Automotive CAN Network Reverse engineering & automotive instrument cluster control tool

Design Manual

BY
THOMAS JORDAN
INSTITUTE OF TECHNOLOGY CARLOW

APRIL 2021

Contents

Introduction

Frontend

The front end of an application is one of the most important parts of creating a product, this key part of the design process is essential to the users experience and how they interact with this product because the experience and layout and aesthetic of a product is found to be just as important to a user than the features or even the stability of the application.

The main aim of this project has had a profound influence on the design of this application, just to remind you of what the aim of this project is to allow the user to enter specific details about a can bus message which is made up of multiple different aspects which the user can entered. which would then be sent onto the network within the vehicle they are connected to and this message would then make the instrument cluster within that vehicle change its values and allow the user to see a real-life visual change within the gauges.

With this application there are three main aspects which will allow me to complete the main requirements that were asked of the application and these are Initialization of the application, write/send a message to the CAN bus network and read CAN bus network messages. In the next section of this documentation, I will go in depth about how each of these three main aspects are designed and how the user will interact with these specific aspects and how some will interact with each other.

INITIALIZE APPLICATION

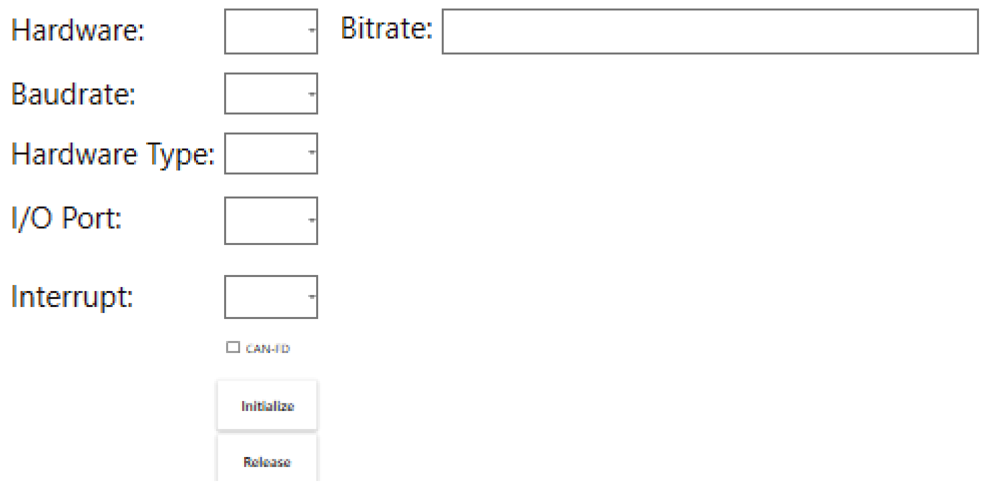


Figure 1

The main functionality of this feature is to setup and sync the PCAN USB device to the specific CAN BUS network and frequency. When designing the specific details that would be needed to initialize the application and these settings would include

Baudrate

As I spoke about in my research document within the CAN BUS network each message is sent on differing frequency depending upon different circumstances like speed of transport and severity of the message itself and these frequencies are known as Baudrates and to visualize the selecting of the baudrate. I chose to use a comboBox which allows me to insert each individual baudrate frequency which can be easily selected for use later.

Hardware

However, the “Baudrate” is not the only setting that is needed to fully initialize the USB device and those other settings include the hardware channel this setting is used to set and signify to the user what channel the USB device will be using,

Hardware Type

Next up is the hardware type and this is used to let the CAN BUS network what version of device the user will be using to send and receive messages to and from the CAN BUS network.

I/O port

This setting is used to set the I/O address which is used within the parallel port of the pcan USB device.

Interrupt

The interrupt is used to set the interrupt number used by the parallel port of the pcan USB device which is used incase of voltage shortage within a specific input line on the device’s connector.

Bitrate

This textbox is used to set the values which are used to configure the “pcan USB device” for a can FD message which is a next generation of CAN BUS messaging and allows for larger messages at a faster rate of transfer onto the network.

Can-FD

This button is simply used to let the “pcan USB device” and the CAN BUs network know to switch to the CAN FD initialization method which is through the use of the bitrate textbox and the setting of values which contain values such as the nominal and data bit time of the bit rate, sync sequence and the sjw.

Initialize

This button is simply used once all the individual values for each of the settings have been selected then once the button is clicked the pcan USB device and CAN BUS network are then given these values and are locked in and then user can use the full ability of the applications features.

Release

This button is then simply used to release those settings from the pcan USB device and CAN BUS network which once clicked the user can then change and adapt their settings to suit their devices and other CAN BUS networks.

Implementation

When in the process of creating I decided the best way for the user to interact with these settings is to create individual combo boxes which would contain preset individual values which would be used to represent the values needed to initialize the pcan USB device which I mentioned in the previous sections.

SEND MESSAGE

Once the pcan usb device has been initialized with the essential information needed to be able to move on to the next step and have the ability to send messages onto to the CAN BUS network and also receive and read messages from that same network then the user can then move on and begin to interact with the send message and read message and in this section of this document I will discuss the send message and what aspects and fields are needed to successfully send a message to the CAN BUS network.

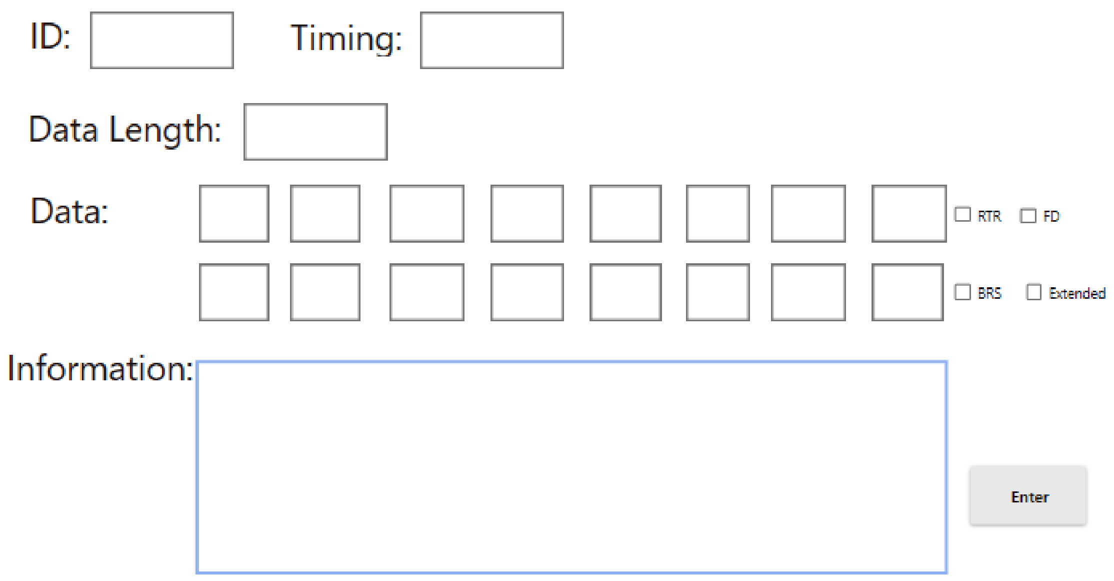


Figure 2

ID

As I spoke about in the research document the ID is used to let the CAN BUS network know how high of a priority that a specific message should get. Within the CAN BUS network there are two different types of packets that can be sent across the network and these are known as standard and extended packets and within the network the ID is used as one of the ways in which an extended packet is created and that is by adding seven extra bits to the ID which in standard size is eleven bits.

Data length

The data length allows the user to signal to the pcan usb device and the CAN BUS network what the length of the actual message is. Within the CAN BUS network as I spoke about in the research document that the size of the data length goes from zero to eight bits.

Data

This section is used to allow the user to enter the sequence of bytes that will contain the actual message. Within this section each byte represents an individual segment of the CAN BUS network and each byte can correspond to a different part of the vehicle.

Timing

Information

The information page is to be used for any errors or any device and application information that may need to be relayed back to the user in case of incorrect initialization of the pcan USB device or message data.

RTR

RTR is a key button in the use of sending a message onto the CAN BUS network because it RTR stands for remote transfer request and this button is specifically used to let the CAN BUS network whether the message being sent onto the network is a data frame or a request frame and since these two message types are extremely similar so within the arbitration field RTR is given a id bit which set to “0” if it is a data frame and is set to “1” and this is done to prevent CAN BUS message collision.

FD

With the FD button it lets the application and the user to advantage of the full size of the data that is allowed within a CAN FD message which is 64 bits which is caped at 8 bits of data size while the FD button is not checked.

BRS

With the BRS button it is used in conjunction with the use of a CAN FD message and when clicked allows the pcan USB device to know to change the speed at which the message is sent onto the CAN BUS network and this is because with the increased size of the message if the message is sent onto the network then there would be slow down of the network and eventual clog that is why this button signifies to increase the speed from “1 megabit/s” to “5 megabit/s”.

Extended

This button when clicked allows the application from to set the IDE (identifier extension) which tells the pcan USB device and the CAN BUS network whether the current message being sent by the user is a standard or extended packet and this is done by setting the IDE as 0 for standard packet and for an extended packet it is set to a 1.

READ MESSAGE

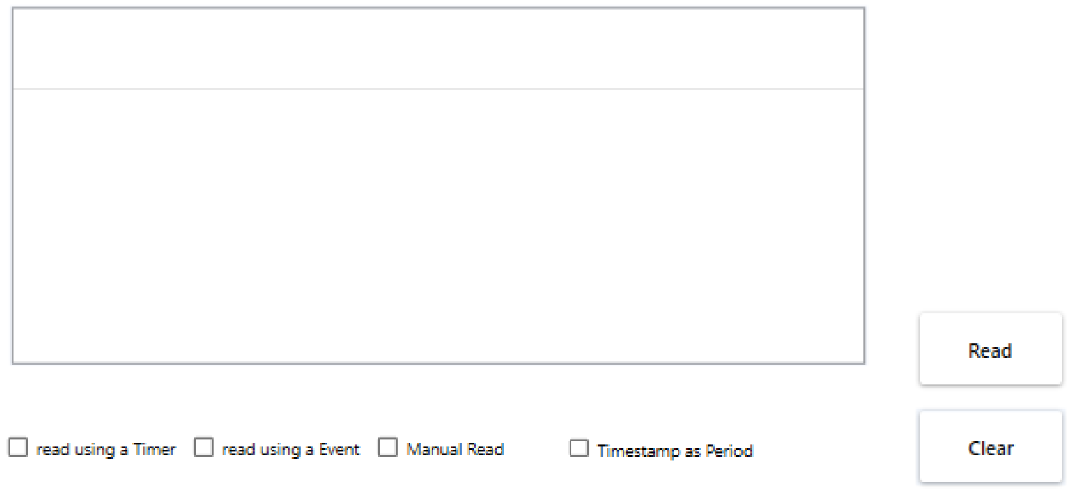
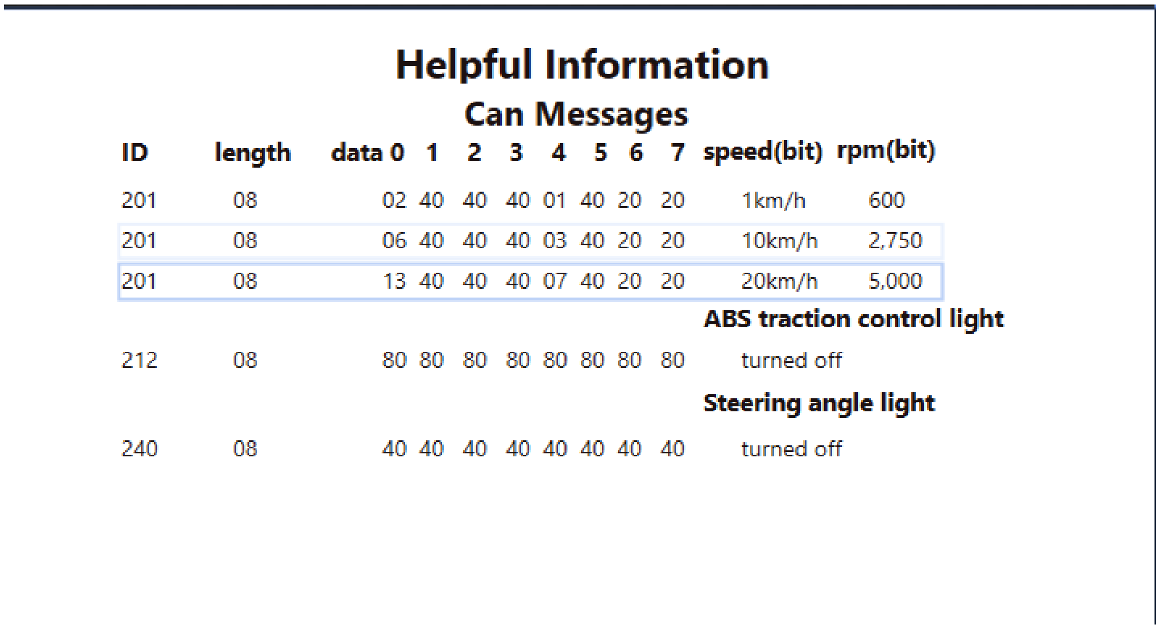


Figure 3

This function is used to read every message that is coming into that CAN BUS network which is on the specific frequency that we spoke about earlier which is set at the start of the application with the initialization of the baudrate. To create this feature, I used a WPF “listviewbox” which allows data to be presented in a list format within a box and this data is selected from a “itemsource” within the application and this “iteamsource” is the CAN BUS network with the help of the API having the ability to pull down specific attributes about that message like the type, ID, length, count, RCV time and data.

The type of message may consider whether it is a standard or extended message frame or whether it is a RTR enabled message or whether the message is a FD message or not and the rest of the information retrieved is pretty self-explanatory which consist of the ID of the message to rank priority on the network and then the length of the message and then we have the message count which is the number given to the amount of messages being retrieved and then a timestamp of when the message was retrieved from the CAN BUS network and finally the data that is retrieved from the network is finally displayed in the "listviewbox"

MESSAGE INFORMATION



Helpful Information

Can Messages

ID	length	data 0	1	2	3	4	5	6	7	speed(bit)	rpm(bit)
201	08	02	40	40	40	01	40	20	20	1km/h	600
201	08	06	40	40	40	03	40	20	20	10km/h	2,750
201	08	13	40	40	40	07	40	20	20	20km/h	5,000
										ABS traction control light	
212	08	80	80	80	80	80	80	80	80	turned off	
										Steering angle light	
240	08	40	40	40	40	40	40	40	40	turned off	

Figure 4

Since one of the main aims of this application is to allow the general public and regular car enthusiasts to be able to use but because of time constraints I was not able to make the application as user friendly as I would of liked and not able to implement some of the ideas that were first discussed but to combat this I have created a separate application screen which contains some of the basic messages which can then be used to demonstrate and use the application and the send message feature to be able to interact with your vehicles CAN BUS network.

UML Diagrams

UML diagrams are an important to part of the design process because They bring a new way at looking at a project while keeping the simplicity and understanding. A UML Diagram can allow an everyday individual with minimal experience with software application design to understand the inner workings of the application and how each aspect of that application interact as well as how the user will interact with the application.

USE CASE

The main aim of a use case diagram and specifically this use case is to simply show how the user and CAN BUS network would interact with the application on a higher level. As you can see in this diagram the user interacts directly with the features of the application and shows an easy-to-read diagram and what features I aim to integrate as base features of the application. As we talked about with the front-end design, I have three main features which give the user the ability to interact and change values within a vehicles instrument cluster.

While with the CAN BUS Network for simplicity of the diagram the can bus network will not directly communicate with the application not without the help of specialty software that is designed to transfer the messages being sent from the application to the CAN BUS Network.

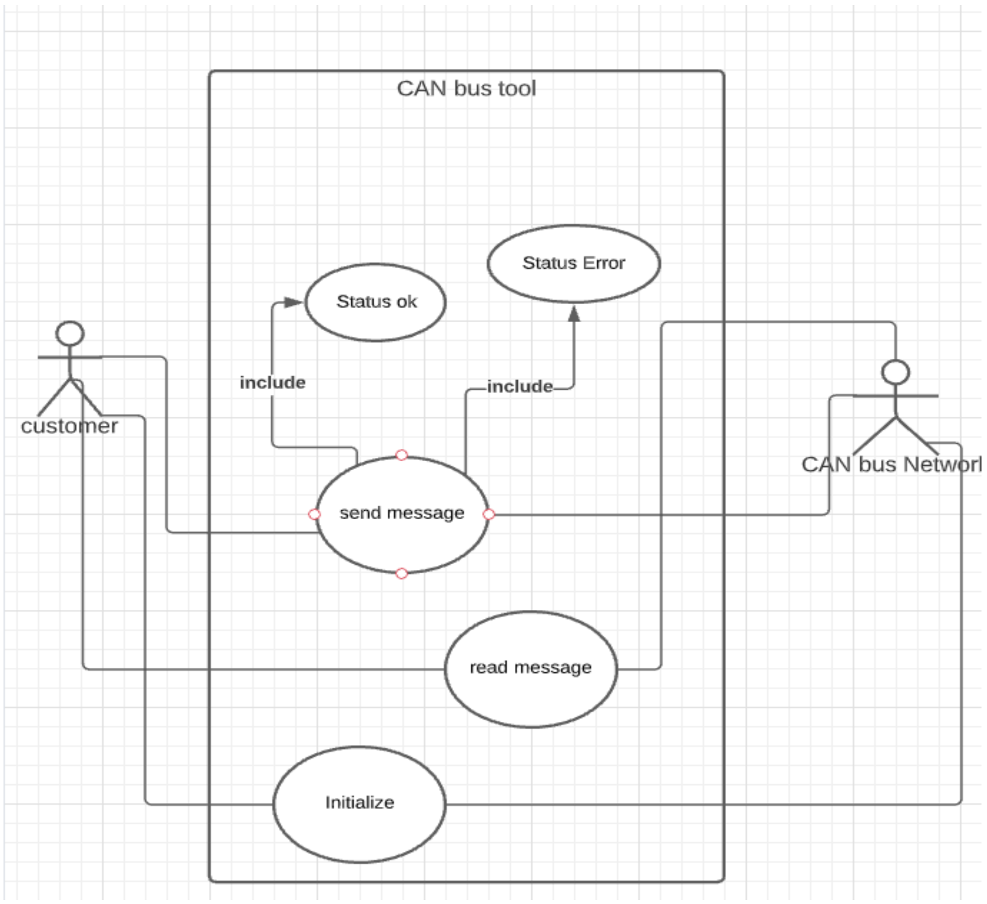


Figure 5

SEQUENCE DIAGRAM

The purpose of a sequence diagram is to show in a visually appealing and an easily understandable way for the person who wants to understand how the applications processes and individual elements interact and to easily demonstrate from start to finish how a process within the application is done and how each individual element of the application deals with that process.

This sequence diagram is one that shows all aspects of the project that will come together to allow the application to work correctly. To start off with this sequence diagram I will first explain the process in which the user goes through to confirm that the application can send and change the values being sent to the CAN BUS network this would be done before the user can enter the main screen which allows them to change multiple values.

First the user opens the application and is presented by a singular button which states that they will be able to check their connection to the network. Once the user clicks on the button the application will select a specific message that would change the value shown on the speedometer gauge on the real vehicle. This message is selected from and stored in a database of some sort which can be queried for that specific CAN BUS Network message which will be obtained before the application is created. With the message queried from the database the application then takes that message and then formats that message to be sent to the CAN BUS Network but as we spoke about earlier to assist in the transmission of that message the application is going to integrate with a pcan device which allows for the smooth transmission of CAN messages to be sent across the network and to integrate properly with the pcan device the application needs to use the previously designed API designed by the same company that creates the pcan device. With the pcan API it gives some essential features that will allow me to realize the features within this project and some of those features are being able to read and write messages to the pcan usb which would send those messages onto the vehicles network. Some of the other features include a initialize function which allows the user to send the bit rate and configure the pcan device. This feature could also be used to when the user first starts up the application to check the status of the application.

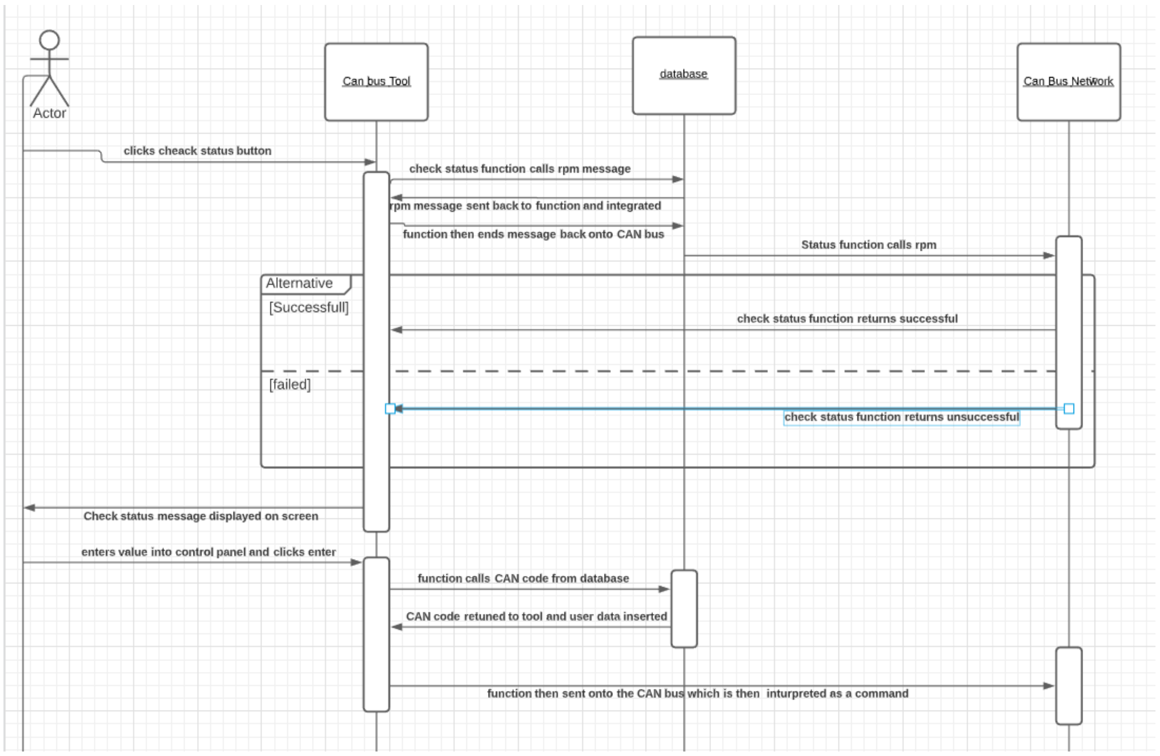

Actor
Can bus Tool
database
Can Bus Network
clicks cheack status button
check status function calls rpm message
rpm message sent back to function and integrated
function then ends message back onto CAN bus
Status function calls rpm
Alternative
[Successfull]
check status function returns successful
[failed]
check status function returns unsuccessful
Check status message displayed on screen
enters value into control panel and clicks enter
function calls CAN code from database
CAN code retuned to tool and user data inserted
function then sent onto the CAN bus which is then interpreted as a command

Figure 6

Conclusion

In conclusion I have talked about the main features of the application and how the user will interact with those features and I also spoke about the techniques that I will use to complete this application and to demonstrate the use of those techniques I chose to use uml and frontend diagrams which clearly shows how the application, and the tools would interact with each other.