Controller Area Network (CAN) – An Introduction

Richard T. McLaughlin – Warwick Control

Warwick Control Technologies has released two new CAN Tool Kit packages that allow an engineer/technician to have all the resources necessary to analyse, test and trouble-shoot a CAN bus system. As part of a series of informative articles to introduce Warwick Control's CAN tool kits, we bring you chapter 1, an introduction CAN.

This article briefly discusses the architecture of the CAN bus and intercommunication of information between ECUs in a vehicle. We then go on to explain the CAN frames that are generated by each ECU, and the information contained in them. This leads us to show you how the information is extracted from the CAN data frame and displayed on an analysis tool.

CAN bus Architecture

The growth of control systems in vehicle caused the growth of ECUs (Electronic Control Units) that require intercommunication. This intercommunication is achieved by networking technology known as CAN (Controller Area Network). Figure 1 illustrates typical intercommunication architecture in most vehicles today.

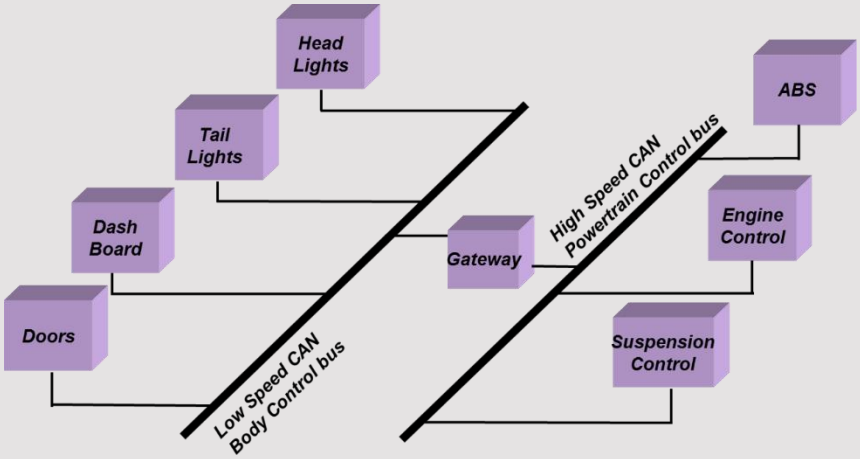

Figure 1: Typical CAN bus architecture

CAN is a contention based communication media, where each ECU must see that the bus is not busy before sending its CAN data frame across the bus. This utilises a procedure known as CSMA/CD (Carrier Sense Multiple Access, Collision Detect). Simply put, The ECU's share one wire, and must contend for the use of that wire to communicate with the other ECU's. As the name implies, the system is multiple access of the ECU's to the wire (or bus), and if an ECU wishes to transfer data onto the bus, it must sense if the bus is not in use by another ECU. If the bus is busy, then that ECU will hold the data until the bus is not busy, and then transmit.

The Collision Detect comes into play when two or more ECU's, at the exact same time, sense then transmit. In this scenario, the data from the different ECU's will collide and cause corruption of that data. CAN utilises a routine that senses this and allow the high priority information (e.g. wheel speed) to have throughput, while lower priority information (e.g. engine temp) will be slightly delayed. This is very similar

to computer networking technology used in our offices and homes, e.g. Ethernet. Ways of extracting the signals within a CAN data field is covered in a later article.

With this type of architecture, some information contending for the bus can be very low priority, not requiring time critical response. To avoid this data from cluttering up the contention arrangement for a time critical data bus, lower priority information bearing ECU's are isolated to a separate CAN bus and connected to the higher priority bus via a Gateway. The Gateway acts as a filter to control data passed from the Body Control CAN bus to the Powertrain Control CAN bus, and vice versa. In Figure 1, this is shown where the Powertrain Control bus is the high priority information carrier, and the Body Control bus is the low priority information carrier.

ECU CAN Interface

In today's vehicles, most of the ECU's are interconnected via the CAN bus. Figure 2 illustrates the structure of an ECU, where it contains a Microcontroller and a network interface in the form of the CAN Controller and the CAN Transceiver. The Microcontroller is the central controller that contains the embedded control program, such as Engine control or Suspension control. The CAN Controller acts as the network interface, where it extracts data from the Microcontroller to be transferred to other ECU's, and puts that data into frames to be transferred across the CAN bus. The CAN Transceiver sets up the electrical signalling to transfer data across the CAN bus. Connecting a CAN controller to a microcontroller is analogous to connecting an Ethernet interface to a PC.

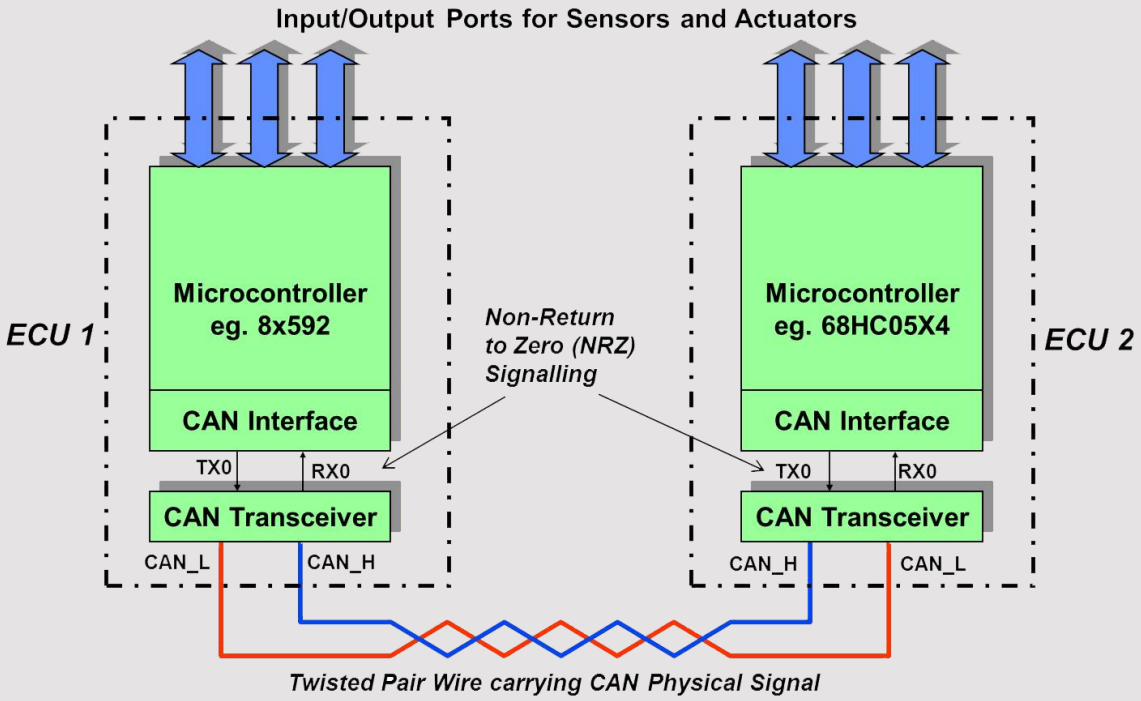

Figure 2: ECU CAN structure

The CAN Frame and its Data Field

When control data (e.g. engine speed) is needed for transfer to another ECU, the Microcontroller sends this data to the CAN Controller, where it puts it into a frame that identifies where the data is to go, as well as scales the data to information concerning its control function. Figure 3 shows a very basic structure of a CAN frame. Here, it can be seen that the header contains the Identifier and info on the size of the Data Field. The Identifier tells the rest of the system what the data is, e.g. Engine Parameters. The Data Field contains the actual parameters e.g. engine speed, temp, oil pressure. The Trailer contains error checking info and establishes the end of frame.

Figure 3: The CAN Frame

An example of this would be the Engine Control ECU needs to transfer engine speed to the Gearbox ECU to indicate the need for a gear change. In return, the Gearbox ECU needs to send shifting info to the Engine ECU to ensure smooth gear shifting. Each ECU will exchange CAN data frames via the CAN twisted pair bus.

As shown in Figure 2, typically the CAN bus is two wires twisted together to allow simple electrical noise immunity. When the CAN frame is to be transferred, the electrical signal transmitted from the CAN Transceiver is a dual differential digital burst of data that makes up the CAN frame. The electrical appearance of the CAN frame will be covered in a later article.

Details of the CAN Frame

Here we would like to show more detail of the CAN frame and illustrate how we can display this information on a CAN analysis tool. Figure 4 is a more detailed view of the CAN frame in figure 3.

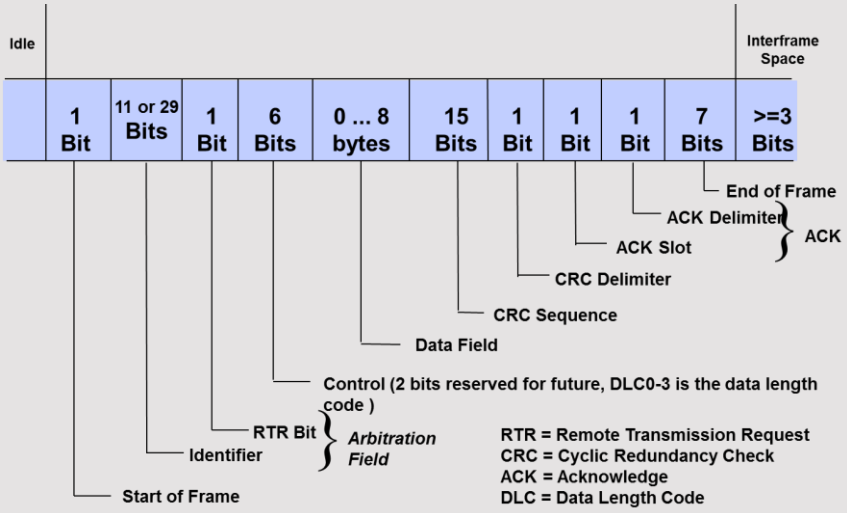

Figure 4: Details of the CAN Frame

Here you can see the main elements of the frame as:

- Start of Frame (SOF) – This indicates to all receivers that a CAN frame is on the way.
- Arbitration Field:
    - Identifier (ID) – ID sets the priority of a message. It indicates the function of the information in the data field, e.g. Engine parameters, Wheel speed, etc. There are 2 types of Identifiers – Standard (11 bit) and Extended (29 bit). Many cars use the 11-bit ID, but there are some vehicles that utilise the 29-bit ID. J1939 designates that all IDs are 29 bit.
    - Remote Transfer Request (RTR) – this bit is set when a node is requesting a particular ID from another ECU. Not often used in the automotive industries.
- Control Field, which contains the Data Length Code (DLC) – the DLC indicates the size of the Data Field – for 0 to 8 Bytes.
- Data Field – Data within the CAN frame, e.g. Engine speed, temperature, etc.
- Cyclic Redundancy Check (CRC) – Standard Error checking method. If check is good, move on to CRC Delimiter. If Check if mis-matched, and Error Frame is generated at this point.
- CRC Delimiter – provides space between CRC and the Acknowledge process.
- Acknowledge (ACK) Slot – bit inserted by all receivers to indicate a good frame has been received.
- ACK Delimiter - provides space between the Acknowledge process and EOF process.
- End of Frame (EOF) – 7 bits to indicate End of Frame before releasing the bus.

Note that the is a minimum Inter-Frame Space of 3 bits to allow space in the Idle before the next CAN node transmits onto the bus.

If we view a CAN frame with a two channel oscilloscope at the output of the CAN Transceiver in Figure 2, it will appear as shown in Figure 5. Here it can be seen that there are two physical signals – CAN High and CAN Low (CAN_H and CAN_L). CAN_H is in red, and CAN_L is in blue. This is known as differential signalling where we sense the difference between the two signals to determine a logic 1 or a logic 0.

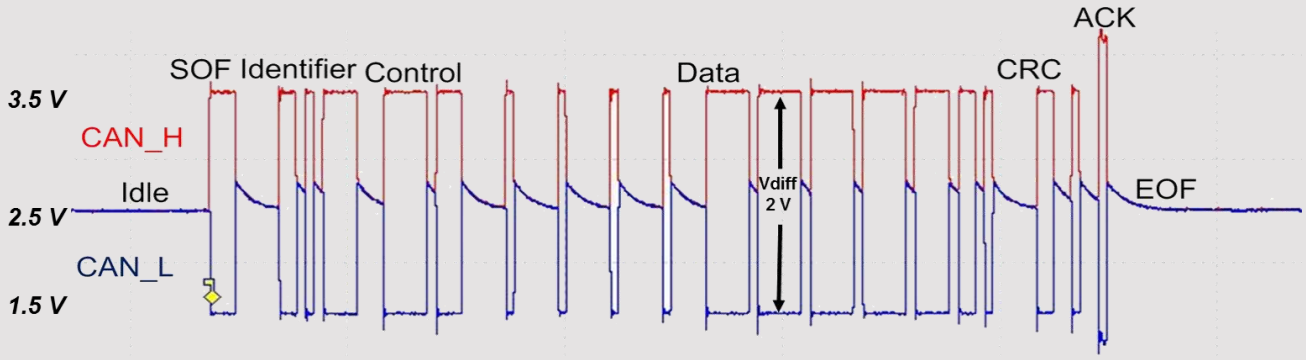

Figure 5. CAN Frame as displayed on an Oscilloscope

CAN Error Frame

As mentioned earlier, if the CRC process is mismatched, the CAN ECU will generate an Error Frame. An error frame is generated by sending 6 Dominant bits in a row. If a CAN ECU sees an error frame, it will break off its receive process and await the CAN frame to be re-sent.

To understand this, you will need to understand the process of bit stuffing. To allow for re-synchronisation, CAN utilises a bit stuffing process where if there are 5 bits or more in a row, the CAN controller will automatically insert an opposite polarity bit. Of course the receiver will expect this, and throw out the stuff bit during the receive process.

Because CAN uses the NRZ (Non-Return to Zero) physical signalling process, bit stuffing is required. This is covered more in a later article. Basically, Error signalling breaks the bit stuff rule.

Viewing CAN Frames on the X-Analyser 3

Utilising the information covered here, we shall have a look what a CAN frame on our Warwick Control analysis tool – X-Analyser. Here, we have connected to a simple CAN configuration on an early model Ford Fiesta. Referring to Figure 6, you can see the various parts of the CAN messages, and bus statistics are shown.

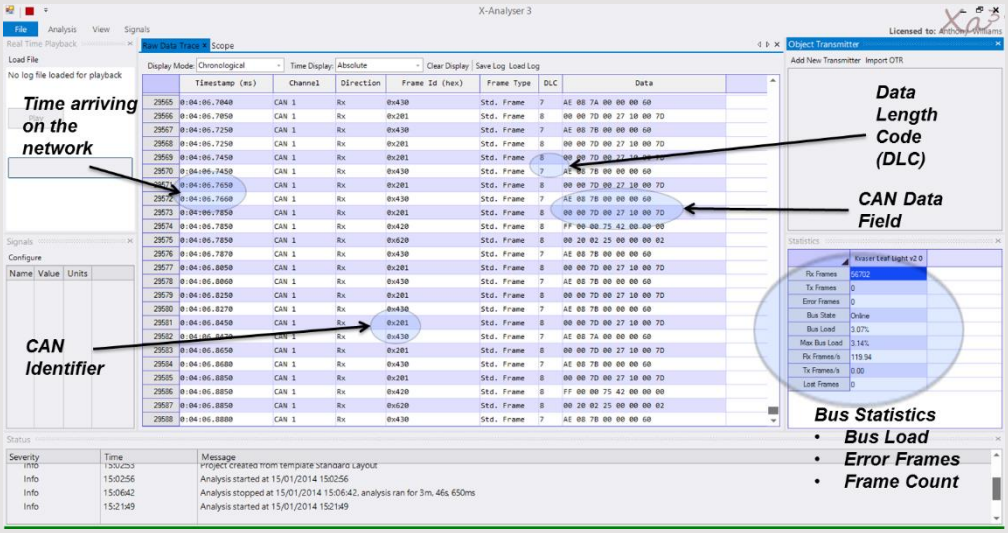

Figure 6: Displaying CAN Frames on Analysis/Test tool – X-Analyser

Notice that you can view the various CAN IDs, their data and their timestamps. Also notice that CAN frames use standard CAN IDs. The bus statistics show the Bus Load (frame rate), Frame count and if there are any Error Frames. This is a basic display of CAN messages with the hexadecimal view of their data. This information can be useful for seeing if there are any missing CAN IDs, if the bus load is normal, or if there are any error frames. Error frames are a good indication that the physical signalling on the CAN bus is noisy for various reason that will be covered in a later article.

Another way to view this display is set up the display mode as Fixed Position as opposed to Chronological mode. As can be seen in Figure 7, all the CAN IDs are viewed in a fixed position, and their time stamps are updated in real time. Of course all the CAN frames are collected chronologically in the background for later logging. Here you can view exactly how many CAN IDs there are on this particular system. Note that also on this display on the bottom left, there are some signals displayed. These are parameters extracted from the CAN data, and interpreted for display as real signals. We will discuss how this is accomplished in a later article.

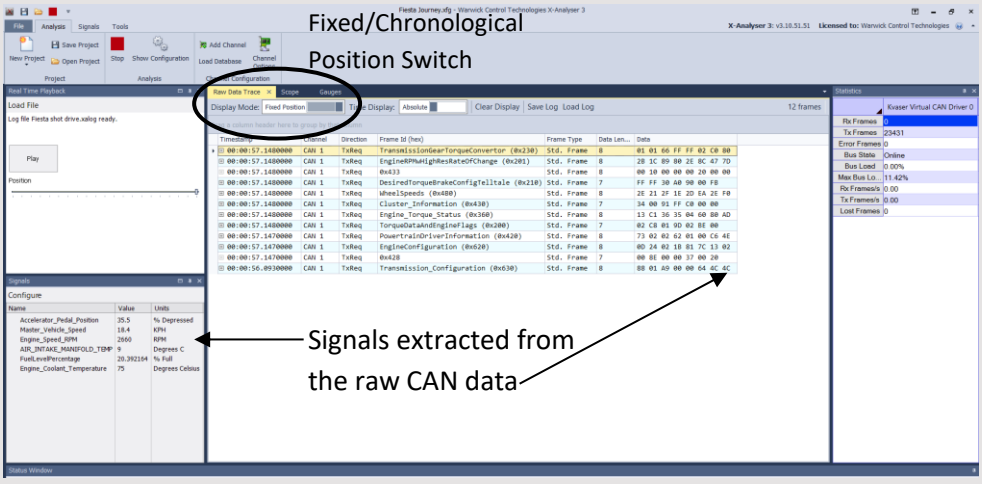

Figure 7. X-Analyser Fixed Position display

In summary, this article gives a very basic idea of the purpose of In-Vehicle networks in the form of the CAN technology, and how it transfers data throughout the vehicle. Analysis of the real time CAN data that flows between the ECUs can often improve the ability to diagnose vehicle electrical system problems. If the number of CAN IDs for a system is known, and an optimal bus load is ascertained, then the state of the CAN system can be basically determined. Bus statistics and the presence of Error Frame can give a quick view of the health of the CAN bus. The next article will cover how we extract the CAN data and display it as real signals, e.g. Engine speed, temp, wheel speeds, etc.