Standardized higher-layer protocols for different purposes

Holger Zeltwanger, CAN in Automation (CiA) e. V.

As the CAN standards specified in the ISO 11898 series cover just the lower layers (*physical* and *data link* layer) of the OSI reference model, the network system designer has to deal additionally with the functionality of the higher-layer protocols (from the *network* to the *application* layer). In many CAN applications, just the *application layer* functions need to be implemented. From the beginning, there was some standardization of CAN-based higher-layer protocols requested, in order to save software investments by means of reusing programs and routines developed for different applications. The paper provides an overview of "open issues" to be considered and solved by the higher-layer protocols, and discusses the different solutions introduced by standardized *application layers*. It is not intended to compare the standardized solutions but to describe the different approaches in respect to system design requirements.

The CAN data link layer is one of the most reliable communication protocols. But it leaves some necessary functions to the user. This includes for example the detection of nodes necessary for a dedicated application. Another missing function is the segmentation of payloads larger than eight bytes and the reassembling of segments in the receiving nodes.

Of course, there are different requirements on these higher-layer protocol functions to be provided by so-called higher-layer protocols such as transport layer or application layer protocols. Often system designers demand to standardize the user data (commands, status, measured values, requests, configuration parameter, etc.). Of course, this is outside of the 7-layer OSI (open system interconnection) reference model, but is necessary to achieve interoperable and interchangeable products.

The standardization of higher-layer protocols and even profiles is just one side of the coin. The other side is the demand of easy-to-use system design and configuration tools. Of course, standardized higher-layer protocols help to develop such tools. The tools provided by the CAN chipmakers don't meet the system designer's requirements. They are suitable to design a device including the low-level driver software. But they don't support the development of segmented data transfer protocols, for example.

The following chapters will discuss the "open" issues to be covered by higher-layer protocols, and will provide a brief history of higher-layer protocols.

Detecting missing nodes

When all messages are event-triggered by means of Change-of-State (CoS) events, it is necessary to detect nodes that are in bus-off state. In CANopen, this was done by the Node Guarding function. It is based on a master/slave relation meaning that the NMT (network management master) remotely requested the status of the NMT slave devices. Using CAN remote frames has several disadvantages (for details see the CiA 801 application note). The NMT slaves also need a time-out mechanism (Life Guarding) to detect the absence of the NMT master. Nowadays, in CANopen the Heartbeat function is recommended for detecting missing nodes. DeviceNet has used from the beginning a similar Heartbeat function.

In J1939-based networks, a Heartbeat function is not necessary, because all unconfirmed messages are transmitted periodically. So, each message provides implicitly an alive-information. This wastes bandwidth, if data is not changing. Most of the carmakers also transmit all CAN messages periodically. To be serious, also in many non-automotive CAN networks all real-time data is transmitted periodically.

The explicit and implicit alive-information is not only necessary to detect bus-off or disconnected nodes, but also nodes, which are in error-passive mode at high

busloads. When a node in error-passive mode produces a passive error flag and the bus is not idle, the passive error flag can't be completed and the node is therefore not able to receive or to transmit messages.

Node management

Self-starting nodes don't need to be configured. In many systems with self-starting nodes, no dedicated node management is implemented. In CAN systems with dedicated node management, normally one device is responsible for the network boot-up process including the configuration of nodes as well as the starting and stopping of nodes. In CANopen, the NMT master is responsible for the node management. In order to overcome a single-point of failure, CANopen optionally provides the "Flying" NMT master protocols. They are used to manage the enabling of hot stand-by NMT master devices and the negotiation process, which NMT master capable device is the active NMT master. Normally, the node management requires a confirmation on the application level (see also below). In CANopen, this performed by means of the Heartbeat protocol.

Overcoming the "addressing" limits

Originally, the CAN protocol specified just the 11-bit identifier, which allows to distinguish between 2048 messages. For more complex networks this is not sufficient. One solution is the prolongation of the CAN-ID. That is why the extended frame format with 29-bit IDs has been introduced. It is used for example by all J1939-based networks. The disadvantages are the longer bus latency times (about 20 bit-times) and the higher busload (about 20% and more, depending on the data-length). Also the CRC performance is lower due the longer frame length (for details see /CHAR/).
The other possibility to address more parameters is, the usage of a part of the data-field as multiplexor. This approach has been chosen for CANopen and DeviceNet, for example. Usually, you use one byte for protocol control purposes and three other bytes for the multiplexer. In CANopen it is the 16-bit index and the 8-bit sub-index. In DeviceNet a more object-oriented method has been used: addressing by means of class, instance, attribute (24-bit address). Also in some automotive diagnostics protocols, multiplexors are used (e.g. Unified Diagnostic Services).
The disadvantage is the limitation to less than 8-byte parameter. On the other hand, the resource of CAN-IDs is saved. In addition, the bus latency time is not prolonged compared to the extended frame format using 29-bit CAN-IDs.

Transport protocols for longer data

Parameters that exceed the 8-byte length of the data field need to be segmented by the transmitter and re-assembled by the receivers. In general, all transport protocols (TP) are similar, but in the details are different. Most of them provide some flow control meaning that there is some confirmation of the received segments. Sometimes each segment is confirmed; in other TPs there is a number of segments confirmed (block transfer). Normally, there are positive and negative acknowledgments (e.g. abort message). Most of the TPs have some limitations regarding the length of the parameter, because the use unique numbers for each segment.

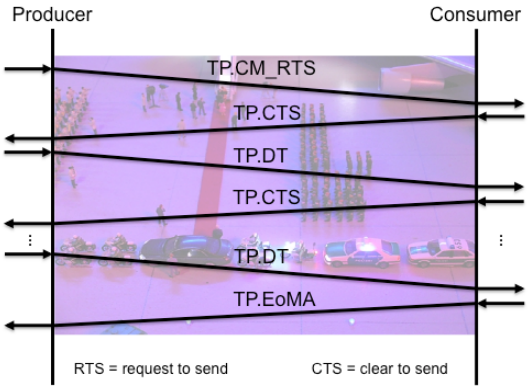

Figure 1: RTS/CTS transport protocol of J1939 with a maximum length of 1792 byte with a CTS time-out of 1250 ms

In case of unlimited length, a toggle-bit is used, in order to detect a doubled received CAN message. A double reception of CAN messages could happen, when the last bit of end of frame (EOF) is dominant. This is interpreted differently by the transmitting and receiving nodes. The transmitter re-

gards this as an error condition (expecting that a receiver has evaluated locally a dominant sixth bit of EOF). The receiving nodes regards a dominant bit in the last bit of EOF as an overload condition, because they have already accepted the message as correct with the sixth bit of EOF, if recessive.

Some of the transport protocols confirm each segment, while others confirms a block of segments. There are protocols limiting the length of the segmented transfer (the SDO protocols of CANopen don't limit the size of segmented down- und uploads). In most these confirmed protocols specify a time-out for the confirmation (positive or negative acknowledgement), but the SDO time-out is manufacturer-specific.

Conformation on different levels

The CAN data link protocol provides just a confirmation by means of the ACK slot bit. This is a confirmation for the transmitting node that it is not alone in the network. It doesn't indicate, if the CAN frame has been received by all nodes that are configured to receive this message.
The above-mentioned confirmation of segmented messages is still not confirming that the message content has been correctly processed. It is still just a confirmation for the transport protocol that this segment has been correctly received.
The confirmation on the application level is in J1939-based networks provided by request messages and status messages. This approach is also used in the carmaker-specific CAN protocols (message matrix). Similar mechanisms are specified in the motion profiles for CANopen and DeviceNet: The host controller sends a command-word, which is confirmed by the motion controller by means of the status-word. Both are mapped in different CAN messages and evaluated by the host controller.

Configuration and program download

In many applications, the CAN network is not only used for control purposes, but also for the configuration of the nodes and the download of programs. Of course, this requires a transport protocol. For open networks, standardized internal addresses are required. If several programs are downloadable, standardized functions to start and stop programs are needed. In CANopen up to 254 internal addresses are standardized for programs to be started and stopped via CAN. There is additional information about the download date and time available. This allows using configuration and programming tools from different suppliers.

Standardized scheduling of real-time messages

The scheduling of messages containing commands and status information is one of the most important tasks of the system designer. The periodical transmission of messages is still one of the most used scheduling modes. In order to save bandwidth, system designer may use an event-trigger (change-of-state) transmission mode. CANopen also provides a combination of both methods: The message is send periodically, if no mapped parameter changes; but when the parameter changes, the messages is transmitted immediately.

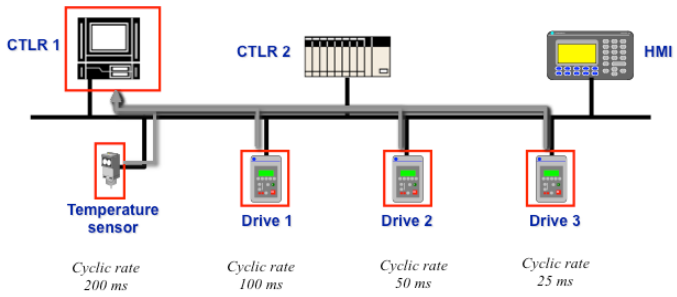

Figure 2: In many CAN networks the real-time data is transmitted periodically (in the shown DeviceNet application periodical transmission is called "cyclic", which is misleading, because the cycle derives from the individual local timer in each device)

In some applications, coordinated and synchronized actions in different nodes are necessary. For example, different sensors should sample at the very same moment their measured values and transmit them in different messages. It might be

also required to synchronize actuations in different nodes. For such applications, CANopen provides the SYNC message [3], which triggers the simultaneous capturing of sensor values or the validation of previously received set-values (e.g. in multi-axis motion control systems). The maximum jitter of the periodically transmitted SYNC message is about one CAN data frame. The synchronization of actions is also achievable by a global network time. In CANopen this is possible by means of the TIME message with accuracy of 1 ms. Both methods have the disadvantage that the synchronization may be lost due to the automatic retransmission of corrupted messages.

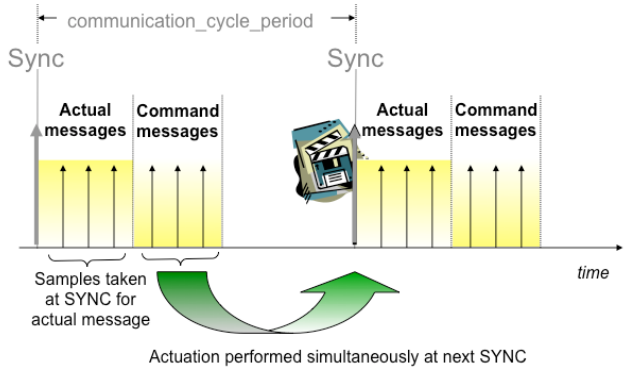

Figure 2: Synchronous sampling and actuating in an asynchronous network by means of a SYNC message

The TTCAN protocol (ISO 11898-4) avoids this disadvantage. It uses Reference messages comparable with the SYNC message, but for the following messages the automatic retransmission is disabled [2]. However, TTCAN has not been adopted in any “open” higher-layer protocol.

Prioritization of functions and pre-assigned CAN-IDs

In embedded networks, the system designer assigns the priority for each message to the application requirements. In open networks, the CAN-ID contains often a unique node-identification. In J1939-based networks, the CAN-ID identifies the content of the messages and a 3-bit priority. In DeviceNet and CANopen, the pre-assigned CAN-ID identifies the kind of communication service (e.g. I/O message or explicit message resp. PDO or SDO) and derives also from the unique node-identification assigned by the system designer.

Bit-timing standardization

Higher-layer protocols for open networks need also to determine the bit-timing. CANopen, DeviceNet, and the different J1939-based networks standardize the allowed bit-rates and the sample-methods (single or multiple sampling) as well as sample-point. The automotive industry has also standardized in the SAE J2284 series the bit-timing for CAN high-speed networks compliant to ISO 11898-2. The only one, which allows data-rates up to 1 Mbit/s is CANopen, the other restrict the maximum bit-rate to 250 kbit/s respectively to 500 kbit/s. Also on the lower end, only CANopen supports data-rates up to 10 kbit/s (note: not all transceiver chips on the market support those low bit-rates).
Most of the higher-layer protocols for open networks also specify the pin-assignment for connectors and sometimes even for the cables to be used. DeviceNet is for example more restrict than CANopen, which limits the usage of DeviceNet for industrial automation. The CANopen recommendations regarding the physical layer are much much more flexible. On the other hand, the “harder” specification reduces malfunction due to “badly” designed CAN networks.
The “most hardest” CAN physical layer specifications (optimized in respect to robustness and cost) are used in passenger cars. The carmakers have spent much effort to design proper CAN in-vehicle networks. Their approaches are dedicated for this application and can’t be used generically.
Some industrial CAN users don’t care on the price for the CAN physical layer components such as cable and connector. Of course, they minimize the disturbance possibilities and maximize the robustness regarding EMC (electro-magnetic compatibility).

Node claiming and layer setting protocols

In particular in open network environments, it is a requirement to configure the bitrate and a unique node-identification. If

this should be done via the CAN interface, there are special protocols necessary.
In J1939-based networks, there is the NAME claiming procedure. It "violates" the design-rule that a CAN message with variable content must not be sent by two or more nodes. The CANopen claiming procedure described in CiA 416 avoids this design-rule violation by using a worldwide unique identification (Identity parameter 1018_h). This unique 128-bit device identification is also used for the CiA 305 layer setting services and protocols (LSS). With LSS, CANopen node-IDs and bit-rates are changeable in a running system.
The boot-loader and program download functionality as well as the control of application software (e.g. starting and stopping) are also features that are not covered by the 7-layer OSI (open system interconnection) reference model internationally standardized by ISO. I would regard them also as layer management functions.

Standardization of application functions

To standardize the communication services and protocols is not sufficient to achieve interoperability between devices. To design interoperable devices requires additional standardization of the application functions – so-to-say the content of the messages (could be process data, configuration parameter, or diagnostic information). In general, there are two approaches:

- The system-oriented approach standardizes all messages transmitted in a dedicated application. In which device, which application function is implemented, doesn't matter: The system designer has to take care on system consistency by means of selecting the "right" devices.
- The device-oriented approach specifies individual communication interfaces, and the system designer needs to program or configure the consistent communication in one or more control units.

Specifications using the system-oriented approach, I like to call application profiles. Interface descriptions compliant to the device-oriented approach are normally named as device or interface profile. J1939-based networks are specified in application profiles (unfortunately they are named application layer – but this is not correct). DeviceNet uses device profiles to define the process and the configuration parameter. In CANopen both approaches are used. There are generic (e.g. for I/O) and industry-specific (e.g. for corrugator in extruder machinery) CANopen device profiles; and there are CANopen application profiles for dedicated applications (e.g. CANopen-Lift, C*le*ANopen, and special-purpose cars).

Additional network functionalities

In some applications, additional network functionality is requested, which is not covered by the OSI reference model. This includes protocols for safety-related transmission of data as specified in CANopen-Safety or CIP-Safety for DeviceNet. Other protocol functions are data security and device redundancy (e.g. Flying NMT master protocols for CANopen). Also protocols dealing with bus-line redundancy, and the related swapping mechanisms of bus-lines fall into this category of protocols.

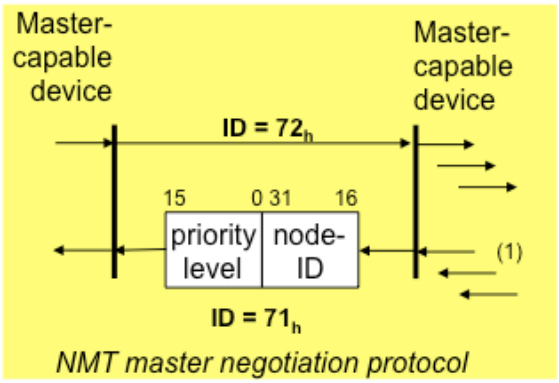

Figure 4: One of the Flying NMT master protocols negotiating the NMT master-capable device with the highest priority

For CANopen a specific bus-line redundancy for maritime electronics [?] has been developed (CiA 302-6). Other protocols requested by complex network architectures include router and gateway functionality. In CANopen the SDO and EMCY router function is specified in CiA 302-7 (multi-level networking). For diagnostic in passenger cars the ISO 15765-2 (transport and network layer services) standard includes the gateway functionality.

Brief history of CAN-based higher-layer protocols and profiles

All standardized communication protocols should follow the OSI reference model. It was introduced in 1984 and is internationally standardized by ISO 7498-1. The OSI model defines internetworking in terms of a vertical stack of seven layers. The upper layers of the OSI model represent software that implements network services like encryption and connection management. The lower layers of the OSI model implement more primitive, hardware-oriented functions like routing, addressing, and flow control.

This 7-layer architecture doesn't cover the content of the data and doesn't specify the communication with system design tools. However, the standardization of the application data and the tool communication interfaces are essential for interoperability and exchangeability of networked devices.

In the CAN community, the Swedish company Kvaser was one of the first standardizing the communication interface for system design tools: In its CAN-Kingdom specification (developed in the late 80ies and early 90ies) the "King" was the system design tool. However, CAN-Kingdom didn't specify the application layer defined in the OSI reference model [1].

CAN-Kingdom "just" provides standardized communication mechanism to design an application layer by means of a standardized device configuration. So-to-say, it is a layer management protocol and not an application layer. All CAN-Kingdom compliant devices support the communication with the "King". CAN-Kingdom provides the following functions:

- The "King" configures, which nodes will receive and transmit which messages.
- It provides an infrastructure for transmitting fixed format data that runs over a single CAN packet payload length. In CAN Kingdom parlance this is known as "document pagination".
- It allows point-to-point transfer of data streams, through the "Block Transfer" mechanism.
- It allows runtime mapping of CAN identifiers with an optional second level of indirection. For the first level, the "King" must map himself the CAN-ID from him to the "folder". The second level, "folder" to "document", can optionally be fixed by the node designer.
- It provides a mechanism for clock synchronization.
- It provides a specification for packed bit fields.
- It provides event driven, "daisy chain", and synchronous messaging.
- It provides a mechanism to set message filters.

In the early days of CAN, there were also developed the first higher-layer protocols based on the OSI reference model. One of the very early solutions was the CAN Application Layer (CAL) by CiA. It was based on ideas developed in Philips Medical Systems. Other CiA members also contributed some ideas and functions. It was a "pure" layer-7 protocol not specifying any message content.

In order to satisfy the requirements regarding standardized data content, several industries started in 1993 and the following years to specify CAN-based higher-layer protocols transmitting standardized data. Members of the SAE (Society of Automotive Engineers) association developed the J1939 set of specifications. Originally, this standard was dedicated for powertrain applications in trucks and buses. It includes transport protocols for data longer than 8 byte (e.g. RTS/CTS and BAM with maximum length of 1792 byte), but specified mainly 8-byte messages identified by the PGN (parameter group number) as part of the 29-bit CAN-ID. Each message has an 8-byte data field containing one or more parameters (equivalent to signals or to process data). Not used bits are reserved. These messages are transmitted periodically. The period is determined by the SAE J1939-71 or equivalent specifications and is not configurable. Nowadays, the J1939 protocols have been adapted to other applications, too:

- ISO 11992-2/3—Truck/trailer communication (but only the network for braking and running gear equipment as defined in part 2 has been implemented)
- ISO 11783 series (also known as Isobus)—Tractor/implement communication in agriculture and forestry vehicles
- IEC 61162-3—network for marine navigation and radio communications

equipment aboard all classes of vessels including Solas (Safety of Life at Sea) vessels.

However, the transport protocol and the application profiles are slightly different. But the structure of the CAN-ID usage is the same as well as the periodical transmission of parameter group messages. Of course, the application profiles will be still enhanced.

In the early 90ties, Allen-Bradley and Honeywell Microswitch developed in the beginning jointly with American industrial users, the idea of a CAN-based network for factory automation. After a while, both companies went their own ways. This resulted in two different higher-layer protocols: DeviceNet and Smart Distributed System (SDS). Both have been internationally standardized in the IEC 62026 series. A few years ago, SDS has been withdrawn due to missing support by the industry. DeviceNet, nowadays supported by the ODVA (Open DeviceNet vendor association) is completely integrated into the CIP (Common Industrial Protocol) application layer and device profile approach.

In Europe, the CANopen application layer and its profiles have been pre-developed within a European research project. End of 1994, it was handed-over to CiA for maintenance and further developments. The CANopen application layer is internationally standardized in EN 50325-4. The CAN-Safety protocol is standardized in EN 50325-5.

The CiA 402 CANopen profile (for drives and motion controllers) is also internationally standardized in the IEC 61800-7 series, in which also the CIP motion profile is standardized. CiA has also submitted other CANopen profiles for international standardization (e.g. CiA 422: C*le*ANopen, and CiA 443: SIIS level-2 devices). The usage of CAN networks in rail-vehicles is standardized in IEC 61375-3-3. The EN 13149-4/5/6 technical reports specify the CANopen application profile for passenger information systems in public transportation.

The family of CANopen specifications also includes program download functionality, (CiA 302-3), configuration manager capability (CiA 302-3) as well as different other layer management functions (e.g. CiA 305).

The carmakers developed within the OSEK project the OSEK-COM application layer. It was even internationally standardized in the ISO 17356 series, but didn't get much support by the automotive industry; meaning it was not really implemented. Carmakers are using still proprietary application layers and profiles (also known as communication matrix).

For diagnostics, the automotive industry has standardized the communication. Even if it sometimes confusing, the unified diagnostic services on CAN (ISO 14229-3) and the sub-layered services and protocols (ISO 14229-2: UDS session layer, and ISO 15765-2: Transport and network layer) are completely standardized and used. The carmakers are using also the quasi standardized CAN Calibration Protocol (CCP) and its successor ECP (Extended Calibration Protocol). With them ECUs can be calibrated during the production and integration phase. All of these protocols use some multiplexer in the data field, in order to save the resource of CAN-IDs.

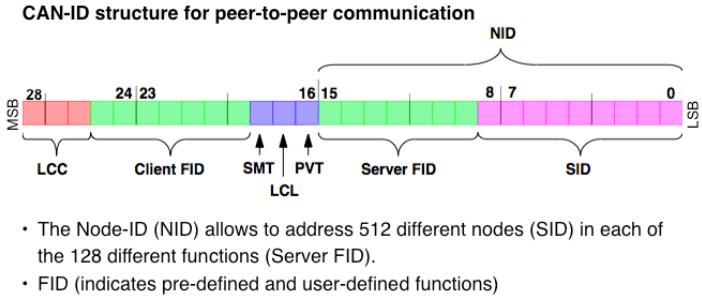

Figure 5: The shown Arinc 825 peer-to-peer protocol allows individual nodes to establish client/server type interactions, called node services; both connectionless as well as connection-oriented communication (compare to UDP/IP versus TCP/IP) is supported

The latest CAN-based higher-layer protocol standardization has been achieved by the aircraft industry. Within the Arinc association, the Arinc 825/6 specifications have been released. These specifications meet specifically the requirements of airborne applications. It is adapted some ideas of the CANaerospace protocol developed in the mid 90ties. For details see [4] in this proceedings.

Summary

Most of the “survived” higher-layer protocols have some similarities. The differences between them are more or less depending on the roots and on the development history. This is similar to human languages: The English and most of the other European languages using not just the same set of characters, but also a similar set of grammar rules.
This means, it doesn’t make a technical sense to develop an additional higher-layer protocol for CAN-based systems. (As, it makes no sense to create a new human language). The application-specific requirements can be solved in the specific profile specifications. For new designs of CAN-based “open” (standardized) networks without historical compatibility requirements, CANopen is a suitable compromise. It provides the most additional functions. The automotive industry has selected CANopen as the base of an open network for special-purpose cars (CiA 447 application profile for car add-on devices).
All existing higher-layer protocols and related profiles suffer sometimes on the limited length of the CAN data-field. If this would be overcome, this would prolong the lifetime of CAN-based networks significantly. The proposed CAN-FD data link layer is an interesting opportunity. For details, see [5] and [6] in this proceedings.

References

[1] **K. Lennartsson:** Servicing a CAN-Kingdom system (in: iCC proceedings 1997)
[2] **Th. Führer**, and others: Time-triggered communication on CAN (in: iCC proceedings 2000)
[3] **M. Rostan, J. Langfermann:** High-precision drive synchronization with CANopen (in: iCC proceedings 2002)
[4] **R. Knüppel:** Standardization of CAN networks for airborne use through Arinc 825 (in: iCC proceedings 2012)
[5] **F. Hartwich:** CAN with flexible data-rate (in: iCC proceedings 2012)
[6] **H.-J. Oertel:** Using CAN with flexible data-rate in CANopen systems (in: iCC proceedings 2012)

All iCC proceedings are published by CiA in Nuremberg (Germany).

Holger Zeltwanger
CAN in Automation e. V.
Kontumazgarten 3
DE-90429 Nuremberg
Tel. +49-911-928819-0
Fax +49-911-928819-79
headquarters@can-cia.org
www.can-cia.org