN message flow is shown in figure 2. Upon the start signal the CAN message will be sent, where arbitration takes place to decide which node should send the data through CAN. Multiple nodes can send data to the CAN at a time but it is through arbitration to which node the CAN bus should be allocated will be decided. The node winning the arbitration will get to transfer data through CAN and the remaining nodes will undergo arbitration again.

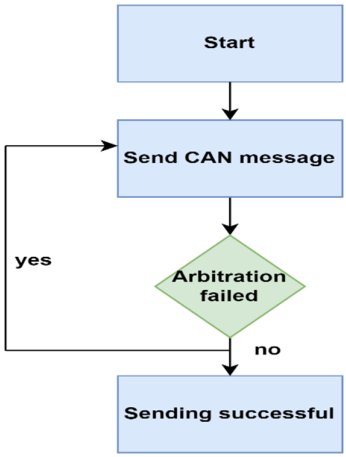

Figure 2 CAN flowchart

D. CAN Transfer Protocol (CAN TP)

Using standard CAN protocol, it was impossible to transfer more than 8–bytes of the data. This problem can be overcome using the CAN-TP or ISO 15765-2 protocol. The ISO transport protocol works on the OSI layer model's fourth layer (transport layer). The maximum data transfer limit of ISO TP is 4095 bytes. The main function of CAN transfer protocol is to segment the CAN message into multiple frames and transfer the data. The implementation of the CAN transfer protocol is given in figure 3 in the flow chart of CAN TP. When the CAN message is received first, it checks the length of the message. If the message length is less than 7 bytes, then the message is transferred as a single frame. If the CAN message is more than 7 bytes, then the message is segmented and transferred using the first frame and consecutive frames. Initially in the first frame, data of 6 bytes will be transferred. The CAN message's data length is present in the first frame, i.e., the total number of bytes to be transferred. Once the data is transferred using the first frame, it waits for the flow control. The flow control frame contains block size, which gives the receiver buffer size. According to the block size given by the flow control frame, the data will be transferred using consecutive frames.

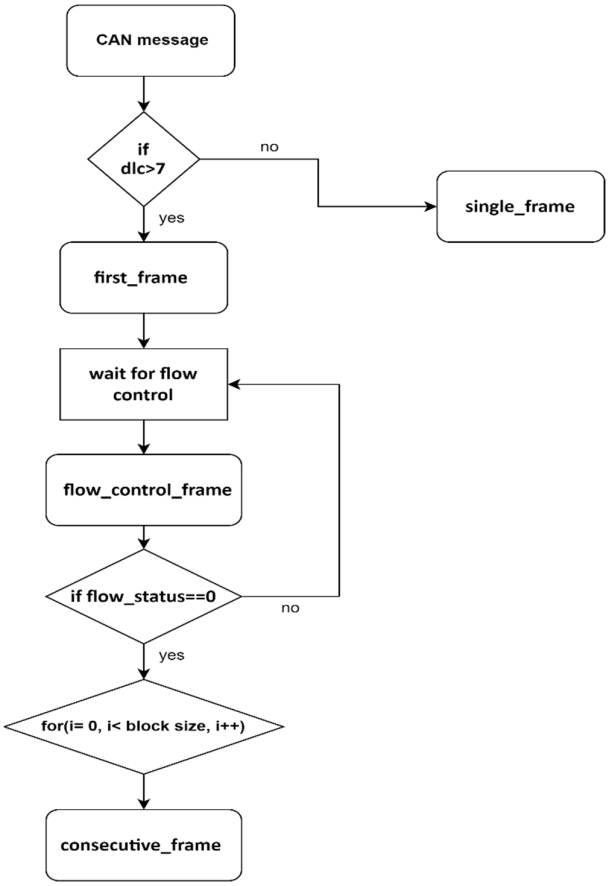

Figure 3 CAN TP flow chart

This implementation can be done on the data link layer also. So, to send the data like CAN, the CAN-TP Protocol has been designed. To transfer data in multiple frames, there are four different frame types in CAN TP protocol:

1. Single Frame.
2. First Frame.
3. Consecutive Frame.
4. Flow Control Frame.

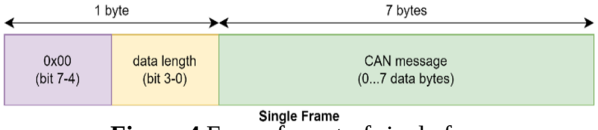

Figure 4 Frame format of single frame

SINGLE FRAME (SF):
If CAN message length is 7 bytes or less, then data will be transferred using single frame. The first byte contains the protocol information, which is further divided into two parts as a type of frame from 7-4 bits, and data length will be stored in 3-0 bits in the first byte of CAN frame. The remaining 7 bytes will be used for the CAN message. The frame format and the information of data carried by respective bytes of a single frame are given in figure 4.

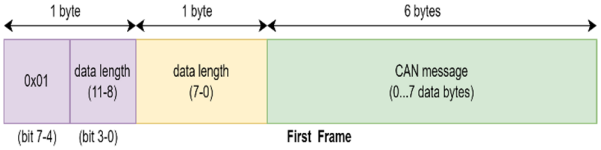

Figure 5 Frame format of the first frame

FIRST FRAME (FF):
If the CAN message length is greater than 7 bytes, then the data will be segmented and transferred as multiple frames. The first frame is an initial message of the multi-frame structure in the CAN TP protocol. The first frame structure contains frame type in 7-4 bits of the first byte, 11-8 bits of data length in LSB of the first byte, and the remaining 7-0 bits in the 2nd byte. 12 bits will be used for data length, which can be up to 4095 bytes. The remaining 6 bytes will be used for transferring CAN messages. This frame acts as the first frame of the consecutive frames. Hence during first time transfer of data, the serial number in the consecutive frame starts from 1 instead of 0. The data length will be obtained through this frame. The information of data carried by respective bytes of the first frame is given in figure 5.

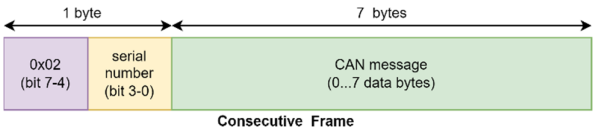

Figure 6 Frame format of consecutive frame

CONSECUTIVE FRAME (CF):
control should be sent to obtain block size i.e., size of the receive buffer (which is further explained in flow control frame). After receiving the flow control frame, if the receiver is ready to receive, the remaining CAN data will be transferred using consecutive frames. Each consecutive frame CAN carry 7 bytes of data. The structure of the consecutive frame is given in the figure. The first byte contains type of frame and serial number. The serial number refers to the order in which frames are being sent. If the block size is 4, the 4 consecutive frames will be sent. For example, if one wants to send 30 bytes of a CAN message, then initially first 6 bytes will be sent using first frame, to transfer the remaining 24 bytes consecutive frame will be used. If the block size given by flow control is 4 then in first 3 consecutive frames, 21 bytes of data will be transferred, each carrying 7 bytes. The 4th consecutive frame carries remaining 3 bytes of data. This is how data will be segmented and transferred using the consecutive frame. The information of data carried by respective bytes of the consecutive frame is given in figure 6.

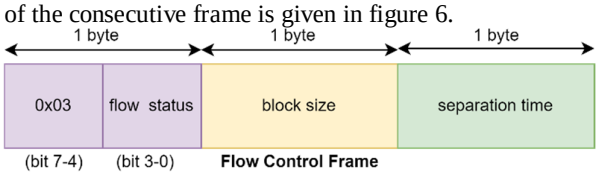

Figure 7 Frame format of flow control frame

FLOW CONTROL FRAME (FC): The receiver details i.e., buffer availability, timing etc are sent to the sender through the flow control frame. The flow control frame consists of 3 bytes of Protocol Control Information (PCI). The first byte is further divided into frame type (7-4 bits) and flow status (3-0 bits). The flow status indicates the receiver's availability, 0 indicates the receiver is ready to receive data, 1 indicates wait, 2 indicates overflow i.e., the receive buffer is full. The next byte gives block size i.e., how many bytes of data can be sent. The number of consecutive frames depends on this parameter. The 3rd byte gives separation time, it's the minimum time difference between each consecutive frames. Based on the data obtained by flow control frame, the consecutive frames will be sent accordingly. The information of data carried by respective bytes of a single frame is given in figure 7.

E. Bootloader and UDS
In this application, the CAN transfer protocol is being implemented for bootloader and UDS applications, requiring more than 8 bytes to transfer the data. A bootloader is software that updates the new firmware without externally tampering the vehicle. Updating the firmware is flashing the data into program memory (non-volatile memory). The ROM of the microcontroller memory includes two partitions, program flash (P-flash) where the application code will be stored and the data flash (D-flash) in which the bootloader code is stored which will be protected. The bootloader and application codes run interchangeably as per the user requirements or new firmware update requests. The CAN communication protocol is used to establish the communication between the host (Personal Computer - PC) and server (Target ECU). CAN protocol in the Transport Layer enables transmission of more than 8 bytes of data transfer which is necessary for fast flashing.
UDS protocol (ISO 14229) is a standard protocol used by the manufacturers and it is implemented over CAN. UDS offers various services, based on the requirement, the developer will include these services. The UDS communication is performed as a client server relationship. The client is a tester-tool and the server is a vehicle ECU. For instance, a CAN bus interface is connected to the OBD2 connector of a car and UDS requests are sent to the vehicle. Here it is assumed the target ECU supports UDS services and will respond accordingly.

III. RESULTS

Receive / Transmit | Trace | PCAN-USB

Recording... | 417.5713 s | 0.01 % | Ring Buffer | Rx: 7 | Tx: 0

Time	CAN-ID	Rx/Tx	Type	Length	Data
29.0672	100h	Rx	Data	2	01 1C
126.5085	100h	Rx	Data	3	02 1C 1B
172.9263	100h	Rx	Data	4	03 1C 1B 1A
211.6990	100h	Rx	Data	5	04 1C 1B 1A 1B
260.2971	100h	Rx	Data	6	05 1C 1B 1A 1B 1D
324.6705	100h	Rx	Data	7	06 1C 1B 1A 1B 1D 1E
417.5713	100h	Rx	Data	8	07 1C 1B 1A 1B 1D 1E 1F

Figure 8: Single frame

In the figure 8 the timing column shows the time period at which the messages are recorded, the CAN ID is the ID of the CAN message received/transmitted, Rx/Tx indicates received or transmitted data, and type indicates the type of the message frame, the length gives length of the CAN message, and the data is the CAN message data. Figure 8 is an example of the transfer of CAN data using a single frame, through which the data will be sent if the message length is < 7 bytes. Here data is some random data sent for checking the developed protocol. In the 1st message frame, 1 byte of data is being transferred. As we can observe in figure 8 the MSB of 1st byte gives frame type 0, which stands for a single frame in CAN transfer protocol, and the LSB of the 1st byte gives the size of the data. Here we have checked the developed protocol by giving 1-7 bytes of data.

Receive / Transmit | Trace | PCAN-USB

Recording... | 590.1873 s | 0.01 % | Ring Buffer | Rx: 12 | Tx: 1

Time	CAN-ID	Rx/Tx	Type	Length	Data
49.3418	100h	Rx	Data	8	10 4E 1A 1A 1A 1A 1A 1B
59.9921	511h	Tx	Data	3	30 0C 00
141.8279	100h	Rx	Data	8	21 1A 1A 1A 1A 1B 1A
173.9209	100h	Rx	Data	8	22 1A 1A 1A 1A 1B 1A
211.5777	100h	Rx	Data	8	23 1A 1A 1A 1A 1B 1A
248.5888	100h	Rx	Data	8	24 1A 1A 1A 1A 1B 1A
277.5028	100h	Rx	Data	8	25 1A 1A 1A 1A 1B 1A
301.0551	100h	Rx	Data	8	26 1A 1A 1A 1A 1B 1A
332.5051	100h	Rx	Data	8	27 1A 1A 1A 1A 1B 1A
358.8842	100h	Rx	Data	8	28 1A 1A 1A 1A 1B 1A
397.7259	100h	Rx	Data	8	29 1A 1A 1A 1A 1B 1A
449.1225	100h	Rx	Data	8	2A 1A 1A 1A 1A 1B 1A
590.1873	100h	Rx	Data	3	2B 1C 1E

Figure 9: First frame , flow control and consecutive frames

If the message length is greater than 7 bytes, the data will be segmented and transferred through the first and consecutive frames. Here we are trying to send 4E, i.e., 78 bytes of data. The first 6 bytes is transferred through the first frame in which data length will be obtained from the second byte of the frame, as observed in figure 9. Then the flow control will be sent where the buffer size is C, i.e., 12. The remaining bytes are transmitted using consecutive frames. The first message in figure 9 is the first frame of the CAN TP protocol, and the frame type of this frame is 1, which can be observed in figure 9. The second message is the flow control sent by the receiver which indicates if the receiver is free to receive or not and also gives the receiver buffer size. The frame type of this frame is 3. The remaining message frames are consecutive frames, where the first 70 bytes of data is transmitted in the first 10 consecutive frames. The consecutive frame starts from serial number 1, which is given in the LSB of 1st byte and ranges till the maximum block size provided by the receiver. Here as the block size 12 (from flow control), the first 10 consecutive frames, i.e., from 1 to A carries 70 bytes of data, and the remaining 2 bytes are sent in the last, i.e., the 11th consecutive frame. Hence a total 78 bytes of data is transmitted successfully through segmentation using CAN transfer protocol. i.e., 1st 6 bytes in the first frame and the remaining 72 bytes through consecutive frames. Therefore the CAN transfer protocol has been successfully implemented for the nxp microcontroller, and the ECUs can successfully transmit data to 4095 bytes through the implemented protocol.

IV. CONCLUSION

In this paper, the developed CAN transfer protocol will be used for data transmission of bootloader and UDS applications. Both require a communication protocol that can carry more than 8 bytes of data. This criterion was satisfied using CAN TP without any further modifications to the hardware. Implementation of CAN TP to the standard CAN protocol gives enough bandwidth for available ECUs. The CAN low-level drivers are configured. The test and application results show that the developed protocol is practical and reliable.

REFERENCES

[1] R. G. Lazar and C. F. Caruntu, "Simulator for the Automotive Diagnosis System on CAN using Vector CANoe Environment," 2020 24th International Conference on System Theory, Control and Computing (ICSTCC), 2020, pp. 705-710, doi: 10.1109/ICSTCC50638.2020.9259683.

[2] Du, Li & Xie, Peng & Zhou, Bing & Yu, Yongyan & Wan, Juan & Hu, Haixia & Cui, Lianghao. (2019). UDS in CAN flash programming. IOP Conference Series: Materials Science and Engineering. 490. 072060. 10.1088/1757-899X/490/7/072060.

[3] P. R. Sawant and Y. B. Mane, "Design and Development of On-Board Diagnostic (OBD) Device for Cars," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-4, doi: 10.1109/ICCUBEA.2018.8697833.

[4] Rohith .N, Sharmila. K. P, 2018, Implementation of Real Time Vehicle Data Acquisition and Sending it as SMS and E-Mail Alerts During Emergency, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCESC – 2018 (Volume 6
– Issue 13)

[5] S. Dekanic, R. Grbic, T. Maruna and I. Kolak, "Integration of CAN Bus Drivers and UDS on Aurix Platform," 2018 Zooming Innovation in Consumer Technologies Conference (ZINC), 2018, pp. 39-42, doi: 10.1109/ZINC.2018.8448921.

[6] K. Khorsravinia, M. K. Hassan, R. Z. A. Rahman and S. A. R. Al- Haddad, "Integrated OBD-II and mobile

application for electric vehicle (EV) monitoring system," 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS), 2017, pp. 202- 206, doi: 10.1109/I2CACIS.2017.8239058.

[7] Yong Xie, Xin Su, Yifan He, Xuhui Chen, Gengliang Cai, Baisheng Xu and Wenjia Ye, "STM32-based vehicle data acquisition system for Internet-of-Vehicles," 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017, doi: 10.1109/ICIS.2017.7960119.

[8] Yu, Jinghua & Luo, Feng. (2016). Research on Automotive UDS Diagnostic Protocol Stack Test System. Journal of Automation and Control Engineering. 10.18178/joace.4.5.388-392.

[9] P.Nagaraju, Avinash K.R., Surendra S., Shivaprasad S., "FlexRay Protocol based an Automotive Application", International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, Volume 2, Issue 5, May 2012.