# OPC for CANopen and DeviceNet

Rainer Gallus, Softing GmbH, Haar, Germany

## OPC for CANopen and DeviceNet

## Abstract

**The availability of the Layer 7 Communications CANopen, DeviceNet™ and SDS has opened opportunities for increasingly complex applications in the CAN application area. At the same time, the applications are expected to be executable for multiple operating system environments and a wide range of interface formats. For the developer of application software, this means that s/he is increasingly confronted with the task of adapting the application for operation in the new and different environments over the lifetime of the product.**

**Therefore, the large demand for standard software and interfaces exists in order to obtain a general overview on the costs and duration of development.**

**This article describes which requirements must are to be taken into consideration in the implementation and how modern APIs (Application Programming Interface) and new standards such as OPC (OLE for Process Control) can simplify development and maintenance of CAN applications, thereby reducing the expenses for maintenance and development considerably.**

### Introduction

Up to now, CAN was primarily employed as an OSI Layer 2 implementation in closed systems. These systems consist of a network having several CAN nodes and habitually do not exchange data with other systems. Typical applications are common, for example, in the textile industry or medical technology. In these applications, company-specific proprietary solutions developed which could not be rehosted in another system environment without related effort.

However, due to the development of standardized communication protocols such as CANopen, DeviceNet™ and SDS, all of which are based on CAN, new opportunities emerge for the usage of CAN.

In addition to the fast exchange of process information representative for CAN, these standards offer the functionality of a fieldbus system for integrating and configuring diverse components of various manufactures. Since controls such as PLCs, industrial PCs or standard PCs are increasingly being applied in these applications, a strong demand for PC interfaces exists for interfacing the control to the CAN network.

The different methods of implementation are presented below, whereby the affects of the applied interfaces and associated software on the project to be implemented are taken into account.

### Program requirements

Before development is initiated, the exact requirements of the system to be developed must be defined. These requirements include, for example: Transfer rates; network expansion; type of device to be developed (PLC, industrial PC; sensor or actuator).

Moreover, the communication protocol must be defined. A CAN OSI Layer 2, CANopen, DeviceNet™ or SDS Communication Protocol can be used for the communication. In this respect, selection of the communication protocol can be determined by a variety of factors such as: Previously existing devices or software; hardware performance capability; target market for the aggregate system.

Which operating system is to be employed has a major influence on the development in the PC sector. Can a standard operating system such as Windows 95 or Windows NT be used? Or is a real-time operating system required? When standard operating systems are used, pre-written portations of the communication and driver software are available. When real-time operating systems are used, in contrast, expensive rehosting must still be made under certain circumstances.

The type of the device applied determines which interface formats (ISA, PCI, PCMCIA or PC/104) are supported. When previously existing device families are applied, a customer-specific re-design of a standard interface may be necessary.

If a customer-specific new development is necessitated, the matching microcontroller and CAN chips generally have to be selected, whereby the following must considered: Microcontroller performance; availability of corresponding communication software for these components; long-term availability of the selected components.

### OSI Layer 7 Communication Protocols for CAN

In addition to numerous, user-specific solutions, three high-level, standardized and application-independent protocols (CANopen, DeviceNet™ and SDS) are available for CAN. These high-level communication protocols correspond to the three fieldbus layers in the ISO/OSI model, whereby the following are implemented: physics; data exchange; application layer.

Services and objects are defined in the application layer for running the communication. These functions defined in the specification can be implemented by the user, alone, or assumed from protocol software already implemented by various companies. The protocol software is provided in the scope of delivery along with the standard interfaces offered or as source code for rehosting to special devices.

In order to simplify the use of standardized devices, the high-level protocols support so-called device profiles. These device profiles describe the typical behavior of device families. This allows the user to use devices of a device family of various suppliers without having to undergo expensive modifications in case of a change.

CANopen is the further development of **CAL** (**C**AN **A**pplication **L**ayer). The device profiles are specified and defined via the CAN User Group **CiA** (**C**AN **i**n **A**utomation). CANopen presently focuses on the European market.

DeviceNet™ was originally developed by Allen-Bradley. The former is managed and represented internationally by **ODVA** (**O**pen **D**eviceNet™ **V**endor **A**ssociation). Even DeviceNet™ disposes of objects consisting of data (attributes) and services. Access is made via a hierarchical addressing scheme. DeviceNet™ has gained wide acceptance in the USA, Japan and UK.

**SDS** (**S**mart **D**istribution **S**ystems) was developed as the third OSI Layer 7 Protocol by Honeywell Micro Switch. SDS provides a special application layer and defines an object-oriented, hierarchical device model. It is specifically employed in applications utilizing sensors and actuators; e.g., conveyor belts in airports and breweries.

### Operating systems and driver software

Derivative CAN interfaces and the complementary APIs and libraries are available for all standard operating systems such as DOS, Windows 3.1,

Windows 95, Windows NT as well as UNIX. Suppliers of the corresponding CAN technology support the user in applications involving special operating systems such as real-time or proprietary operating systems. Rehosting in the user-specific operating system environment can be achieved either by the supplier of the CAN technology or by users themselves. For this purpose, the relevant program sections of the API are provided as source code, allowing the portation. Qualified system providers also back the user extensively and provide the necessary support.

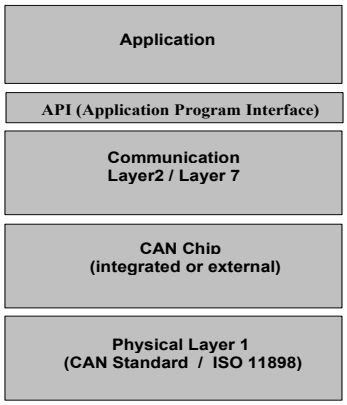


**Fig 1 Schematic of a CAN Interface**

### Standard formats available for PC CAN interfaces

CAN interfaces are offered as both "intelligent" and "non-intelligent" interfaces. The difference between these are that intelligent interfaces exhibit a separate microcontroller, whereas non-intelligent interfaces only provide CAN controllers and transceivers (see Fig. 1).

An intelligent interface uploads the communication software to the interface. All time-critical factors associated with the communication software are executed on the interface, thereby relieving the PC considerably. Data is usually transferred between the interface and the PC via a dual-port RAM which can be mutually read and written. Interfaces of these types are especially suitable for implementing applications intended for high-speed networks and processing the complex Layer 7 Communication (see Fig. 2).

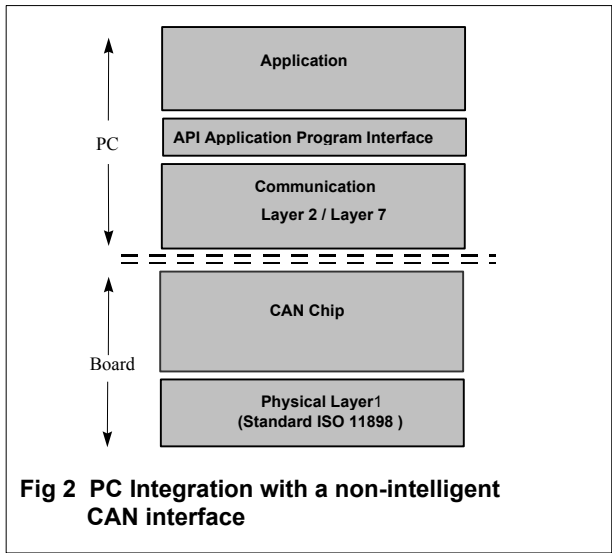


**Fig 2 PC Integration with a non-intelligent CAN interface**

In applications involving non-intelligent interfaces, the communication software is processed on the PC. Due to the fewer number of components, these interfaces are less expensive, but have restrictions in processing networks having higher data transfer speeds and larger quantities of data. Before being applied, however, these cards must be checked whether suitable for the proposed application (see Fig. 3).

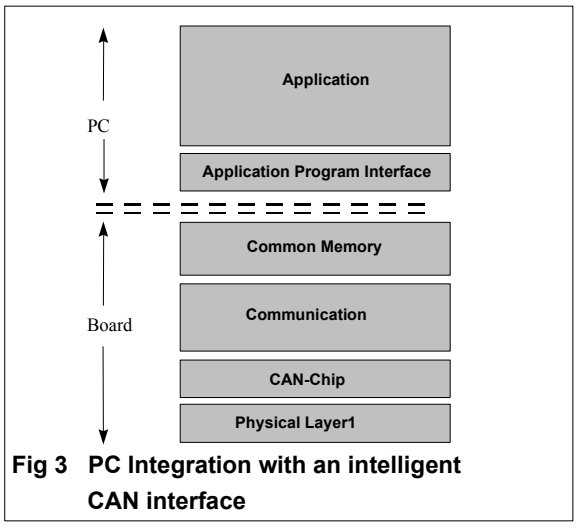


**Fig 3 PC Integration with an intelligent CAN interface**

CAN interfaces are offered in all common formats. ISA and PCI interface formats are available for use in the PC. PC Cards in

PCMCIA format type II (see Fig. 4) are specially available for mobile use, together with tools used in the configuration, integration, maintenance and service.

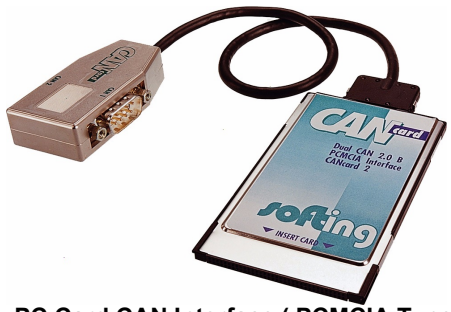


**Fig 4 PC Card CAN Interface ( PCMCIA Type II )**

PC/104 is another industry-accepted interface format. The bus interface of a PC/104 board behaves similar to the bus interface of a board having ISA format. This technology permits extremely small and compact devices to be implemented. All standard PC components such as motherboards, interface cards, etc. are offered in PC/104 format. Together with the corresponding CAN interfaces, extremely powerful CAN stations can be developed in this technology. These devices, with their almost cube-like enclosures, are already applied today in diverse industrial solutions.

## API Application Programming Interface

The CAN **A**pplication **P**rogramming **I**nterface, simply **API**, forms the interface between the customer application and the communication software. The API provides services and interfaces for transferring the data of the respective communication software, the API of which represents the conversion of the specification relating to the applied communication software. Even though, however, the different suppliers refer to the respective standard when implementing their communication software, the products offered are always company-specific solutions; i.e., the call interface of manufacturer A does not match that of manufacturer B!

However, it is not self-evident - even within a product family - that all call interfaces offered are compatible to one another. In most of the solutions offered, the API operates directly with the device drivers supplied for the respective interface. Moreover, different APIs are usually offered for the various operating systems. It is therefor important that the call interfaces offered are not only inter-compatible for the various interface formats, but also for the different operating systems.

Some system developers place special attention on this compatibility under various API versions in the implementation and further enhancement of the communication software, meaning no unnecessarily high portation expenses occur when varying the operating system environment or the CAN interface.

A exemplary 32-bit API implementation is depicted in Fig. 5. The application (a CANopen implementation in this case) can be employed with both Windows 95 and Windows NT. Which device drivers are required (device drivers for Windows 95 or Windows NT) is decided first in a library below the API, designated Vcard32.dll in Fig. 4. Another essential aspect is that hardware-relevant information does not exist any more in the API.

Hence, long-term assurance is made that a large number of different interface formats for various operating systems can be operated via a single API. This type of implementation offers the advantage that the number of program versions which must be maintained is significantly reduced.

Reduction of the application program versions signifies a considerable savings in cost and time in the continued development and support of these products.

## OPC - A new standard for visualization and fieldbus systems

A further development of this concept is **OPC**, **O**LE for **P**rocess **C**ontrol, whose objective is to achieve a standardized, PC-internal and computer-wide communication standard, enabling the interfacing of manufacturer-neutral programs and the

use of these as components in a complete system.

Among components, the aim of the open, vendor-independent communication between field equipment is continued here within the PC. Similar to the use of printer drivers in Windows, devices and/or driver specifications beyond the OPC interface are to remain concealed.

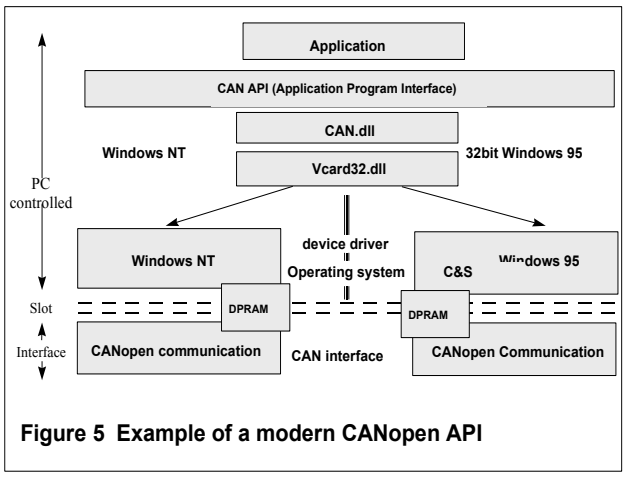


**Figure 5 Example of a modern CANopen API**

In this case, clear demarcation of component manufacturers and software manufactures occurs (see Fig. 6). The suppliers of components such as CAN interfaces need only develop an (OPC) driver for the respective hardware component.

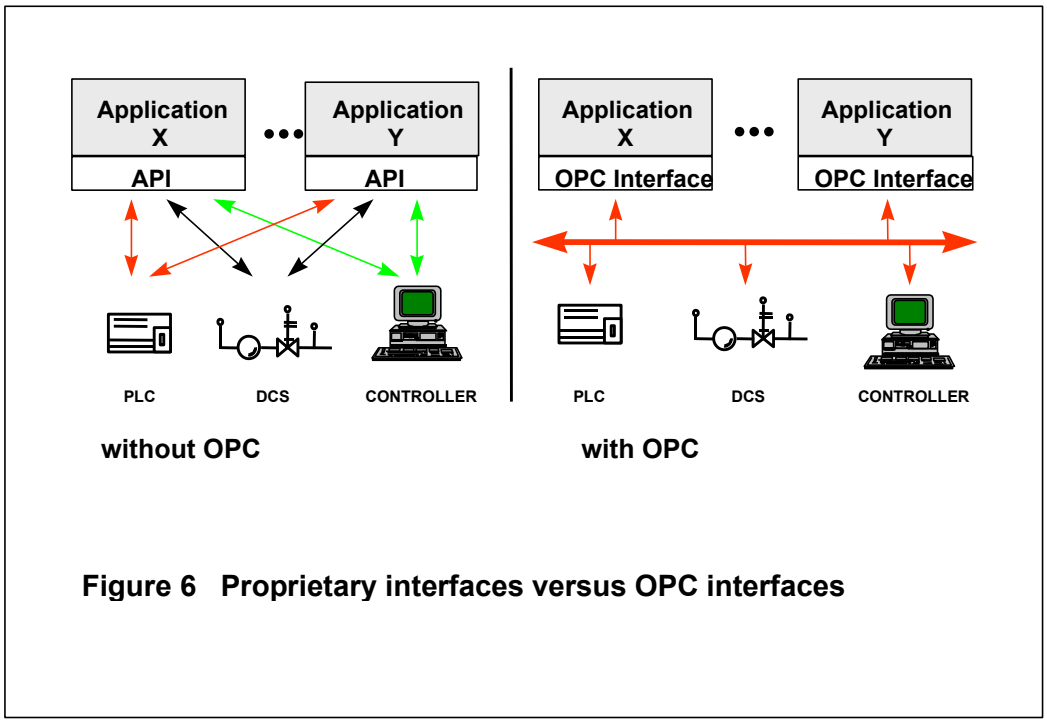


**Figure 6 Proprietary interfaces versus OPC interfaces**

The integration of standards for CAN such as CANopen, DeviceNet™ and SDS with OPC leads to solutions compatible and interoperable to solutions of other OPC suppliers.

### What is concealed behind OPC technology?

As OPC Client, various application programs such as visualization systems and process control systems can use the functionality integrated in the OPC Server (interface with OPC driver). OPC Clients and OPC Servers can be executed in a PC or even distributed in the network over several PCs. Distribution of the

components among several PCs is backed by DCOM (Distributed COM), Microsoft's distributed interaction protocol.

In this case, DCOM utilizes the Ethernet technology and is a constituent of all new Windows operating systems for Windows NT 4.0 or later. An update can be obtained for Windows 95. The OPC Server enables process variables to be read and written, not only variable values can be transferred, but status and time information as well. In addition, the OPC Server features further functionality, enabling both the OPC Client and the user to differently combine the process variables to be captured and to adapt their acquisition to specific requirements.

Examples of this are the combination of dynamic variables in a group with common values in regard to update rate and threshold value for reporting changes. How the available technology is applied is presented below:

OPC is based on the well-known OLE/COM technology by Microsoft. Two requirements were the basis for developing OLE:

1. To be able to process documents with different information representations in a program.

2. To ensure binary compatibility between software components (backward compatibility, autonomous upgrades, co-existence of old and new software).

The efforts led to the definition of a **C**omponent **O**bject **M**odule (**COM**) by Microsoft in the first half of the decade. The aim of this model is the guarantee of binary compatibility between software components.

Special aspects of this model are interaction between the components based on access to methods, components and objets combined in interfaces. A Component Object contains several interfaces which encapsulate access to the object. Furthermore, the term "Object" is used as a synonym for the term "Component Object".

The following are defined in COM:

- A methodology how an object can be interrogated at runtime to determine whether it supports a specific interface.

- A procedure how the life cycle of objects is managed. For his purpose, a reference counter is used which is incremented the first time an interface is used and decremented again when released. If the counter has the value zero, the object is no longer required and is able to terminate itself.

- How interactions between objects can be executed beyond process limits. This is necessary because pointers to interfaces and method parameters are only valid within a process space.

- Procedure for identifying and loading objects.

OPC defines two objects:

- Object structure

- Interfaces, methods and parameters for individual objects.

The term "OPC Server" was used twice in the definition of the terms. An OPC Sever denotes an executable file which yields

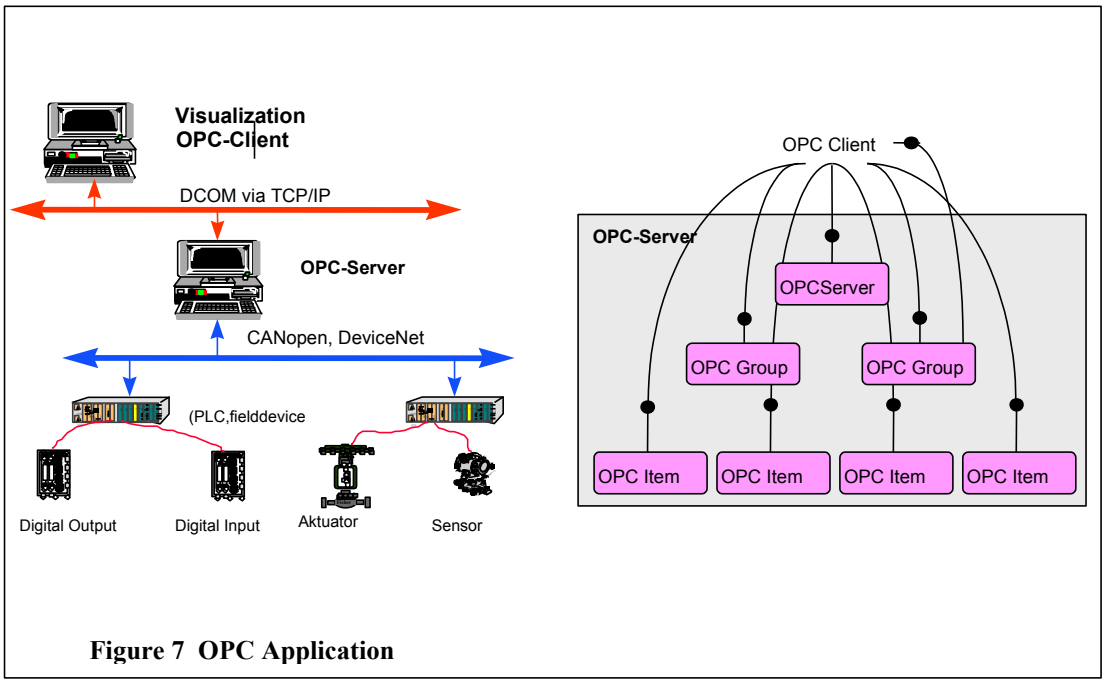


**Figure 7 OPC Application**

functionality as determined in the OPC specification. Parallel to this, an "Object" is also termed as OPC Server within the latter.

An OPC Server can contain entities of the objects: OPCServer (1x); OPCGroup (at least 1x); OPCItem (at least 1x) (see Fig. 7). In concern to this, an OPC Server always accesses the object interfaces; i.e., in order to properly terminate an OPC Server, an OPC Client must release all objects again by calling "Release".

### Where can OPC Servers optimally be applied?

One particular area of application for the OPC Server is use as a constituent in visualization systems. Potential use also exists, however, in process control systems as well as systems used in remote maintenance and diagnostics.

The use of the OPC Servers is in its pioneer phase, meaning that further applications are certainly still to evolve in the future.

### Summary

CAN and the CAN-based OSI Layer 7 implementations CANopen, DeviceNet™ and SDS give the user access to powerful technologies, allowing implementation of projects in industrial automation. Moreover, the costs for further development and maintenance can be significantly reduced through the selection of modern CAN interfaces and corresponding call interfaces.

New technologies such as OPC provide users with additional opportunities in ensuring long-term interfacing of their program systems to many different applications, thereby eliminating expensive portations.

OPC is operated by an independent organization. A large number of companies which cooperate in the advanced development and maintenance of OPC are represented in this organization. Corresponding committees perform conformity and interoperability tests and therefor act as superior inspection authority.

The main advantages in using OPC are all that the suppliers of software solutions (e.g., a visualizations system) still need is a generic driver, namely an OPC Client

interface and no longer a large variety of specific drivers, as in the past.

Hence, system integrators can combine a desired process visualization with various PC cards, for which OPC software is available. The solutions which result are also network-capable due to the Distributed COM incorporated by OPC.

Despite all the simplification of the interfaces in the project design, an important factor remains which should not to be overlooked: Training of employees involved in the project. Training for CAN, CANopen, DeviceNet™ and SDS and corresponding OPC interfaces is offered by various system houses. The training can be conducted either in-house at the supplier or remotely at the customer site.

Softing GmbH
Richard-Reitzner-Allee 6
85540 Haar near Munich
Tel: ++49 / 89 / 45656 – 323
Fax: ++49 / 89 / 45656 – 399
Email: rainer.gallus@softing.com
Homepage: http:\\www@softing.com