Modeling of Inter-modular Interaction Based on the CANopen Protocol in Vibration Monitoring Systems

D A Plotnikov, V I Lachin, K Y Solomentsev
Chair of «Automatic and telemechanics» Platov South-Russian State Polytechnic University (NPI), 132 Prosveshenia Str., Novocherkask, Rostov Region, 346428, Russian Federation

E-mail: dpl68@mail.ru

Abstract. This paper concerns the modeling of communication processes in modular information-measuring and control systems applying the CANopen protocol. Brief information about the format of CAN interface messages is represented and the method for estimating the messages delivery time is proposed and justified. The basic transmission mechanisms for data used by the CANopen protocol are described, and the principles of constructing a program-logic model that allows analysing the processes of inter-modular interaction based on this protocol are formulated. Implementation of the model for a turbine generator vibration monitoring system is exemplified and simulation results are presented.

1. Introduction

Vibration monitoring and diagnostics are the most universal methods for preventing accidents and determining the technical condition of rotary machines. In this connection turbine units of power plants are equipped with stationary vibration monitoring systems (VMS). The number of vibration measurement points and associated parameters can vary significantly for units of different power and design, therefore, to simplify adaptation to specific technological equipment the VMSs usually have a modular structure where the amount of measuring modules depends on the complexity of the monitored object.

Measured vibration parameters are transmitted by modules to the higher levels of VMS. Furthermore, the modules must interact in order to detect certain equipment operation modes. Regardless of the interfaces and protocols used for such interaction, the design of VMS involves estimating the data network bandwidth and messages delivery time. The possibility of connecting additional modules or increasing the inputs polling frequency in order to obtain additional diagnostic information will depend on the successful implementation of this problem. Errors at this stage of development can lead either to the total inoperability of VMS or to the need to reduce the amount of data transferred, which will negatively affect the system’s functionality. Therefore, the task to develop a methodology and tools for estimating the messages delivery time and the bandwidth of a data network used for inter-module communication becomes actual. Onwards the issues related to solving this problem for VMS using the CAN interface and the CANopen protocol will be considered.
2. Basic information about the CAN interface

The CAN interface (Controller Area Network) [1] was developed by Bosch for vehicle units control systems. It is a serial high-speed and highly reliable interface for data transmission in a broadcast mode and multi-master environment. A good combination of low cost connection, simplicity and reliability with available diverse element base and development tools resulted in the wide spread of CAN technology not only in the automotive industry but in industrial automation systems as well. The initially provided possibility to operate in the conditions of strong electromagnetic interference makes this interface convenient to use in electric power plants, including organization of inter-module interaction in VMS. The CAN parameters, fixed in Bosch specification [1] and in the international standard ISO 11898 [2], correspond to the first two levels (physical and data link ones) of the seven-level OSI model [3].

Devices, referred to as nodes in the CAN terminology, interact via transmission of messages, which are also called frames. Depending on the initiator of the transmission and its purpose, there are four types of frames defined as Data Frame, Remote Frame, Error Frame and Overload Frame. For data transmitting the Data Frame is used, owing to which this frame type is most commonly applied. Logical «zero» and «one» levels are encoded, respectively, by the dominant and recessive communication line state. With simultaneous attempts of several transmitters to put the line into different states, the priority is given to the dominant state.

Data Frame structure is shown on figure 1.

![Figure 1. Formats of Data Frame.](image_url)

(a) – standard format

(b) – extended format

The Arbitration Field of the frame includes an identifier ID that uniquely identifies the content and priority of the message, as well as the service bits RTR, SRR and IDE. The CAN 2.0A specification describes a standard message format that provides an 11-bit identifier (figure 1a), allowing to distinguish up to 2048 message types. In the CAN 2.0B specification the length of the identifier is increased to 29 (figure 1b): 18 additional bits are added to the standard 11 ones; with the number of possible message types increased to 536 million. The RTR bit (Remote Transmission Request) – always has a dominant level for the standard format Data Frame. In the extended frame format, the recessive SRS (Substitute Remote Request) bit replaces the following one (in the standard format) after the 11-bit identifier of the RTR bit. The format recognition bit IDE (ID Extension) has a dominant level for the standard frame format and a recessive level for the extended format. The Control Field contains a 4-bit Data Length Code (DLC) specifying the length of the data field (0…8 bytes) and two reserved bits r0 and r1. The CRC Field includes the 15-bit check sequence for the message followed by the CRC Delimiter (one recessive bit). In the Acknowledgment (ACK) field, the node that transmits the data always sets the recessive level. In case of successful transmission, each receiving node signals this by setting dominant level in the ACK field. At the end of the Data Frame a completing sequence of seven recessive bits is transmitted.

In the CAN network, all nodes are equivalent: each of them can initiate transmission. A non-destructive arbitration procedure is provided to avoid bus access conflicts, ensuring the priority transfer of more important messages.
3. Estimation of CAN message delivery time

Obviously, the time of message delivery from one module to another is an essential factor determining the consistency of the VMS modules functioning. With the simultaneous interaction of multiple modules, the evaluation of this time is a non-trivial task. Let’s consider the factors influencing the timely delivery of messages over a network with a CAN interface. The time of data transfer from one node to another includes the following parameters:

- message preparation time $T_{\text{PREP}}$ (includes all operations for generating the message code and its parameters as well as for node transmitter setup);
- waiting time for bus access to send a message $T_{\text{WAIT}}$;
- message transmission time $T_{\text{TX}}$.

The total message delivery time is given by:

$$T = T_{\text{PREP}} + T_{\text{WAIT}} + T_{\text{TX}}$$ (1)

The message preparation time $T_{\text{PREP}}$ depends significantly on the modules performance and the complexity of the task performed on the upper levels of the OSI model [3], and is not considered in detail in this paper. The rest components of equation (1) are determined by the CAN operation logic and message formats.

To estimate the message transmission time $T_{\text{TX}}$ the format of the Data Frame intended for data exchange should be analyzed.

Figure 1 shows that all message fields, except the data field, have a fixed length (if the length is not specified, it means 1 bit). The frame format bit IDE (ID Extension) has a dominant level of “0” for the standard frames and a recessive “1” for the extended frames. The DLC parameter specifies the length of the data field in the range 0…8 bytes. On this basis, the expression for calculating the number of bits $N$ is written as:

$$N = 44 + 20 \cdot \text{IDE} + 8 \cdot \text{DLC}$$ (2)

Thus, the total length of the standard Data Frame varies according to the size of the data field in the range of 44 to 108 bits, and the length of the extended Data Frame is 64 to 128 bits.

The bit transmission time $T_0$ is a fixed value for all nodes of the CAN network and it is determined by the selected baud rate. Eg.: for a 1 Mbps baud rate, $T_0 = 1 \mu s$. However, even if the number of bits $N$ in the message and the bit time $T_0$ are known, it is impossible to calculate the transmission time of the entire message precisely. This is due to the use of bit stuffing technology, the essence of which lies in the following: if the transmitter finds five consecutive bits with the same values in the flow of transmitted data it automatically inserts an additional bit with an opposite value into the flow. A receiver accepting the data removes additional bits automatically. The described procedure is performed to prevent a long-term unchanged dominant or recessive state of the bus, as this would impede to synchronize receivers with the data flow.

Bit stuffing does not affect the result of data exchange, but it may increase the message transmission time. The number of additional bits will depend on the content of the particular message, and therefore, in general, is a random value.

In the best case, the number of additional bits is zero. Then the transmission time of standard and extended Data Frames $T_{\text{MSG\_MIN}}$ will be defined as follows, taking into account the equation (2):

$$T_{\text{MSG\_MIN}} = T_0 \cdot (44 + 20 \cdot \text{IDE} + 8 \cdot \text{DLC})$$

Consider next the worst case for standard Data Frames. If the "11 bit ID" field is 0x000 or 0x7FF, then twelve bits of the same polarity are transmitted at the beginning of the message: SOF and "11 bit ID" or "11 bit ID" and RTR, respectively. Due to this, two additional bits are added to the message; then changing the level from “1” to “0” between the RTR and IDE fields zeroes the counter of consecutive same-polarity bits.
The maximum number of additional bits $N_{DS}$ in the IDE, $r_0$, and Data fields will be added when transferring “zeroes” (for $DLC = 3$ – “ones”); it can be calculated by the equation (3), in which the square brackets denote the operation of finding the maximum integer not exceeding a given value:

$$N_{DS} = \lceil 8 \cdot DLC / 5 \rceil + A,$$

where $A = 1$ for $DLC \in \{0, 1, 3, 8\}$; otherwise $A = 0$.

The value of the CRC field is determined by the contents of the previous fields, however, the dependence of the amount of consecutive identical bits in this field on the previously transmitted values is quite complex and its identification to the authors’ opinion is inexpedient. Instead, we assume that a maximum of three additional bits can be added to the CRC field (in cases where its value is 0x0000 or 0x7FF). Additional bits are not added to the remaining message fields.

Thus, the maximum number of additional bits $N_{BS}$ for standard Data Frame can be calculated by the formula:

$$N_{BS} = 5 + \lceil 8 \cdot DLC / 5 \rceil + A.$$

Therefore, the maximum transmission time for a standard Data Frame $T_{MSG\_S\_MAX}$, taking into account equations (2) and (4), can be defined as:

$$T_{MSG\_S\_MAX} = T_0 \cdot (49 + 8 \cdot DLC + \lceil 8 \cdot DLC / 5 \rceil + A).$$

Let us analyze the worst case for extended Data Frames. If the "11 bit ID" and "18 bit ID" fields are 0x7FF and 0x3FFF respectively, so thirty bits with a value of 1 are transmitted at the beginning of the message: "11 bit ID", SRR, IDE and seventeen high bits of the "18 bit ID" field. Due to this, six additional bits will be added to the message. The maximum number of additional bits $N_{DE}$ in the fields RTR, $r_0$, and "Data", taking into account the zero value in the low-order bit of the "18 bit ID" field, can be calculated by the formula:

$$N_{DE} = \lceil 8 \cdot DLC / 5 \rceil + 1 + B,$$

where $B = 1$ for $DLC = 3$; otherwise $B = 0$.

It was demonstrated earlier that three additional bits can be added to the CRC field. Therefore, for extended Data Frames the maximum number of additional bits $N_{BE}$ is given by:

$$N_{BE} = 10 + \lceil 8 \cdot DLC / 5 \rceil + B.$$

The maximum transmission time $T_{MSG\_E\_MAX}$ for the extended Data Frame, with equations (2) and (6) taken into account, can be determined as:

$$T_{MSG\_E\_MAX} = T_0 \cdot (74 + 8 \cdot DLC + \lceil 8 \cdot DLC / 5 \rceil + B).$$

Thus, the minimum value of the $T_{TX}$ parameter corresponds to the value of $T_{MSG\_MIN}$, and the maximum value corresponds to $T_{MSG\_S\_MAX}$ for standard Data Frames and $T_{MSG\_E\_MAX}$ for extended Data Frames.

The access waiting time for the communication channel $T_{WAIT}$ is determined by priorities and the number of other messages waiting to be transmitted, as well as by the channel status (free or busy) at the moment of the transfer request. Since the CAN interface has a bus architecture, only one message can be transmitted at a time. High-priority messages are transmitted first, afterwards, less priority ones. Priority assignment are settled at the upper levels of the OSI model, which are not described by the CAN specification, and fall beyond the scope of this paper. Some recommendations for estimating the message delivery time, with their priorities taken into account, are given in [4].

Let's analyze the access waiting time for the most priority message. If the communication channel is free at the request time for its transmission, the transfer starts immediately and the waiting time is zero. The communication channel being busy, the request for transmission will be processed only after the channel is free irrespective of the priority of the waiting message. The maximum duration of the busy status of the communication channel is equal to the maximum transmission time of the message.
(equations (5), (7)), increased by the minimum interval between messages (interframe space), which, according to [1], is $3T_0$. Therefore, the maximum access waiting time for the communication channel for the highest priority message of the standard format is given by:

$$T_{\text{WAIT}_S_{\text{MAX}}} = T_0 \cdot (52 + 8 \cdot \text{DLC}_\text{MAX} + [8 \cdot \text{DLC}_\text{MAX} / 5] + A),$$

where $\text{DLC}_\text{MAX}$ is the maximum number of bytes in the data field of the transmitted messages.

For extended format messages

$$T_{\text{WAIT}_E_{\text{MAX}}} = T_0 \cdot (77 + 8 \cdot \text{DLC}_\text{MAX} + [8 \cdot \text{DLC}_\text{MAX} / 5] + B).$$

Thus, the minimum value for the $T_{\text{WAIT}}$ parameter is zero, and the maximum value is equal to $T_{\text{WAIT}_S_{\text{MAX}}}$ for standard format messages and $T_{\text{WAIT}_E_{\text{MAX}}}$ for extended format messages.

These expressions demonstrate that to reduce the average message delivery time, one should make efforts to reduce the message preparation time $T_{\text{PREP}}$, as well as the average length of transmitted messages. This can be achieved in the following ways.

1) Prepare requests and responses in advance. If the number of different requests is small, they can be generated when the application is initialized and stored in special sections of the CAN controller's memory - in the so-called message objects (MO). The controller C_CAN [5], currently widely used in single-chip microcontrollers (MCU), provides operation with thirty-two MO. In this case, the request preparation for transmission in the operating mode is reduced to writing a command "Transfer of MO" to the corresponding control register. Preparation of responses can be performed similarly: transmission data are recorded in MO immediately after their modification (and only in this case), and when the request is received, the application generates the "Transmission" command for the corresponding MO. The described approach was used by the authors in developing the modular event recorder based on MCU LPC11C24 [6] and allowed to reduce the message preparation time $T_{\text{PREP}}$ to several dozen of nanoseconds.

2) Data transmission either complete or partial can be performed not in the data field but in the message identifier field with the data field size reduced to the minimum required value – to zero in the limit. This is possible when a minor number of messages types are required: then one part of the identifier field is used to indicate the type of the message, and the other part for the data transmission. Examples of data completely transmitted in the identifier field can be status attributes (serviceable / defective, ready / not ready), the number of the requested module, the indication of the requested information, etc.

3) Data transmission with long bit sequences of the same value should be avoided to prevent insertion of bits. From equations (4) and (6) it follows that due to such sequences up to eighteen additional bits can be added to standard Data Frames, which comprises 16.6% of the nominal message size. Up to twenty-two bits (17.2%) are added to the extended Data Frames. Accordingly, the messages transmission time also increases. If the described sequences are likely to occur with high probability, then it is advisable to modify the data before transmission (for example, to add by module "2" with the 0x55 number), and after reception to restore them. The duration of operations for modifying and restoring data on modern MCUs is typically much shorter than the transmission time of additional bits.

4. CANopen protocol

The CAN interface described above provides only two lower layers of the OSI model: physical and data link layers. To solve simple communication problems this may be sufficient, however, in systems consisting of dozens of nodes and exchanging hundreds of parameters the use of higher-level protocols solving addressing problems, error correction, synchronization, and some others is more appropriate. One of such protocols is CANopen [7] – a high-level communication protocol based on the CAN interface, as well as a set of specifications for interacting devices. The protocol was developed for using in embedded systems, such as vehicle on-board devices, industrial equipment, etc. The CANopen specification describes common approaches to the setting up interactions, device logical
models, interface definitions and implementation details for various typical applications. CANopen standardizes communication between devices and applications from different manufacturers. It is used in a wide variety of industries, particularly in industrial automation [8].

The CANopen protocol provides the top five levels of the OSI model: network level (addressing, routing), transport level (message delivery from a source to a destination without losses, unchanged and in the correct order), session level (interaction synchronization), presentation level (data encoding) and application level. The top-most one defines how to configure devices and transfer application-level data objects.

To estimate the messages delivery time and the network load, the main functions of the CANopen application level are to be considered. Their entire description is given in [9].

4.1. The object dictionary

One of the basic concepts of CANopen is an object dictionary, which represents a table containing configuration parameters and current device or process data. Each dictionary entry (object) has a 16-bit index; within one object up to 256 sub-objects can be defined, which are addressed to by using an eight-bit sub-index. To simplify recording the dictionary objects will be designated as "XXXX-YY", where "XXXX" is the object index, and "YY" is the sub-index. The standard specifies that the objects with some indices or indices ranges have a drastically defined assignment, while the assignment of other objects can be arbitrary. Some predefined dictionary objects, such as the device type (1000h), are mandatory and must be implemented, while the others (e.g. the 100Ah – version of the software) are optional. For the device to conform with CANopen specification, its object dictionary must contain at least all mandatory objects.

Devices in the CANopen network interact with each other through by referring to object dictionary of their counterparty. For example, one of the devices records the value "True" in a certain object of the user part of another device’s dictionary. Another device can interpret this as an enabling signal to receive data from the corresponding analog input. And vice versa: when reading information from the object dictionary, we can get the accumulated data or information about the current state of the device. Two services are used to access the dictionary objects: Service Data Object (SDO) and Process Data Objects (PDO).

The CANopen specification defines the basic types of object dictionary data: boolean, unsigned and signed integer, floating point, and character. More complex data types, such as strings or timestamps, can be generated on the basis of basic types.

4.2. Message format

The format of CANopen messages is based on the CAN Data Frame format (figure 1). The control bits and the CAN identifier field form the so-called message identifier COB ID. The 11-bit CAN identifier is divided into two fields: a four-bit function code and a seven-bit node identifier Node ID. Due to the Node ID bit capacity the total number of nodes in the CANopen network is limited to 127.

4.3. Synchronization

In order to synchronize the nodes of the system, one of them can periodically generate a special message – a synchronization object (SYNC object). The remaining nodes in response to the SYNC object can transmit any data or perform any other operations that require a time reference.

4.4. SDO Service

The CANopen protocol specifies that each node on the network must be a server that processes requests for writing and reading its object dictionary. This allows the master device to function as a client relative to the other nodes. The mechanism of direct access to the object dictionary of the server is realized through the SDO service. A node whose object dictionary is addressed to is called an SDO server, and the node that executes this access is called the SDO client. Interaction always begins on the initiative of the SDO client. At each access to the server, only one dictionary object can be written or read.

Due to the above features, the SDO service has a poor performance and is intended primarily for configuring nodes, as well as for recording and reading large amounts of data (e. g. for firmware
updates). It is not used in the operating mode of the system and in this connection not considered further.

4.5. PDO Service

The process data in CANopen are the data that can change over time, such as input and output signals, current hardware status, etc. These data, like the entire information about the CANopen device, are stored in the object dictionary and, therefore, accessible by using the SDO service. However, since SDO allows only one dictionary object to be accessed in a single transaction, the periodic transmission of information about the set of changing parameters will cause a significant increase in the load on the data network. In addition, the SDO service does not allow the device to transmit any data on its own initiative, for example, when they are changed or periodically, this requires information to be received in the polling mode only. In this connection, to transfer process data another method is used: the PDO service.

There are two types of PDOs: Transmit-PDOs (TPDOs) and Receive-PDOs (RPDOs). TPDO is the data sent by a node, and RPDO is the data coming into the node. For PDO of any type, there are two groups of parameters: communication parameters and mapping parameters. For these parameters a range of indexes 1400h … 1BFFh is reserved in the object dictionary.

The communication parameters for each PDO allow to specify the message identifier COB ID, a transmission method, the post-transmission inhibit time (for TPDO), and the transmission rate. There are several methods to initiate a PDO transmission: by event, by timer, by request and by synchronization event. This method applied for each PDO is specified by the communication parameters. Mapping parameters provide index, sub-index and length of dictionary objects to be transmitted or received in each PDO message. This promotes flexible configuration of rate and priority of the process data transmission, depending on their importance.

4.6. Error control services

The CANopen specification defines two services for the network nodes to inform about their availability and serviceability: guarding and heartbeats, the latter being preferred.

The heartbeats service suggests that the monitored nodes periodically send special messages informing that the node is on the network and serviceable. The network master or any other designated device verifies the presence of such messages. In the absence of messages from any node for a given time, a conclusion is made about its malfunction and then appropriate action is taken (attempts to restart the faulty node, informing the operator, record in the system log, etc.).

When using the guarding service, the network master periodically requests information from the monitored nodes about their status. If the node does not respond within the specified time, then a conclusion is made about its malfunction. This service creates an increased load on the network, as a result of which it is not recommended for use.

5. Modeling of communication processes in VMS

Using the technique described in Section 3, the authors have developed a program-logic model that allowing to study the processes of inter-modular interaction in VMS and other systems based on the CANopen protocol.

By its structure and functioning in the part of inter-modular interaction this model is identical to the real VMS and represents a set of blocks corresponding to the system modules. This provides the input of the initial data corresponding to the setting parameters for individual modules and the entire system. Each block of the model is a source of different type messages, the number, content and dispatch time of which are determined by the settings of a specific block and the rules of the CANopen protocol.

The VMS model, like the simulated system itself, has a hierarchical structure. At the upper level, the blocks in accordance with their functional purpose and settings generate messages intended for transmission over the CAN network. At the lower level, the network model determines the priority of messages, the order of their transmission, and the time of delivery.

Since CAN messages are not addressed to any particular node, but unconditionally received by all nodes and only afterwards either processed or ignored, the process of receiving messages being not
modeled: any message is considered to be sent to the network and accepted by all nodes at the end of transmission. This assumption is consistent with the CAN network logic. Later, such network model characteristic can be added as a message transmission / reception error probability – in this case erroneous messages will be resent to the network.

The main window of the model interface is shown in figure 2. Using it an operator can add and remove blocks, set their type, configure the operation of such CANopen services, as heartbeat, synchronization (SYNC), TPDO. Taking into account the entered parameters, the model generates messages of each block, puts them in the transfer queue and forms sequence of transmitted messages based on queuing time, priorities and communication line load. The result of modeling is presented in a form of a table containing information for each message about its type and source, the queue time, the transfer start time, the transfer duration, the time of delivery and the delivery duration. An example of the results obtained is given in table 1. The results in a graphical form (figure 3) clearly show the degree of the network loading and the order of the messages transmission.

![Figure 2. Main window of the model interface.](image)

6. Conclusions
The methodology, developed by the authors for estimating the messages delivery time in the CAN network, and the program-logical model based on it significantly simplify the study of the network load in the initial stages of developing modular VMS. Using the model provides selection of optimal CANopen protocol parameters and observe their effect on communication processes in the CAN network.

| Unit   | Msg    | ID | Enqueued, us | TX Start, us | TX Duration, us | Delivered, us | Delivery Duration, us |
|--------|--------|----|--------------|--------------|-----------------|---------------|-----------------------|
| SC     | SYNC   | 1  | 0            | 0            | 50              | 50            | 50                    |
| SC     | TPDO1  | 385| 0            | 53           | 87              | 140           | 140                   |
| SVU1   | TPDO1  | 400| 53           | 143          | 106             | 249           | 196                   |
| SVU2   | TPDO1  | 401| 53           | 252          | 106             | 358           | 305                   |
| SU     | TPDO1  | 404| 53           | 361          | 87              | 448           | 395                   |
| SVU1   | TPDO2  | 656| 53           | 451          | 106             | 557           | 504                   |
| SVU2   | TPDO2  | 657| 53           | 560          | 106             | 666           | 613                   |
| SU     | TPDO2  | 660| 53           | 669          | 68              | 737           | 684                   |
| Unit  | Msg    | ID  | Enqueued, us | TX Start, us | TX Duration, us | Delivered, us | Delivery Duration, us |
|-------|--------|-----|--------------|--------------|-----------------|---------------|-----------------------|
| SVU1  | TPDO3  | 912 | 53           | 740          | 87              | 827           | 774                   |
| SVU2  | TPDO3  | 913 | 53           | 830          | 87              | 917           | 864                   |
| SC    | HBeat  | 1793| 0            | 920          | 59              | 979           | 979                   |
| SVU1  | HBeat  | 1802| 0            | 982          | 59              | 1041          | 1041                  |
| SVU2  | HBeat  | 1803| 0            | 1044         | 59              | 1103          | 1103                  |
| SC    | SYNC   | 1   | 20000        | 20000        | 50              | 20050         | 50                    |
| SC    | TPDO1  | 385 | 20000        | 20053        | 87              | 20140         | 140                   |
| SVU1  | TPDO1  | 400 | 20053        | 20143        | 106             | 20249         | 196                   |
| SVU2  | TPDO1  | 401 | 20053        | 20252        | 106             | 20358         | 305                   |
| SVU1  | TPDO2  | 656 | 20053        | 20361        | 106             | 20467         | 414                   |
| SVU2  | TPDO2  | 657 | 20053        | 20470        | 106             | 20576         | 523                   |
| SVU1  | TPDO3  | 912 | 20053        | 20579        | 87              | 20666         | 613                   |
| SVU2  | TPDO3  | 913 | 20053        | 20669        | 87              | 20756         | 703                   |

![Graphical representation of modeling results.](image)

**Figure 3.** Graphical representation of modeling results.

**References**

[1] CAN Specification. Version 2.0 Robert Bosch GmbH, 1991.

[2] ISO 11898-1:2015 Road vehicles -- Controller area network (CAN) Part 1: Data link layer and physical signaling. URL: https://www.iso.org/standard/63648.html

[3] ISO/IEC 7498-1:1994 Information technology Open Systems Interconnection Basic Reference Model: The Basic Model. ISO/IEC Copyright Office

[4] Nikiforov V V, Shkirtil V I 2010 Estimation of message delivery time in distributed real-time systems with CAN-interface. *Journal of Instrument Engineering* Vol. 7 pp. 33-39.

[5] C_CAN User’s Manual. Revision 1.2. / Robert Bosch GmbH, 2000. URL: http://www.keil.com/dd/docs/datashts/silabs/boschcan_user.pdf.

[6] LPC11Cx2/Cx4 32-bit ARM Cortex-M0 microcontroller. Rev. 3.2 – 4 January 2016. © NXP B.V. – 2016. URL: http://www.nxp.com/documents/data_sheet/LPC11CX2_CX4.pdf.

[7] CANopen – The standardized embedded network. URL: https://www.can-cia.org/canopen/

[8] The Basics of CANopen. © National Instruments, 2013 URL: http://www.ni.com/whitepaper/14162/en/#toc2

[9] CiA 301. CANopen application layer and communication profile. Version: 4.2.0. © CAN in Automation (CiA) 2011 URL: https://www.can-cia.org