CAN Bus XML Database Program

Márcio Monteiro, Ivo Ferreira

Abstract - **When we have a CAN network with several nodes that need to interact with each other, it is important to see their configuration, and so we need a tool that enable us to change/update the list of messages that are consumed/produced in each of those nodes in a quick and easy way. For that purpose,, in this paper we will discuss the implementation of our CAN XML Database program, that is a software that save the nodes configuration and enable the user to change/update/find and delete the configuration in each node.**

The information is saved in a XML file..

I. INTRODUCTION

When we have a CAN network with several nodes that interact with each other, it is important to have software that enable the user to save the nodes configuration and to change/update/find and delete those configurations in a easy and quick way. To obtain this message's configurations from the CAN network, we connect the computer serial port to the CAN network serial port, and with the interface created we can read the message's. Another useful addition to the program is the capability of showing the content of the XML file in the software, so that the user doesn't need to open the XML file with the internet explorer.

Our intent was to create a tool, with an intuitive graphical interface so that the use of the software could be the easiest as possible. To accomplish this task, it was decided to use Visual Basic 6.0, and determine the fields that compose each message that configures the node.

II. AN OVERVIEW OF CAN [3]

CAN (Controller Area Network) [1] is a serial bus system, which was originally developed for automotive applications in the early 1980's. The CAN protocol was internationally standardized in 1993 as ISO 11898-1 and comprises the data link layer of the seven layer ISO/OSI reference model. CAN, which is by now available from around 40 semiconductor manufacturers in hardware, provides two communication services: the sending of a message (data frame transmission) and the requesting of a message (remote transmission request, RTR). All other services such as error signalling, automatic re-transmission of erroneous frames are user-transparent, which means that the CAN chip automatically performs these services. The equivalent of the CAN protocol in human communication are e.g. the Latin characters. This means a CAN controller is comparable to a printer or a type writer. CAN users still have to define the language/grammar and the words/vocabulary to communicate.

A. CAN features

- A multi-master hierarchy, which allows building intelligent and redundant systems. If one network node is defective the network is still able to operate.
- Broadcast communication. A sender of information transmits to all devices on the bus. All receiving devices read the message and then decide if it is relevant to them. This guarantees data integrity as all devices in the system use the same information.
- Sophisticated error detecting mechanisms and re-transmission of faulty messages. This also guarantees data integrity.

CAN is a well designed communication bus to send and receive short real-time control messages at speeds up to 1Mbit/sec.

The CAN Network serial bus system is used in a broad range of embedded as well as automation control systems. It usually links two or more micro-controller-based physical devices.

The Original Equipment Manufacturers (OEM) design embedded control systems; the end-user has no or only some knowledge of the embedded network functions and is therefore not responsible for the CAN communication system.

Opposed to that, automation control systems are specified by the end-user. The system design including the CAN network services may be implemented by the end-users themselves or by a system house.

The main CAN application field include:

- Passenger cars.
- Trucks and buses.
- Off-highway and off-road vehicles.
- Passenger and cargo trains.

- Maritime electronics.
- Aircraft and aerospace electronics.
- Factory automation.
- Industrial machine control.
- Lifts and escalators.
- Building automation.
- Medical equipment and devices.

All nodes in the system receive every message transmitted on the bus and will acknowledge if the message was properly received. It is up to each node in the system to decide whether the message received should be immediately discarded or kept to be processed. An obvious benefit of this message based protocol is that additional nodes can be added to the system without the necessity to reprogram all other nodes to recognize this addition. This new node will start receiving messages from the network and, based on the message ID, decide whether to process or discard the received information. The ID (identifier) is represented as an 11 bit number. The priority of a message in CAN is defined by the identifier (ID), being this the most important part of CAN regarding real-time performance. This protocol is a CSMA/CA, where CSMA stands for Carrier Sense Multiple Access and what this means is that every node on the network must monitor the bus for a period of no activity before trying to send a message on the bus (Carrier Sense). Also, once this period of no activity occurs, every node on the bus has an equal opportunity to transmit a message (Multiple Access). The CA stands for Collision Avoidance. If two nodes on the network start transmitting at the same time a non-destructive bitwise arbitration method is utilized. This means that messages remain intact after arbitration is completed, even if collisions are detected.

All of this arbitration takes place without corruption or delay of the higher priority message. In order to support non-destructive bitwise arbitration logic states need to be defined as dominant or recessive, and each transmitting node must monitor the state of the bus to see if the logic state it is trying to send actually appears on the bus. CAN network defines a logic bit 0 as dominant bit and a logic bit 1 as a recessive bit. A dominant bit state will always win arbitration over a recessive bit state, therefore the lower the value of the message ID the higher the priority of the message.

Next we show in figure 1 the fields of the CAN frame:

Figure 1: Standard CAN Frame

A CAN base frame message begins with the start bit called "Start Of Frame (SOF)". This is followed by the "Arbitration field" which consist of the identifier and the "Remote Transmission Request (RTR)" bit used to distinguish between the data frame and the data request frame called remote frame. The following "Control field" contains the "IDentifier Extension (IDE)" bit to distinguish between the CAN base frame[3] (A CAN base frame message begins with the start bit called "Start Of Frame (SOF)", this is followed by the "Arbitration field" which consist of the identifier and the "Remote Transmission Request (RTR)" bit used to distinguish between the data frame and the data request frame called remote frame. The following "Control field" contains the "IDentifier Extension (IDE)" bit to distinguish between the CAN base frame and the CAN extended frame, as well as the "Data Length Code (DLC)" used to indicate the number of following data bytes in the "Data field". If the message is used as a remote frame, the DLC contains the number of requested data bytes. The "Data field" that follows is able to hold up to 8 data byte. The integrity of the frame is guaranteed by the following "Cyclic Redundant Check (CRC)" sum. The "ACKnowledge (ACK) field" compromises the ACK slot and the ACK delimiter. The bit in the ACK slot is sent as a recessive bit and is overwritten as a dominant bit by those receivers, which have at this time received the data correctly. Correct messages are acknowledged by the receivers regardless of the result of the acceptance test. The end of the message is indicated by "End Of Frame (EOF)". The "Intermission Frame Space (IFS)" is the minimum number of bits separating consecutive messages. Unless another station starts transmitting, the bus remains idle after this.) and the CAN extended frame [3] (The difference between an extended frame format message and a base frame format message is the length of the identifier used. The 29-bit identifier is made up of the 11-bit identifier (“base identifier”) and an 18-bit extension (“identifier extension”). The distinction between CAN base frame format and CAN extended frame format is made by using the IDE bit, which is transmitted as dominant in case of an 11-bit frame, and transmitted as recessive in case of a 29-bit frame. As the two formats have to co-exist on one bus, it is laid down which message has higher priority on the bus in the case of bus access collision with different formats and the same identifier / base identifier: The 11-bit message always has priority over the 29-bit message. The extended format has some trade-offs: The bus latency time is longer (in minimum 20 bit-times), messages in extended format require more bandwidth (about 20 %), and the error detection performance is lower (because the chosen polynomial for the 15-bit CRC is optimized for frame length up to 112 bits), as well as the "Data Length Code (DLC)" used to indicate the number of following data bytes in the "Data field". If the message is used as a remote frame, the DLC contains the number of requested data bytes. The "Data field" that follows is able to hold up to 8 data bytes.

The integrity of the frame is guaranteed by the following "Cyclic Redundant Check (CRC)" sum. The "Acknowledge (ACK) field" compromises the ACK slot and the ACK delimiter. The bit in the ACK slot is sent as a recessive bit and is overwritten as a dominant bit by those receivers, which have at this time received the data correctly. Correct messages are acknowledged by the receivers regardless of the result of the acceptance test. The end of the message is indicated by "End Of Frame (EOF)". The "Intermission Frame Space (IFS)" is the minimum number of bits separating consecutive messages. Unless another station starts transmitting, the bus remains idle after this.

III. RATIONALE ABOUT XML

What is XML? [2]
• XML stands for E**X**tensible **M**arkup **L**anguage.
• XML is a markup language much like HTML.
• XML was designed to describe data
Because describing data is what this Database is all about and given the XML flexibility, the file used can be understandable by both humans and computers. This allows one person, given only a basic text editor, to write the whole configuration table for the network.

Born with data transfer in mind and the fact that it is completely open, instead of proprietary, easily readable and writable, makes it attractive to exchange data between applications.

About the implementation of the configuration itself the choice of using the Visual Basic programming language is supported by massification of the use of MS Windows OS, permitting us to take advantage of using this easy to learn graphical programming language.

A. XML Database functionality list:

The XML Database functionality is, allow the user to change list variables, manage the database historic, insert variables and delete variables.

IV. INTERFACE

A Communication protocol

The need arise to define a communication protocol between the CAN nodes and the configuration application.

The communication between the PC and the node is done via RS-232, see screen in figure 2. Having this in mind, and bearing in mind that the time efficiency is not a big issue here. Because the configuration application interacts with a human, we decided to establish a communication protocol that is all done in ASCII because it leads to a simplification of the processing and implementation of the communications

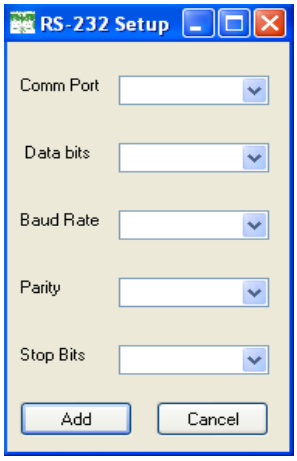

Figure 2: RS232 configuration

B. Configurator list editor window

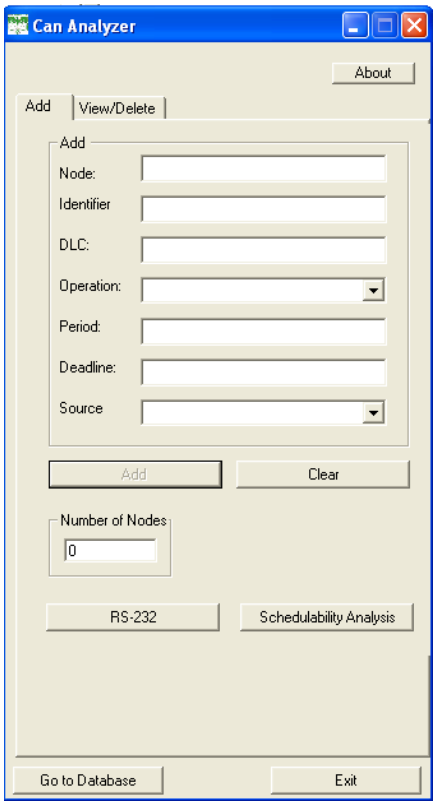

Figure 3: Node configuration

The node configuration is done editing its configuration list. In figure 3 is represented the tool screen were we can see a counter for the number of nodes already configured in the can bus. The “Go to Database” button allows the user to see the content of the XML file without opening the internet explorer.

C. View/Delete Configuration

In the screen showed above (figure 4) the user can make the search of configuration of the node by the identifier of the node. The results of the research will be shown in the "Edit Message" window. The buttons "Node Navigate" make the search one by one in all the field of the XML Database file. The "Delete" button, deletes the configuration of the node that the user wants to.

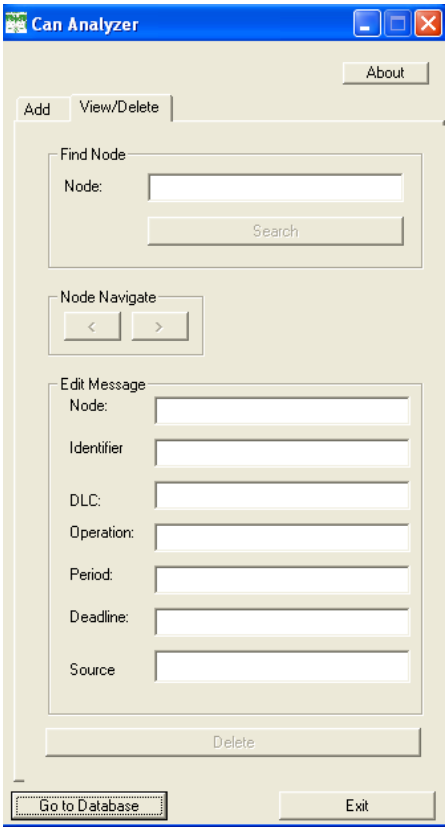

Figure 4: Delete node Configuration

D. View XML Database file

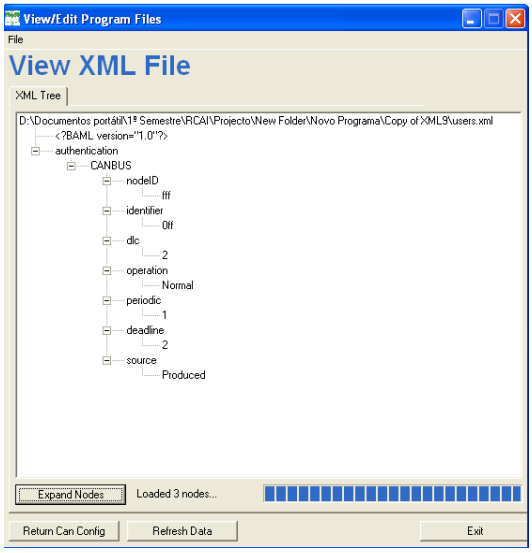

Figure 5: View Database Content

The tool screen above (figure 5) shows the way that the user sees the XML file without exiting from the tool. Here the user see all the configurations in the tree form.

E. XML Database file

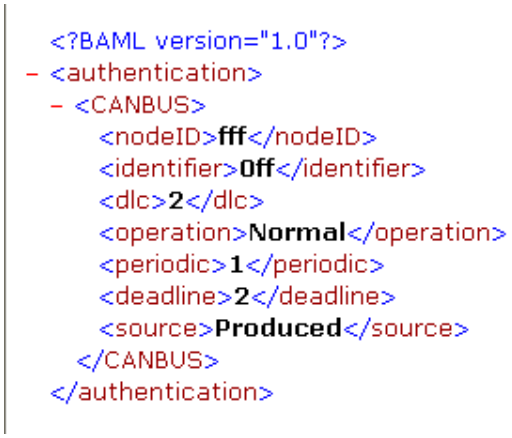

Figure 6: View XML file with Internet explorer

In the figure 6 we see the other way to view the XML file. We open the XML file that is in the program directory with the internet explorer.

V. TOOL TESTS

The application was tested by the observation method. This method includes two users.

This method allows obtaining usability information of the system, observing the users interacting with the tool and making tasks defined by the observator.

It was asked to two university students, possible system users that make some tasks.

Tasks to the users:

1. Configure 5 nodes
2. Configure RS-232 port
3. View node configuration
4. Delete node configuration
5. View database content
6. Exit Application

First user observation:

1ª Task:

He have configured the five nodes without problems, no error message has appeared because he never tried to configure an already configured node.

2ª Task:

He take sometime to understand that have some values already defined, and only have to choose between them.

3ª Task:

He has no problem so search the node and see its configuration.

4ª Task:

The delete operation was made successfully, but has been confusion when the application has changed window.

5ª Task:

He have see the database content, but have not understand immediately that had a “Expand Node” buttons that shows the all content of the database.

6ª Task:

No problem exiting from the application.

Second user observation:

1ª Task:

He has configured the five nodes without problems, but he tried to configure the same node two times by mistake, and it appears a message warning.

2ª Task:

He take sometime to understand that have some values already defined, and only have to choose between them.

3ª Task:

He has no problem so search the node and see its configuration.

4ª Task:

The delete operation was made successfully.

5ª Task:

He has see the database content

6ª Task:

No problem exiting from the application.

After the observations analysis has been verified that the options hidden and have a *ScrollBar* to see this option is not a very good option, but with some seconds of observation the users could achieve their objectives.

The application has a function that when a node configuration is deleted, the application goes from the delete screen (Fig.4) to the messages configuration screen (Fig.3), this solution turn on the sensation on the user that he is controlled by the system and not the inverse situation.

There was no task to see the content of the XML file by editing this file with a text editor, or internet explorer, and none of them as make the question about that, this give us the indication that the visualization that has been implemented inside of the application was very well accepted.

This test was made for Interface and human/computer course, to evaluate the usability of the tool.

VI. CONCLUSIONS

The administrator can manipulate all the data inside the XML Database file, and save all the message node configurations in the XML Database file and open this file with internet explorer, Microsoft word, etc.

In addition to this we also implemented a facility allowing the the administrator can see the content of the XML Database file in the tool created by us without the need of open the XML file.

Schedulability analysis.is extepcted to be implemented in the future

REFERENCES

[1] K. Pazul, “Controller Area Network (CAN) Basics), Microchip Technology Inc.,1999.

[2] James Britt, Teun Duynstee, “Professional Visual Basic 6 XML”, Wrox press

[3] **© 2001 - 2005 CAN in Automation (CiA).** http://www.can-cia.org/can/