Matus Danko - Michal Taraba - Juraj Adamec - Peter Drgona*

VISUALIZATION OF SKODA INSTRUMENT CLUSTER

Most signals in modern vehicles which are transmitted from sensors to control units or actuators are transmitted via bus. The most common automotive data bus is Controller Area Network(CAN). Some of transmitted signals are speed of the vehicle, engine speed and coolant temperature which are used for visualization of Skoda Octavia instrumental cluster. For this visualization, CAN BUS communication is filtered and processed, to obtain three message objects to observe required data. For data acquisition from bus is used modular system CompactRio with module for CAN BUS from National Instruments. The whole application is created in visual programming language LabVIEW using FPGA module.

Keywords: *CAN, CAN BUS, Skoda instrument cluster, CompactRIO*

1. Introduction

At the end of 80´s German company Robert Bosh GmbH proposed data communication network called CAN (Controller Area Network). The original intention was transmission of information between sensors, control units and power components of the car and saving cabling. With development of electronics in vehicles arise requirement to divide systems of car to specific section by reliability and usefulness. Communication between these systems is necessary to monitor their function, setting of parameters, calibration and in first place, distribution of data which are no longer only simple binary or analog values.

2. CAN BUS

2.1 Layered structure

The object layer of CAN BUS is concerned with message filtering as well as status and message handling [1].

The transfer layer represents the kernel of the CAN protocol. It represents messages received to the object layer and accepts messages to be transmitted by the object layer. The transfer layer is responsible for bit timing and synchronization, message framing, arbitration, acknowledgement, error detection and signaling, and fault confinement [1].

The physical layer defines how signals are transmitted. The physical layer is not defined here, as it will vary according to the requirements of individual applications (for example, transmission medium and signal level implementations) [1].

2.2 Message routing

The content of a message is described by an identifier. The Identifier does not indicate the destination of the message, but describes the meaning of the data, so that all nodes in the network are able to decide by message filtering whether the data is to be acted upon by them or not. Within a CAN network, it is guaranteed that a message is accepted simultaneously either by all nodes or by no node. Thus, data consistency is a property of the system achieved by the concepts of multicast and by error handling [1].

2.3 Arbitration

Whenever the bus is free, any node may start to transmit a message. If two or more nodes start transmitting messages at the same time, the bus access conflict is resolved by bit-wise arbitration using the Identifier. The mechanism of arbitration guarantees that neither information nor time is lost. If a Data frame and a Remote frame with the same Identifier are initiated at the same time, the Data frame prevails over the Remote frame. During arbitration (Figure 1), every transmitter compares the level of the bit transmitted with the level that is monitored on

* **Matus Danko, Michal Taraba, Juraj Adamec, Peter Drgona**
Department of Mechatronics and Electronics, Faculty of Electrical Engineering, University of Zilina, Slovakia
E-mail: matus.danko@fel.uniza.sk

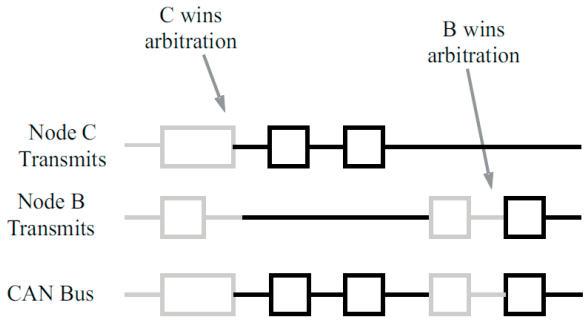

Figure 1 *Arbitration on a CAN Bus [2]*

Figure 2 *Standard data frame [2]*

Figure 3 *Extended data frame [2]*

the bus. If these levels are equal the node may continue to send. When a recessive level is sent, but a dominant level is monitored, the node has lost arbitration and must withdraw without sending any further bits [1].

3. Frame types of CAN BUS

Message transfer is manifested and controlled by four different frame types:

- A Data frame carries data from a transmitter to the receivers.
- A Remote frame is transmitted by a bus node to request the transmission of the Data frame with the same Identifier.
- An Error frame is transmitted by any node on detecting a bus error.
- An Overload frame is used to provide for an extra delay between the preceding and the succeeding Data or Remote frames [2].

3.1 Data frame

A Data frame is composed of seven different bit fields: Start of the frame, Arbitration field, Control field, Data field, CRC field, ACK field, End of the frame. The Data field can be of length zero [1].

The meanings of the bit fields of Figure 2 are:

- SOF - The single dominant start of frame (SOF) bit marks the start of a message, and is used to synchronize the nodes on a bus after being idle.
- Identifier-The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.
- RTR - The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way, all data being used in a system is uniform.
- IDE - A dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.
- r0 - Reserved bit (for possible use by future standard amendment).
- DLC - The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.
- Data - Up to 64 bits of application data may be transmitted.
- CRC - The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.
- ACK - Every node receiving an accurate message overwrites this recessive bit in the original message with a dominate bit, indicating an error-free message has been sent. Should a receiving node detect an error and leave this bit recessive, it discards the message and the sending node repeats the message after rearbitration. In this way, each node acknowledges (ACK) the integrity of its data. ACK is 2 bits, one is the acknowledgment bit and the second one is a delimiter.
- EOF - This end-of-frame (EOF), 7-bit field marks the end of a CAN frame (message) and disables bit-stuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is stuffed into the data.
- IFS - This 7-bit interframe space (IFS) contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area [2].

As shown in Figure 3, the Extended CAN message is the same as the Standard message with the addition of:

- SRR - The substitute remote request (SRR) bit replaces the RTR bit in the standard message location as a placeholder in the extended format.
- IDE - A recessive bit in the identifier extension (IDE) indicates that more identifier bits follow. The 18-bit extension follows IDE.
- r1 - Following the RTR and r0 bits, an additional reserve bit has been included ahead of the DLC bit [2].

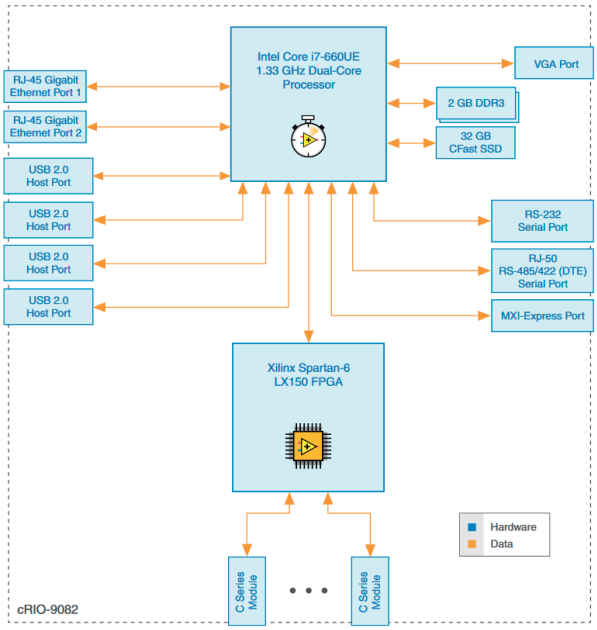

Figure 4 NI cRIO 9082 Hardware overview

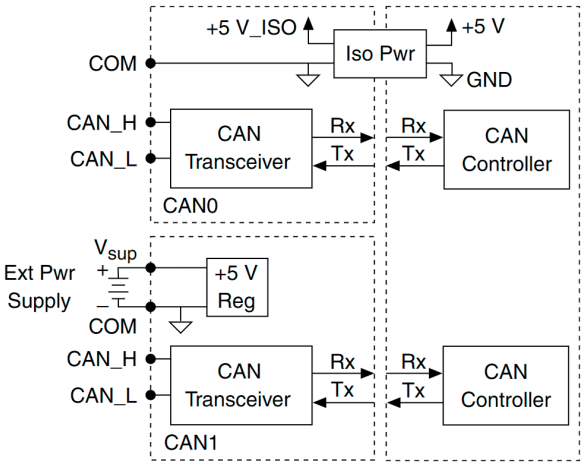

Figure 5 NI 9853 Hardware overview[5]

3.2 Remote frame

The intended purpose of the remote frame is to solicit the transmission of data from another node. The remote frame is similar to the data frame, with two important differences. First, this type of message is explicitly marked as a remote frame by a recessive RTR bit in the arbitration field, and secondly, there is no data [2].

3.3 Error frame

The error frame is a special message that violates the formatting rules of a CAN message. It is transmitted when a node detects an error in a message and causes all other nodes in the network to send an error frame as well. The original transmitter then automatically retransmits the message. An elaborate system of error counters in the CAN controller ensures that a node cannot tie up a bus by repeatedly transmitting error frames [2].

3.4 Overload frame

The overload frame is mentioned for completeness. It is similar to the error frame with regard to the format, and it is transmitted by a node that becomes too busy. It is primarily used to provide an extra delay between messages [2].

4. Used hardware

4.1 NI cRio 9082

The high-performance multicore NI cRIO-9082, block schematic shown on Figure 4, provides advanced Intel Core i7 dual-core processing, built-in VGA display output for an integrated user interface, and the option to use a Microsoft Windows Embedded Standard 7 (WES7) or LabVIEW Real-Time OS. The increased processing power of the cRIO-9082 makes it well suited to perform the advanced processing tasks required by complex applications such as machine vision and rapid control prototyping. Choose LabVIEW Real-Time to take advantage of deterministic execution and the highest degree of reliability in continuous operation environments. The high-performance multicore cRIO-9082 also offers the widest array of connectivity and expansion options available in the CompactRIO platform, including the high-bandwidth and low-latency MXI-Express bus for expansion using the 14-slot MXI-Express RIO chassis [3, 4].

4.2 NI 9853

The NI 9853 has two 9-pin male D-Sub connectors that provide connections to a CAN bus. Each port on the NI 9853 has pins for CAN_H and CAN_L, to which you connect the CAN bus signals. Each port has two common pins (COM) that are internally connected to the module's isolated reference and serve as the reference ground for CAN_H and CAN_L. You can connect the CAN bus reference ground (sometimes referred to as CAN_V-) to one or both COM pins. The port also has an

Table 1 Meaning of LED

LED	SD Card Slot	Description
Present (Card 0), Green	SD Card Slot 0	This LED is lit when an SD Card is in Slot 0 and the SD Card slot door is closed.
Busy (Card 0), Yellow	SD Card Slot 0	This LED is lit when the card in Slot 0 is active. This LED is flashing when the NI 9802 is performing I/O on the card in Slot 0. Do not remove the SD Card while this LED is lit or flashing.
Present (Card 1), Green	SD Card Slot 1	This LED is lit when an SD Card is in Slot 1 and the SD Card slot door is closed.
Busy (Card 1), Yellow	SD Card Slot 1	This LED is lit when the card in Slot 1 is active. This LED is flashing when the NI 9802 is performing I/O on the card in Slot 1. Do not remove the SD Card while this LED is lit or flashing.

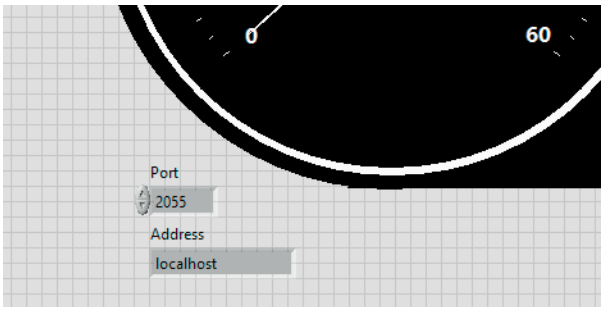

Figure 6 *Visualization of the cluster by TCP with extra control*

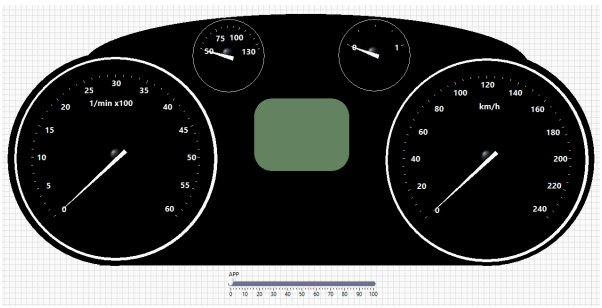

Figure 7 *Visualization of Skoda instrument cluster*

optional shield pin, SHLD, that you can connect to a shielded CAN cable. Connecting SHLD may improve signal integrity and EMC performance in a noisy environment [5].

CAN0 of the NI 9853 is internally powered (Figure 5), and therefore requires no external power supply. CAN1 requires an external power supply of +8 to +25 V to operate. Supply power to the V_{SUP} pin of CAN1 from the CAN bus [5].

4.3 NI 9802

The NI 9802 is an SD Card storage module that can read from and write to SD Cards. The NI 9802 has two SD Card slots into which you can insert NI-approved SD Cards. The card slots each have a door to protect the SD Cards while in the module. Each door is connected to a switch that indicates to the NI 9802 software when it is safe to read from or write to the SD Card. The NI 9802 has four LEDs that indicate which slots contain an SD Card and if the SD Card is active. Refer to Table 1 for descriptions of each of the LEDs on the NI 9802 and when it is safe to remove SD Cards from the NI 9802 [6].

5. CAN BUS analysis

The whole application is created in visual programming language LabVIEW using FPGA module. The project has four programs, first is for data acquisition from CAN BUS. At start of the program, it is necessary to set up delay for transmitter and receiver initialization and CAN BUS speed for each port. CAN BUS speed can be set in two different ways. The first way is by setting the speed in the configuration of NI 9853. The second way is by setting of the speed through constant for bit timing register. In the second program, received data from CAN BUS are saved as local variables for next processing. The last two programs processing local variables for visualization and transferring data by TCP protocol on other devices such as touch panel. Visualization on other devices through TCP/IP has two extra controls (Figure 6), IP address of the source and port [7-9].

For visualization of instrument cluster from Skoda vehicle (Figure 7) three messages of CAN communication are filtered, which are further processed. Compared to the physical cluster our visualization of the cluster does not display mileage and fuel gauge because these signals are not sent by CAN BUS.

The first CAN message contains data for speedometer and accelerator pedal position. As mentioned above, CAN BUS can transmit only eight-bit number, this means 255. If there is needed a value greater than this number, it is necessary to use more than one eight-bit number. For RPM, the second byte is used as lower eight bits and the third byte as upper eight bits. Real RPM is the result of division by four after joined of these upper and lower eight bits. The seventh byte is used for displaying accelerator position in percent.

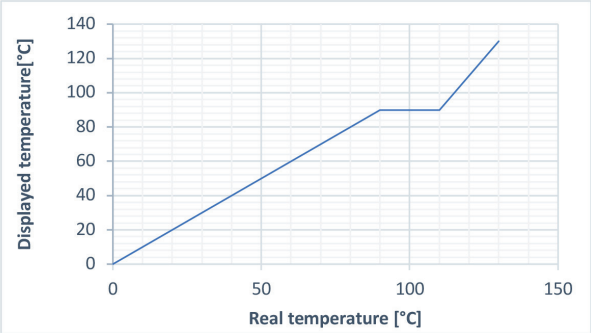

Figure 8 *The non-linearized curve of the temperature gauge*

Third CAN message is needed to get the speed of the vehicle. For data of RPM, two bytes, second and third are used. After joining, the result is divided by 100.

The first bit of the second CAN message contains information about the temperature of the coolant. To get the right range of temperature, division by ¾ and subtraction 48 is necessary, due to the fact, that the temperature of coolant can be below zero. Next step in data processing is nonlinearization of temperature gauge as a physical gauge so we can't see hysteresis of the thermostat.

Temperature displayed on gauge up to 90 °C is real, range from 90 °C to 110 °C, which is operating temperature of the engine is displayed as 90 °C, from 110 °C is temperature curve nonlinearized as shown in Figure 8.

6. Conclusion

The main purpose of this paper was to design highly sophisticated CAN BUS analyzer using powerful CompactRIO with dual-core processor and FPGA, a module for CAN BUS and module for SD Cards for data logging. Data from CAN BUS are acquired every 10 or 20ms and processed for visualization of Skoda Octavia II instrument cluster. RPM, the speed of the vehicle, coolant temperature and positions of accelerator pedal are processed data, yet. Data processing for other indicators are possible except fuel gauge because the signal of fuel level is transferred analog.

Acknowledgment

Results of this work are supported by grant APVV-15-0571 and VEGA 1/0928/15.

References

[1] Freescale Semiconductor, Inc., Bosch Controller Area Network (CAN) Version 2.0.
[2] Texas isntrument, Introduction to the Controller Area Network (CAN), 2008.
[3] National instruments, Getting started guide NI cRIO-9082, 2016.
[4] National instruments, User manual NI cRIO-9082, 2016.
[5] National instruments, Getting started guide NI 9853, 2015.
[6] National instruments, Operating instruction and specification NI 9802, 2009.
[7] KONIAR, D., HARGAS, L., SIMONOVA, A., HRIANKA, M., LONCOVA, Z.: Virtual Instrumentation for Visual Inspection in Mechatronic Applications, Procedia Engineering (Modelling of Mechanical and Mechatronic Systems), 96, 227 – 234, 2014.
[8] SPANIK, P., HARGAS, L., HRIANKA, M., KOZEHUBA, I.: Application of Virtual Instrumentation LabVIEW for Power Electronic System Analysis. Proceedings of the 12th International Power Electronics and Motion Control Conference (EPE-PEMC 2006), Slovenia, 1699-1702, 2006.
[9] HARGAS, L., KONIAR, D., STOFAN, S.: Sophisticated Biomedical Tissue Measurement Using Image Analysis and Virtual Instrumentation, LabVIEW Practical Applications and Solutions. Chapt. 8. Ed: FOLEA, S. InTech, Rijeka, p. 155-180, 2011.