CANopen virtual device architectures

Holger Zeltwanger (CAN in Automation)

In order to standardize network system applications independently of the device implementation it is necessary to introduce the concept of virtual device interfaces. Several CANopen application profiles using the concept of virtual devices will be introduced briefly. In particular, there are methods how to define virtual device interfaces and how to describe a default PDO communication scheme allowing peer-to-peer data transmission. Hints are given on how to overcome the limitations regarding the number PDOs and object dictionary entries.

In industrial network technology it is commonplace to specify the OSI lower-layer and the higher-layer protocols in order to achieve the coexistence of devices in the same physical network. Normally this includes a network management. In order to achieve inter-connectability, inter-workability, or even inter-operability of devices, additional definitions of data access, data types, and parameter semantics are required.
The IEC device profile guideline defines the following levels of functional compatibility:

- Coexistence: lower-layer protocols
- (Tolerance: higher-layer protocols)
- Interconnection: data access
- Inter-workability: data types
- Interoperability: parameter semantics and application functionality
- Interchangeability: dynamic behavior.

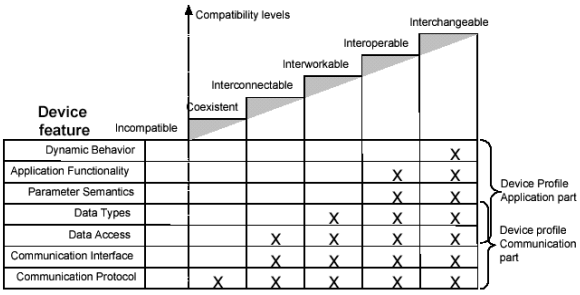

Fig. 1: IEC compatibility definitions

In CANopen technology, standardized and proprietary device profiles define data access (object dictionary entries), data types, parameter semantics and application functionality. For more sophisticated profiles this includes device-specific state-machines. In all generic device profiles, the PDO communication as well as the emergency transmission is pre-defined. There are only up to four TPDOs with valid COB-IDs and up to four RPDOs with valid COB-IDs. The default TPDOs transmitted by the NMT slave devices are received by the NMT master device. And the default RPDOs consumed by the NMT slave device are produced by the NMT master device. The reason is that generic CANopen devices will provide only a plug-and-play function with a default PDO master/slave communication capability. If the user likes to have plug-and-play systems with predefined PDO cross communication between NMT slaves, he has to define an application profile.
Using different generic device profiles limits not only the PDO communication to master/slave behavior, but also requires configuration effort in order to integrate the network system. This is due to the fact that the CANopen dictionary entries differ from one profile to another. Depending on the device type object (1000_h), the object entries from 6000_h and following have a specific meaning. This may require PDO mapping configuration as well as PDO linking.

Network classification definition

There are different requirements regarding plug-and-play capability, flexibility, and configurability in automation, embedded and deeply embedded networks. Deeply embedded networks are highly optimized requiring very high communication flexibility by means of PDO configuration (linking

and mapping). Designers of deeply embedded systems should have a deep knowledge of the communication system. Embedded networks are not visible for the end-user. Designers of embedded networks like to use standardized profiles in order to limit the configuration effort. Off-the-shelf plug-and-play capability is required. In those embedded networks it is acceptable that specific communication functionality needs configuration of the PDO parameters. Automation networks are designed by end-users. The demand on off-the-shelf plug-and-play capability is very high.

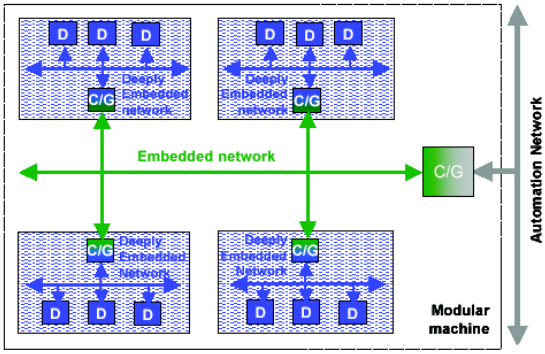

Fig. 2: Networking architecture for modular machines

In modern modular machine systems, the machine module manufacturer uses deeply embedded networks. The machine-system builder implements embedded networks, and the production line designer utilizes automation networks. In deeply embedded networks, it is commonplace to integrate devices implementing proprietary and standardized device profiles. In embedded networks, a pre-defined PDO broadcast and peer-to-peer communication may be required. In addition, the designer of embedded networks may request a higher granularity of devices not related to the physical implementation. CANopen provides the possibility, to implement up to eight devices in one physical device. This means also up to eight motion controllers may reside in one physical CANopen device. However, not all CANopen device profiles allow this possibility. For example, the one device compliant to the CANopen profile for measuring devices and closed-loop controllers may need the entire object dictionary address space.

Device definitions

According to the ISO 15745 international standard, a device is an entity that performs control, actuating and/or sensing functions and interfaces to other such entities within an automation system. The IEC 61499-1 uses another device definition: Networked independent physical entity of an industrial automation system capable of performing specified functions in a particular context and delimited by its interfaces.
The IEC device profile guideline describes additionally a device interface model. Using this model, the internal structure is not relevant to the application. The following interfaces are defined:

- Control interface
- Process interface
- Configuration interface
- Diagnostics interface

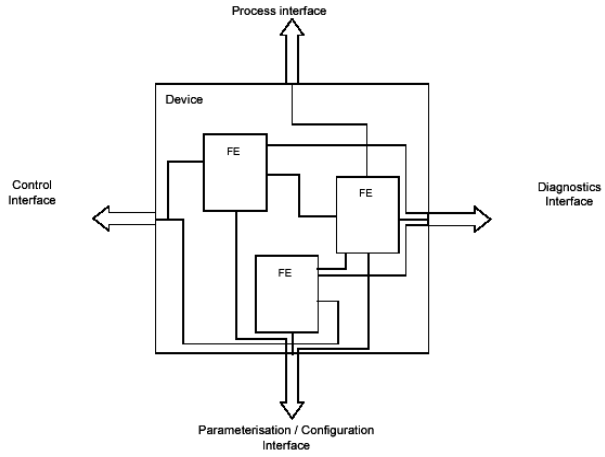

Fig. 3: IEC device interface model

The device profile guideline also describes a modular device structure. This hierarchical device structure regards the modules as resources (e.g. in IEC 61131-3 and IEC 61499) or logical devices or virtual devices, which can be subdivided into functional elements. “Functional element” may be described as parameter list members, function blocks and objects. In CANopen, the parameter list members are defined by several attributes (e.g. index and sub-index, data type, category, access type, value range, default value). Parameter list members may be variables, array, or records.

The concept of virtual devices

The concept of virtual devices allows defining a granularity of functionality that is optimized for the application field. A virtual device is a set of application functions. Several virtual devices may reside in the very same physical device. However, a virtual device shall not be distributed to several physical devices. Depending on the required granularity of functions, the defined virtual device provides one or more process data and additional configuration parameters.

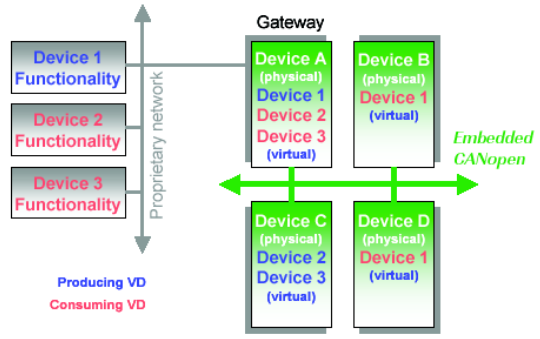

Fig. 4: Network transparent gateways using the concept of virtual devices

The concept of virtual devices allows designing transparent gateways. In one system design, the user may integrate just one CANopen physical device to a gateway representing all the other virtual devices. In another implementation, the system designer may connect just one virtual device via the gateway and a proprietary interface to the CANopen network. Or he may do something in between.
It is also possible to split the logical CANopen network into different physical segments. This allows application-specific network topologies and architectures depending on timing, busload, and other requirements.

Profile for passenger information

The first CANopen application profile was specified for public transportation passenger information systems. This profile was defined jointly with the VDV (German non-profit organization for public transportation systems) and the Finnish Buslan project. In the meantime, the profile has been submitted for European standardization.
The profile defines several virtual devices. Each virtual device supports a number of mandatory and optional application objects. All virtual devices share the same object definitions meaning that the object entry definitions in the area from 6000_h to $9FFF_h$ are unique for all devices compliant to this application profile. Some of these entries are related to the physical device, e.g. object 6000_h that indicates the supported virtual devices.
There are two pre-defined PDO connection sets. In the typical configuration, the main on-board computer virtual device transmits PDOs to the destination indicator virtual device, the next stop indicator virtual device, the ticket canceller virtual device, and the ticket printer virtual device. The main on-board computer virtual device receives PDOs from the identification device. If both of these virtual devices reside on the very same physical device, CAN communication is not necessarily required. The application profile defines the PDO COB-IDs and the PDO mapping. The PDO COB-IDs do not depend on the CANopen node-ID as they do in generic device profiles. The system designer cannot integrate un-configured generic CANopen devices due to the risk of doubly used CAN identifiers.
This default minimum and typical configuration does not satisfy more complex passenger information systems. In this case, the system designer has to configure dedicated PDOs regarding communication and mapping parameters. The configuration effort is still high.

Profile for lift control systems

In the application profile for lift control systems, all physical devices may support MPDOs to be received and to be transmitted. Some virtual devices specify predefined PDO with COB-IDs depending form the physical device's node-ID. In addition, the physical device has to support those PDOs with virtual device-specific COB-IDs. For example: The call controller virtual device for lift 1 transmits the PDO with the CAN identifier 400_h, the call controller for lift 2 transmits the PDO with the

CAN identifier 401_h, etc. The CANopen application profile for lift control systems can run eight single lift control systems each for up to 254 floors.
In this application profile, most of the 512 TPDO and 512 PDOs are used. For non predefined PDO communication, the optional MPDOs may be used. Each physical device may support up to 127 MPDOs to be received.
The MPDO transmitted in Destination Address Mode (DAM) is received by all physical devices that implement the corresponding MPDO consumer. Depending on the multiplexer (corresponding to index and sub-index) the MPDO consumer stores the received data into its object dictionary or it ignores the received data.

Basic function	Byte 0	*LSB*
Sub function	Byte 1	
Source lift	Byte 2	
Source panel	Byte 3	
Source door	Byte 4	
Function data	Byte 5	*MSB*

Fig. 5: Input multiplexer for lift panels

In order to overcome the limitation in the object dictionary with respect to the supported PDOs, the application profile uses not only MPDOs but also other multiplexing objects. The virtual input mapping object (6010_h) and the virtual output mapping object (6020_h) contain one of the digital input or output group objects (6100_h to $611F_h$, respectively 6200_h to $621F_h$). The application profile defines a similar multiplexing object for sensors, the virtual sensor mapping object (6012_h) containing one of the sensor group objects (6500_h to $651F_h$).
The Unsigned48 virtual input mapping objects contain function data, source door, source panel, source lift, sub function, and basic function information. The Unsigned48 virtual output mapping objects and the sensor group objects are structured similar.

Profile for door control systems

The application profile for door control systems specifies a high granularity of virtual devices.

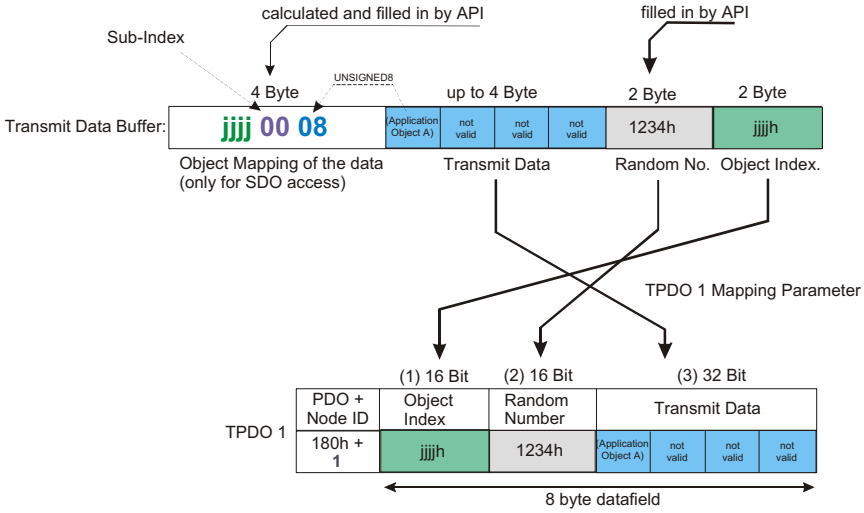

Fig. 6: Door profile transmit object

Each virtual device producing data uses just one or two 32-bit objects for process data. The corresponding virtual device consumers provide one or two 32-bit objects for the received process data. The process data objects to be transmitted are mapped into the transmit data buffer's application data sub-object (6001_h). This sub-object is mapped into the standard TPDO_1 of the physical device.
All physical devices consume by default all TPDOs. The physical device that supposed to process information from a specific virtual device will store the received application data in the appropriate collection object. Of course, it is also possible to configure a different behavior using a configuration tool. In particular, the user may like to disable RPDOs that are not required. The node-ID of the physical devices may also be assigned automatically by means of a node-claiming procedure. It was the intention of this application profile to unburden the system designer from any PDO configuration.

Profile for road construction machinery

The application profile for road construction machinery was developed by the OSYRIS (Open System for Road Informa-

tion Support) research project. It defines the CANopen communication in several road construction machines. The sensor functionality object (6010_h) specifies, which sensors are implemented in this physical device. The required information object (6410_h) describes, which sensor signals are necessary to be received by this physical device. The NMT master device reads all this information, in order to configure automatically all TPDOs and RPDOs in the NMT slave physical devices. Each sensor object entry (e.g. angle position or paver speed) can be regarded as virtual device.

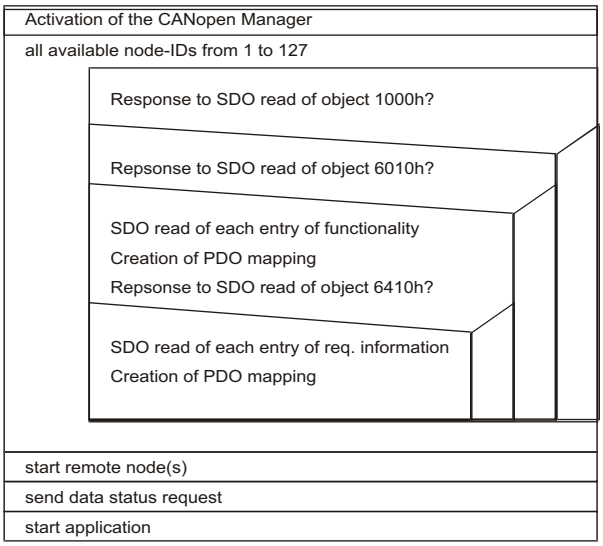

Fig. 7: CANopen boot-up procedure for virtual road construction machine devices

Summary

The CANopen application layer provides the possibility to sub-divide a physical device into virtual devices. An option is to split the standardized device profile area (6000_h to $9FFF_h$) into eight 800_h segments using each for one standardized device profile. In addition, it is possible to describe virtual devices in application profiles using a number of object entries. In extreme, a virtual device supports just one object. In to order to overcome the CANopen dictionary limitations regarding the number of PDOs and the number of object entries, it is possible to use several multiplexer techniques.
The virtual device concept allows the device manufacturer to implement several functions in one physical device. The granularity of physical devices may differ, so that there may be physical devices produced embedding just one virtual device whereas other physical devices may support several virtual devices. In addition, this concept allows manufacturing network transparent gateways. Those gateways may be used to sub-divide one physical CANopen network into several segments in order to overcome CAN bandwidth limitations or to overcome number of node limitations.

References

[1] CANopen application layer and communication profile (CiA DS 301:2002)
[2] CANopen application profile for passenger information (CiA DSP 407:2002)
[3] CANopen application profile for road construction machinery (CiA DSP 415:2003)
[4] CANopen application profile for building door (CiA DSP 416:2003)
[5] CANopen application profile for lift control systems (CiA DSP 417:2003)
[6] Device profile guideline (IEC 65/314A/NP: 2003)
[7] Industrial automation systems and integration – Open systems application integration framework – Part 1: Generic reference description (ISO 15745-1:2002)
[8] Industrial automation systems and integration – Open systems application integration framework – Part 2: Reference description for ISO 11898 based control systems (ISO 15745-2:2003)

Holger Zeltwanger
CAN in Automation (CiA)
Am Weichselgarten 26
D-91058 Erlangen
Phone +49-9131-69086-0
Fax +49-9131-69086-79
E-mail: headquarters@can-cia.org
Website: www.can-cia.org