

canAnalyser/32 - The indispensable Assistant for Design, Test and Service of CAN-based Communication Systems

by Roman Hofmann
and Christian Schlegel

IXXAT Automation GmbH
Doggenriedstr. 40
88250 Weingarten
Germany
Tel. (49) (751) 56146 0
Fax (49) (751) 56146 29
email: info@ixxat.de
web: http://www.ixxat.de

Within the last years CAN-based automation systems worldwide became very popular for any kind of industrial application. To support the effort, which is necessary for the development of CAN devices and systems, as well as for their installation, test and service, appropriate tools are necessary. Depending on the actual phase of the life-cycle of a distributed system, normally specific tools for development, installation and service of the system are in use. With the new generation, scaleable canAnalyser/32, each phase of the life-cycle of a CAN-based system is supported.

In the following the concept of this very versatile tool, which assists the complete life-cycle of a CAN-based system from the system design and development to system installation and operation will be presented.

The Communication Server

The basic technical concept of the canAnalyser/32 is the strict separation of communication and analyzing tasks. Communication tasks include the real-time critical transmission and reception of CAN messages, the management of identifiers as well as the management of the applied hardware platform (PC/CAN interface). Analyzing tasks include the user interfaces and the different functions for monitoring, stimulation or analyzing of the CAN bus traffic which are provided by the canAnalyser/32.

The kernel of the canAnalyser/32 is represented by the so-called 'Control Panel' which acts as the administration and management unit and Server of the

communication-oriented functionality of the canAnalyser/32. For accessing the PC/CAN interface, the Control Panel uses the Virtual-CAN-Interface (VCI), a universal programming interface provided by IXXAT Automation. The VCI supports any kind of PC/CAN interface board according to different interfacing standards, like ISA, PCI, PC cards or parallel port by a common programming interface. Therefore, an application which is build according to the VCI becomes independent of the specific type of interface hardware. This means that an application may be used stationary in the laboratory with a desktop PC and ISA interface or mobile on a laptop with PC-card or parallel-port-adapter. The VCI supports handling of 11- and 29-bit identifiers, signals error frames and allows a time stamp for each receive object with a resolution of up to 25 µs (fig. 1).

Client Applications provide User Functionality

The real functionality of the canAnalyser/32 is provided by so-called 'Clients'. These independent applications are based on the server functionality of the Control Panel and communicate with the server via an open application programming interface (Control-Panel-API). This API provides the functionality of an abstract CAN controller and provides functions for the configuration of object buffers and transmit/receive queues, the transmission and reception of CAN messages and the control of the real-time message tracing, which is integrated in the Communication Server. For the implementation of the canAnalyser/32, multithreading and intercommunication mechanisms provided by Windows 95/Windows NT are used as far as possible. Much effort was also invested in the design and development of the open Control-Panel-API, in order to ensure the fastest possible transfer of CAN messages between the Server and the Clients.

The basic version of the canAnalyser/32 comes with several basic Clients for transmission and reception of CAN messages (Transmit Client, Receive Client) statistical analysis (Statistic Client) and for processing of sequences of transmit messages (Batch Client). As an important advantage of the separation between communication and analyzing tasks, each client application can be started several times which results e.g. in multiple receive windows. Since in the Receive Client the CAN messages to be accepted can be configured by means of a message filter, a Receive Clients may be used to display only data of a certain device or function in

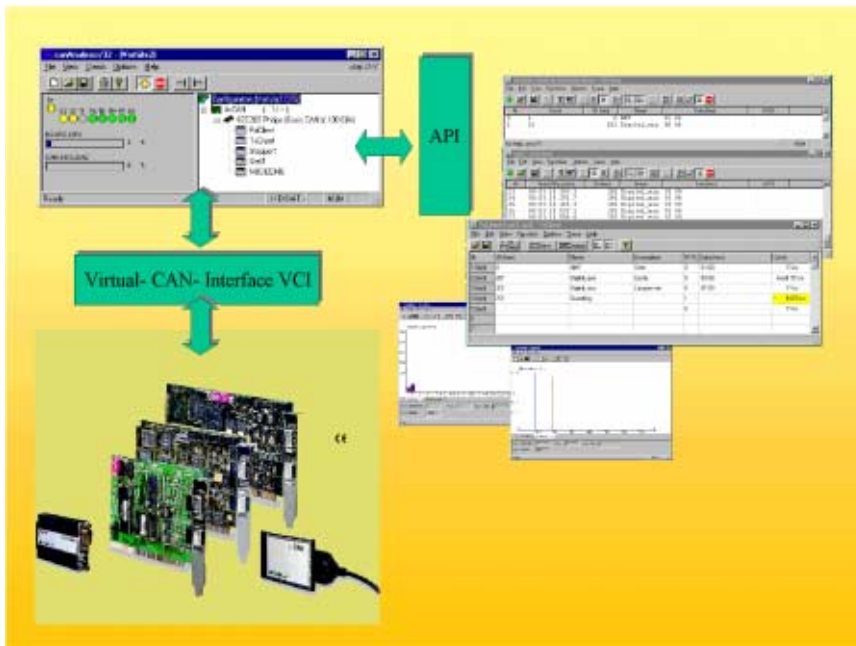


Fig. 1: Technical concept based on separation of communication and analyzing tasks

the system. So each device or function can be displayed in an own receive window. The received data can be displayed consecutively with symbolic names and time stamps or in statistical form with a message counter and the last received data of each message. The Transmit Client serves for the transmission of single messages. A message may be transmitted on user request or cyclically.

For a statistical analysis of the bus traffic the Statistic Client is provided. The Statistic Client displays the total number of transmitted CAN messages, error frames and the actual number of transmissions per second for each message. Additionally this Client also shows the bit rate in the system by the time.

The transmission of complete message sequences (e.g. for a simulation of a device or system) is supported by the Batch Client. This client may also be used for an online replay of recorded trace data.

Besides of these basic clients the Control Panel itself provides a real-time, high performance, trace functionality. Since a message trace is recorded directly on the harddisk the maximum length of a trace only depends on the available storage capacity. A message trace can be triggered by means of configurable trigger events. Triggering is possible via the message identifier, message data bits or the occurrence of a remote or error frame.

Standard Clients provide additional Functionality

In addition to the functionality of the basic Clients specific functionality is available in form of further standard client applications like the Data Interpretation Module (DIM-Client), CANopen or Programming Client. With the DIM Client the interpretation of transmitted or received data is supported by almost any type of data and conversion rules. Therefore, process variables, parameters, status and control information is presented in comprehensible form to the user. This client therefore is especially suitable for installation, test and maintenance of distributed, CAN-based Automation systems, since the service people must not know any detail of the communication protocol.

In a similar manner, the specific CANopen Client supports the developer of a CANopen-based system during development of the system. In this case the CANopen Client provides an user interface according to the services and data formats of the CANopen standard.

Especially for the fast realization of more complex test or simulation procedures the

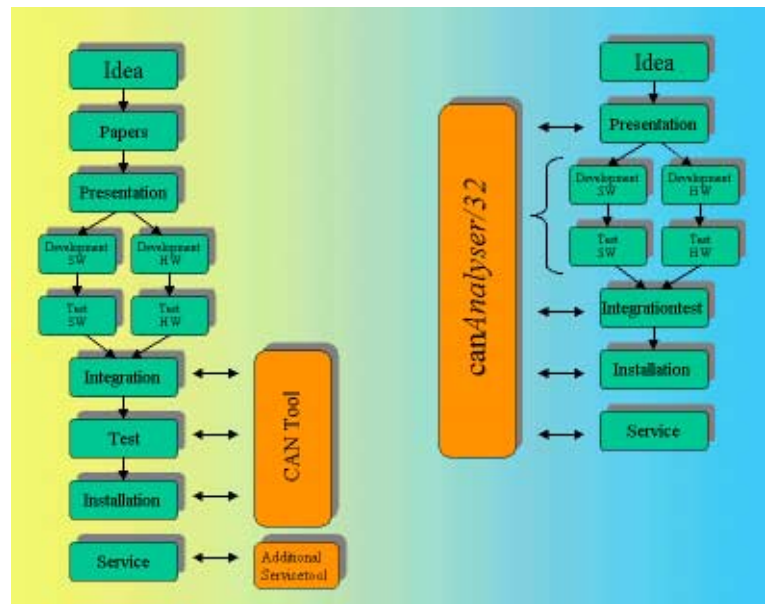


Fig. 2: System development process without and with canAnalyser/32

„Programming Client“ is provided. This Client supports the development of specific test and diagnosis applications with minimum effort. Specific test applications may be programmed by using a subset of the C programming language ('Little-C') and an event-oriented programming concept. Thereby application specific functions are assigned to external events like 'Start-of-Program', 'Received-CAN-Message', 'Time Value', 'Pressed Key' or 'Pressing of a User defined Button' on a toolbar.

User-Defined Clients for System Design, Emulation and Testing

Besides of the above-mentioned basic and standard clients the realization of application-specific applications for device or system emulation is supported by means of the open Control-Panel-API. This open programming interface represents one of the main advantages of the canAnalyser/32 concept since it allows to use the canAnalyser/32 indeed over the complete system development cycle.

In the traditional way of developing a system a certain CAN tool is used only during the development of devices and perhaps for the installation of the system. Later on other tools are required for servicing the system. With its open API the canAnalyser/32 can also be used in the design steps and later on for service and diagnosis. (fig 2)

The usage of the canAnalyser/32 for system emulation and testing now shall be explained by means of the design and development of a simplified elevator system.

For a simplified elevator model only a elevator/drive controller unit, a car panel and several floor panel units shall be considered. A user request is transmitted from the car or floor panels to the elevator controller unit which will process the request and control the moving of the car to the desired floor. The elevator controller also distributes the actual car position for indication on the car and floor panels.

Already for a first presentation and evaluation of the system, a complete functional prototype of the system can be developed, based on the canAnalyser/32. With the open Control-Panel-API, specific clients may be written by using any of the common Windows programming environments like Delphi, Visual-C++ or Visual Basic. In our example, specific clients have to be written for the emulation of the elevator controller, car and the floor panel units. The clients communicate with each other already by means of CAN messages. Because more than one instance of a client can exist in the system, only one floor panel client must be written.

The development of the elevator controller and panel software first can be performed in the PC environment with a later portation of the developed code to the target hardware. A major advantage of this proceeding is the usability of the comfortable debugging provided with the PC environment. Additionally the hardware can be developed at the same time (fig. 3).

During development and testing of a unit, for example the elevator controller, the simulation of the functionality of the other units (car and floor panels) is required. For the case, that the complete system first was

