CANopen gateway functionality in distributed I/O systems

Heinz Schaffner, Jochen Weiland, Schneider Electric

In Automation Control Equipment, CANopen is more and more emerging because of its flexibility and openness. For larger systems one single CANopen network seems to be too restrictive. Multiple networks are the standard way of implementing such large networks, where CANopen takes the field oriented or machine oriented part. For communication over network borders, gateway functionality has already been defined in CiA309, in CiA400 and CiA446. This paper gives an overview on such functionality used in an I/O system and shows possible further evolution of gateway technology.

1 Introduction

Distributed I/O systems nowadays have to fulfill a lot of requirements such as:

- Covering the whole range of input and output functionality, digital, analog, counters, ...
- adaptable in a flexible way to the peripheral needs
- supporting different field busses by dedicated Network Interface Modules
- extendable for special functions like motor starters, drives, ...
- rugged for harsh environments
- covering some tens of meters for wide-stretched machines

Especially the requirement for extendibility could not be covered with a certain range of I/O modules available in the system. Instead an open system is preferable, which could be complemented by special I/O devices from other systems or third-party suppliers. With this approach nearly every I/O function could be provided.

A precondition for such open systems is an open, internal “backplane” bus such as CANopen.

For some reasons CANopen seems to be oversized for an I/O system, but it has major advantages compared to other system approaches.

The robust and cheap CANopen bus drivers allow a distribution over a reasonable area. This permits to distribute the inputs and outputs in a flexible way close to the machine. As machines are often designed in a modular way, the I/O system could be configured in segmented clusters to reflect those modules.

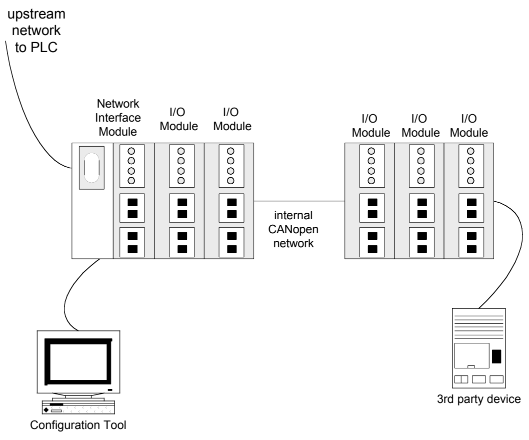

Figure 1: I/O system overview

Since the I/O modules are CANopen devices, they have a microcontroller on board. This provides the opportunity to put some intelligence into the I/O modules, thus releasing the PLC and also saving bus bandwidth on the upstream network.

Simple predefined logical functions, so called reflex actions, can be activated by configuring the system. Another advantage of these reflex actions is, that they may still run even if the PLC is not acting.

Even though the system provides self-configuration for simple applications, most of the applications will need to be configured. In typical applications, user has to set some parameters to dedicated values, for optimizing the I/O behavior or optimizing the internal communication. For this reason a system dedicated configuration tool is provided. Besides the download of the system configuration, including the reflex actions, online diagnostic and test functions are performed.

Users are not forced to use a configuration tool, as the system supports self-configuring. First level of this self-configuring is Auto addressing, where each I/O module gets it unique node ID according to the physical slot position. Next level of self-configuring is the Auto configuration, where the inherent CANopen master/manager scans the whole network for connected devices. The TPDO of the detected devices are linked to the own RPDO of the master/manager and vice versa.

2 CiA309

Configuring the I/O system in fact means configuring the inherent CANopen master/manager. Configured objects have to be downloaded to the Network Interface Module.

There are two ways to download the configuration to the Network Interface Module, over a local serial port or Ethernet in case of Ethernet Network Interface Module.

The Modbus protocol was chosen for this interface, which is a widely used standard protocol for such type of communication. In case of Ethernet network, the Modbus TCP protocol was chosen.

Now the problem arose how to bring the CANopen configuration down to the Network interface Module via Modbus protocol. This was the starting point for the CiA 309 gateway specification.

CiA309 part1, general principles and services: Three classes of gateways are specified.

- Class 1: gateway is a CANopen slave
- Class 2: class1 device plus SDO requesting device functionality
- Class 3: gateway is a CANopen manager

Several services are specified for controlling the state of the network, configuring the network and accessing the CANopen objects.

- SDO access services (i.e. download SDO)
- PDO access services (i.e. configure PDO, write PDO data)
- CANopen NMT services (i.e. start node, reset communication)
- Device failure management services (i.e. read device error)
- CANopen interface config. services (i.e. init gateway, store configuration)
- Gateway management services
- Controller management services
- Manufacturer specific services

With the specified services, the CiA309 allows to control all nodes in the network as well as the gateway itself. The intention of the CiA309 was not to exchange data with the network in a real time fashion, but more to open it for remote control, maintenance and diagnostics.

CiA309 part2, Modbus/TCP mapping: The part 2 specifies the mapping of services to Modbus commands by using the function code FC43/13, known as “CANopen general reference command”.

CiA309 part2, ASCII mapping: The part 3 specifies the mapping of services to ASCII-based communication syntax.

In the described I/O system we face a Class 3 gateway with CANopen manager functionality. The Modbus/TCP mapping of services is used, where only a subset of all defined services is provided. The main services are used for downloading the network configuration to the CANopen manager and for diagnostic purposes. During the download of the configuration, the services for controlling the gateway and the CANopen manager are used.

Relevant used commands are:

- Gateway initialization
- Store – and Restore configuration
- SDO upload and – download
- Enter preop all nodes
- Start all nodes

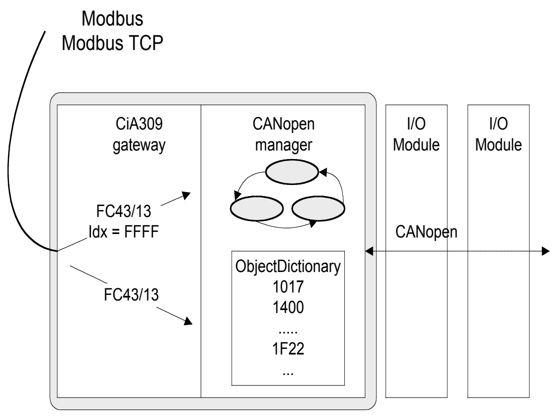

Figure 2: Application of CiA309 in I/O system

For downloading the configuration, the destination address of all FC43/13 commands is the node ID 127, which is the CANopen manager. If the destination index is FFFFh, then the state of the CANopen manager and thus the state of the network is controlled. If the index is unequal to FFFFh, then the object dictionary is accessed.

The CANopen slave nodes are indirectly controlled as they are configured through the configuration manager during boot up.

3 Proposal for CiA309 evolution

The CiA309 today specifies the conversion of Modbus messages to CANopen. A routing of Modbus messages through CANopen is not provided. This is necessary i.e. if a serial Modbus I/O module should be placed in the I/O system. Since this module is connected to CANopen, a CANopen to Modbus conversion has to be provided somehow.

For this reason two dedicated objects could be used:

- Modbus-Request-Object
- Modbus-Response-Object

Both objects are supposed to be of type domain with a length of 256 bytes, which is sufficient for all Modbus frames.

Index /sub	Type	Meaning
5000 ? /0	Domain	Modbus-Request-Object
5001 ? /0	Domain	Modbus-Response-Object

Table 1: Modbus object proposal

As a first approach the index here above is in the manufacturer specific area. A future standardization would choose an index in the profile specific area.

A Modbus request, which should be forwarded through CANopen has to be converted to a SDO download to the Modbus-Request-Object on the addressed device. It is obvious that this requesting device has to have SDO client functionality. If there are other SDO requesting devices in the network, a SDO manager is required or a separate SDO channel has to be used then.

When the SDO reception is indicated on the device, the content of the object will be reconverted to the Modbus request. Sooner or later the Modbus response is put to the Modbus-Response-Object. During this time the client has to cyclically scan the Modbus-Response-Object for the expected response.

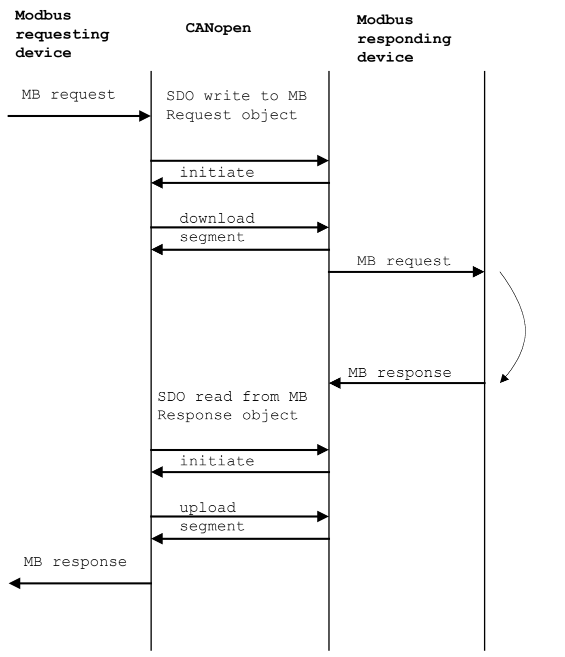

Figure 3: Modbus request over CANopen

This approach would allow forwarding all Modbus commands, particularly FC43/13, which is generally used in CiA309.

The function code 00 indicates the absence of a Modbus message in the Modbus-Request-Object or Modbus-Response-Object. As this, the function code has to be cleared to 00 after confirmation of the according message.

4 CiA400

For connecting the I/O system to CANopen as the upstream field bus, the Network Interface Module has two CANopen interfaces and hence plays the role of a CANopen to CANopen gateway.

For this functionality CiA400 was created. It specifies a multi-level network solution for CANopen. Following a short overview of the CiA400 functionality.

A CANopen to CANopen gateway has up to 32 CANopen network interfaces, so called ports. Each has its own object dictionary, with the mandatory CiA301 entries and some optional entries. Each has its own NMT state machine and SDO server and hence each interface is a CiA301 compliant CANopen device first. The interface may also have NMT master functionality as well as CANopen manager functionality. In this case it shall comply with CiA302. The additional gateway functionality is composed of the functional elements: SDO-, PDO- and emergency forwarding.

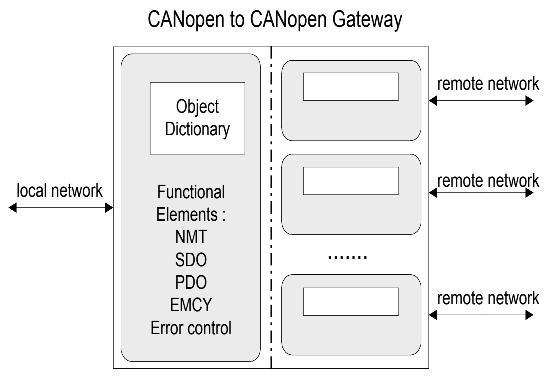

Figure 4: CiA 400 Gateway

There is one local and several remote network interfaces. This is a relative denomination, depending on the standpoint of the viewer and not related to a physical port.

With this kind of gateway a quite complex network could be created. Hierarchical networks, which are controlled top-down, but also non-hierarchical networks are covered by this specification. Non-hierarchical networks may offer multiple message routes from source to destination device over different network paths. Assigning a so-called “cost factor” to each port could configure preferred routes and alternative routes. The preferred route is consequently the route with the least cost factor.

Today most control applications will use a hierarchical network, consisting of few CANopen networks connected with such gateways. Those multiple level networks allow to control the overall network from the top. Even the configuration for the whole network could be downloaded from the top. Of course some parameters on the gateway have to be set to reasonable default values, e.g. the network IDs, node IDs and port numbers.

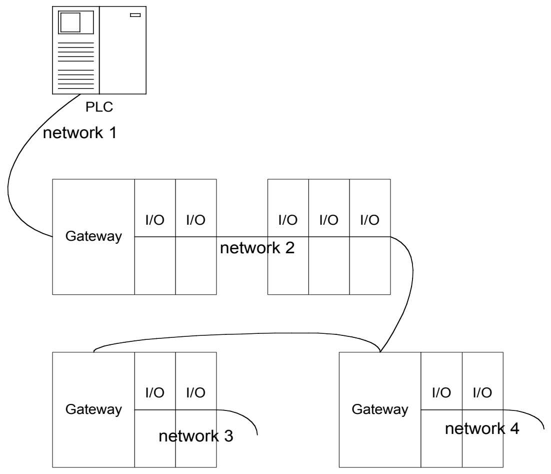

Figure 5: Hierarchical multilevel network

The above figure gives a simple example of a multi level network, even though the CiA400 allows to set up much more complex non-hierarchical networks e.g. with multiple PLCs.

Each CANopen networks gets its unique network ID. The gateway has the knowledge of all connected network IDs as this is configured for each interface, which is the base for the forwarding of messages.

SDO forwarding: SDO forwarding is initiated by a network indication request message sent by the SDO client. A special command specifier is used for this SDO message, which was not yet used. This message carries the target network ID and the target node ID and is forwarded from one gateway to the next until a gateway can identify the targeted device.

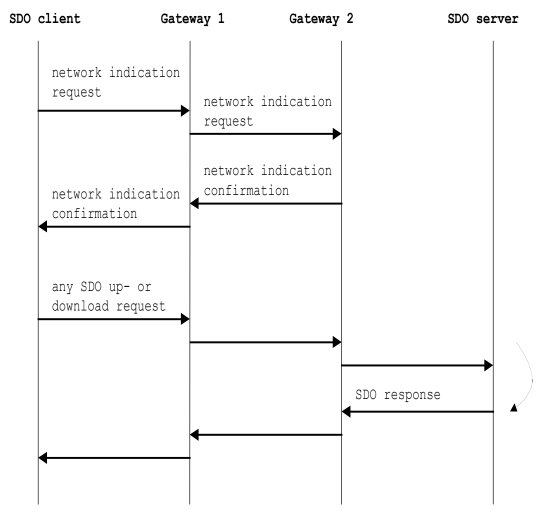
SDO client
Gateway 1
Gateway 2
SDO server
network indication request
network indication request
network indication confirmation
network indication confirmation
any SDO up- or download request
SDO response

Figure 6: Forwarding SDO

This gateway confirms the network indication. The gateways are now prepared to forward the following standard SDO sent by the client and will forward the message accordingly. All standard SDO, expedited, segmented or block upload –or download SDO are allowed. The network indication is valid only for one following SDO access. Subsequent SDO accesses need to be initiated again with a network indication cycle.

NMT forwarding: Controlling the NMT state of a device is only allowed for the NMT master. If a non-NMT master device wants to control the NMT state of another device is has the opportunity to use the Request-NMT (1F82) object of the NMT master according to CiA302. This is also true for controlling of a device in a remote network by achieving a remote SDO read or write request to the NMT master in the target network.

Emergency message forwarding: It is probably impractical to forward every emergency message from the local network to all remote networks because of overcrowding the network especially in exceptional situations. For this reason configurable emergency reception filters and a configurable emergency routing list is specified. By this means the relevant emergency messages could be filtered first and then sent to the network gathering the information.

The forwarded emergency message contains information about the originating node ID and the network ID. Since the message is limited to 8 byte, only the original emergency error code is included in the forwarded emergency message.

PDO forwarding: PDOs are not directly forwarded, because not all objects of a PDO may be relevant for all remote networks. Instead objects are mapped to system variables, which can be mapped to PDO as usual. Those system variables are data type specific, similar to network variables known from CiA405. Since gateway functionality can coexist with IEC61131-3 programmable device functionality, the index range B000$_h$ to BFFF$_h$ was chosen for the system variables.

An object from an RPDO is mapped to a certain index in the system variable area (see the following object dictionary) according to its data type and port number, where it has been received. This variable, indicated by its index, can now be mapped to TPDOs on the other ports, by means of well known PDO mapping parameter. See the following simple example for a 2-port gateway.

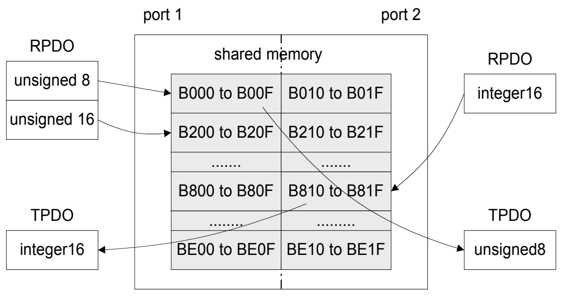

Figure 7: **Forwarding system variables**

Each of the maximum of 32 ports may provide 16 indexes per data type. Each of these indexes may provide 254 sub-indexes. The access attribute of the variables is rww (meaning read, write and mappable to RPDO) for the local port and rwr (meaning read, write and mappable to TPDO) for the remote ports.

Heartbeat forwarding: Heartbeats are not forwarded. The state of a device can be acquired by NMT services. If the state of the devices in a network is of major interest, the state could be observed and kept in manufacturer specific objects, which could also be mapped to PDO.

Specific Object dictionary entries: *1000$_h$ Device type:* profile 400, additional information specifies the gateway functionality

6000_h Lock gateway configuration: protection of the gateway parameters

6001_h Local network ID

6002_h Remote network routing list: indicates the remote networks with its network ID, gateway node ID and port number

6003_h Cost factor: additional information to routing list

6010_h Emergency routing list: indicates to which remote network the emergency shall be forwarded

6011_h to 6031_h Remote emergency reception filter for port 1 to 32: indicates from which device and network emergency messages should be accepted

$B000_h$ to $B1FF_h$ Unsigned8 system variables: 16 variables (arrays) per port

$B200_h$ to $B3FF_h$ Unsigned16 system variables: 16 variables (arrays) per port

$B400_h$ to $B5FF_h$ Unsigned32 system variables: 16 variables (arrays) per port

$B600_h$ to $B7FF_h$ Integer 8 system variables: 16 variables (arrays) per port

$B800_h$ to $B9FF_h$ Integer16 system variables: 16 variables (arrays) per port

$BA00_h$ to $BBFF_h$ Integer32 system variables: 16 variables (arrays) per port

$BC00_h$ to $BCFF_h$ Boolean system variables: 8 variables (arrays) per port

$BD00_h$ to $BDFF_h$ Real 32 system variables: 8 variables (arrays) per port

$BE00_h$ to $BEFF_h$ Unsigned64 system variables: 8 variables (arrays) per port

$BF00_h$ to $BFFF_h$ reserved system variables

Even though few refinements could be done, such as broadcasting of NMT messages and heartbeat messages, the CiA400 specification is ready for implementation. Besides the gateway functionality to be developed, the communication stack of the SDO client needs to be evolved, as this interface is today not prepared for accesses to devices in remote networks.

Additionally a network configuration tool supporting the overall network seems to be valuable especially for complex networks, as the configuration of the single networks one after the other is error prone and might be a challenge for the user.

5 CiA 446

AS-Interface is a very well suited field bus for the sensor/actuator level. It has a lot of advantages on this level, like extremely easy IP67 installation, easy configuration, or auto-addressing for faulty device replacement. AS-Interface is the perfect field bus below a CANopen installation.

There are a lot of installations using both networks in one application to get the most advantages out of both networks. This standard architecture with a CANopen network and underlying AS-Interface networks is addressed by the CiA 446 [5]. This proposal defines an interface profile for CANopen to AS-Interface gateways.

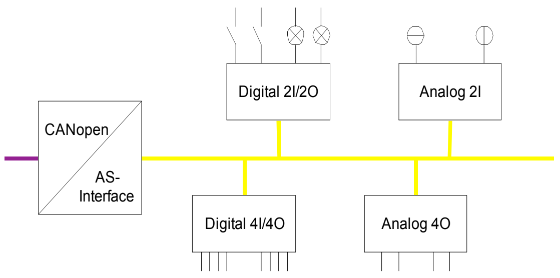

Figure 8: CANopen/AS-Interface gateway

The CANopen/AS-Interface gateway can be seen as a CANopen NMT slave device and an AS-Interface master combined in one device. The AS-Interface master handles the AS-Interface like any other AS-Interface master and scans all slaves as defined by the AS-Interface Complete Specification [6]. The CANopen part of the gateway provides access to the information on the AS-Interface line.

The cyclical refreshed input/output data is available as digital input and output images and analog input and output images. A digital input or output image of a complete AS-Interface bus needs 248 bits. This image fits into 16 octets. A complete analog image can contain as much as 31 slaves with four channels of 16 bit values.

The proposal considers the fact that AS-Interface slaves are typically digital input/output devices and provides a default mapping for the digital input/output image, which is enabled by default (see Figure9 for the default mapping of the digital input image). The digital output image is mapped in a similar way onto the RPDOs.

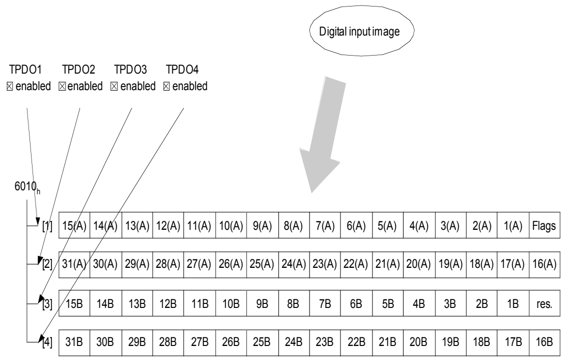
Digital input image
TPDO1 ☒ enabled
TPDO2 ☒ enabled
TPDO3 ☒ enabled
TPDO4 ☒ enabled
6010h
[1] 15(A) 14(A) 13(A) 12(A) 11(A) 10(A) 9(A) 8(A) 7(A) 6(A) 5(A) 4(A) 3(A) 2(A) 1(A) Flags
[2] 31(A) 30(A) 29(A) 28(A) 27(A) 26(A) 25(A) 24(A) 23(A) 22(A) 21(A) 20(A) 19(A) 18(A) 17(A) 16(A)
[3] 15B 14B 13B 12B 11B 10B 9B 8B 7B 6B 5B 4B 3B 2B 1B res.
[4] 31B 30B 29B 28B 27B 26B 25B 24B 23B 22B 21B 20B 19B 18B 17B 16B

Figure 9: Default mapping of digital input image

The analog input/output image is also mapped, but the PDOs are not enabled by default (see Figure 10 for the default mapping of the analog input image). The analog output image is mapped in a similar way onto the RPDOs.

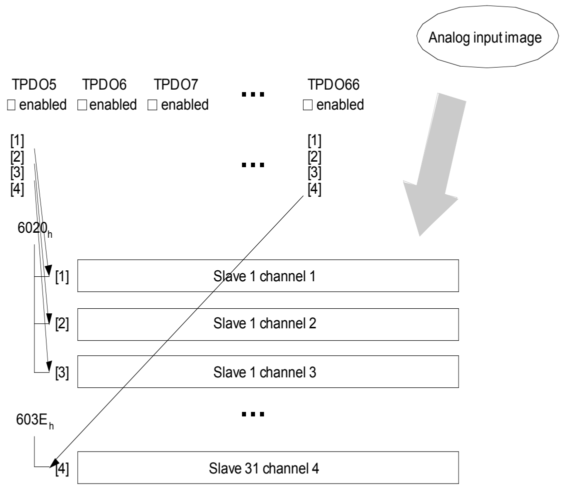

Figure 10: Default mapping of analog input image

This fixed default mapping guaranties the ease of use the user is used to from AS-Interface. Replacing a faulty gateway with any other implementing this interface profile is easy, as the maintenance personal has only to remove the faulty and insert a new one.

A gateway can support up to two AS-Interface networks in one device. The proposal calls these lines "circuit" in conformance to the AS-Interface Complete Specification V3.0 [6]. The second master is handled in the default mapping of the CANopen/AS-Interface gateway like the first or single master.

The main usage of the CANopen/AS-Interface gateway is providing access to the digital and analog image of the sensors on the AS-Interface network for CANopen devices. This is handled by the default mapping of the PDOs. Never the less, CANopen devices may also need to get or set additional information from the AS-Interface network like getting the status of the slaves, reading or writing the parameter bits of individual slaves. The proposal also addresses these needs and provides objects to access these functions of the AS-Interface master (Table 2). The function of circuit 2 is always available at the <object number at circuit 1> + 1. These objects are available with SDOs.

For example, a CANopen device will read the object $61AA_h$ subindex 35_h of the gateway to get the slave configuration (the AS-Interface profile) of slave 15B on the AS-Interface circuit 1. The object contains in an Unsigned16 the ID1 code, ID2 code, I/O code, and ID code from the most to the least significant bit.

Object	Name
6010_h	Digital inputs circuit 1
6011_h	Digital inputs circuit 2
6020_h to $603E_h$	Analog inputs circuit 1 – slave 1 to 31
$603F_h$ to $605D_h$	Analog inputs circuit 2 – slave 1 to 31
$605E_h$	Digital outputs circuit 1
$605F_h$	Digital outputs circuit 2
6060_h to $607E_h$	Analog outputs circuit 1 – slave 1 to 31
$607F_h$ to $609D_h$	Analog outputs circuit 2 – slave 1 to 31
$619E_h$	AS-Interface master flags circuit 1
$61A0_h$	List of active slaves (LAS) circuit 1
$61A2_h$	List of detected slaves (LDS) circuit 1
$61A4_h$	List of periphery faults (LPF) circuit 1
$61A6_h$	AS-Interface master control word circuit 1
$61A8_h$	AS-Interface master status word circuit 1
$61AA_h$	Read actual AS-Interface

	slave configurations circuit 1
61AC$_h$	Write permanent configuration AS-Interface slaves circuit 1
61AE$_h$	Read permanent configuration AS-Interface slaves circuit 1
61B0$_h$	Write list of configured AS-Interface slaves circuit 1
61B2$_h$	List of configured AS-Interface slaves circuit 1
61B4$_h$	Read acyclic request circuit 1
61B6$_h$	Read acyclic response circuit 1
61B8$_h$	Write acyclic request circuit 1
61BA$_h$	Write acyclic response circuit 1

Table 2: Additional CANopen/AS-Interface gateway objects

AS-Interface slaves may implement an AS-Interface device profile, which is called “combi slave profile” starting with the AS-Interface Complete Specification Version 3.0 [6]. The combi slaves have one part, which is mapped into the digital input/output image. The other part allows the transfer of data strings. The proposed interface profile also handles these data strings with the objects 61B4$_h$ to 61BB$_h$.

The proposal provides a method to integrate CANopen and AS-Interface in the sense of the AS-Interface philosophy: “simple and easy to use”.

6 Conclusion

With CANopen a versatile network for distributed intelligent I/O systems is available. The gateway specifications CiA309 and CiA400 establish a base for integration of CANopen into heterogeneous or homogenous networks.

References

[1] CiA DS 301, CANopen application layer and communication profile

[2] CiA DSP 302, Framework for CANopen managers and programmable CANopen devices

[3] CiA DSP 309, Interfacing CANopen with TCP/IP, Part 1 General principles, Part 2 TCP/Modbus mapping

[4] CiA DSP 400, CANopen Interface profile - Multi-level networking

[5] CiA WDP 446, CANopen Interface profile for AS-i gateways

[6] AS-International, AS-Interface Complete Specification, Version 3.0