Prof. Dr.-Ing. Gerhard Gruhler, Dipl.-Ing. Martin Rostan
STA Reutlingen, Germany

Interoperable Automation: Components Using CANopen Profiles

Abstract

Since the number of available automation components with CAN interfaces is increasing more and more, there is a strong demand on interoperability between these components in multi vendor systems. However, the CAN in Automation community is still using a wide range of manufacturer specific communication solutions. In order to achieve interoperability of control components, communication and device profiles are to be employed together with the CAN communication layers that form the basis for specific implementations.

CANopen, a set of existing and emerging profiles based on CAN Application Layer (CAL) is presented. These profiles are open to manufacturers and users. The CAL based Communication Profile For Industrial Applications (CiA standard DS 301) allows the definition of a wide range of device profiles e. g. for decentralised I/O, drives, vision systems, encoders, etc. The communication profile which is presented in detail provides fast event driven or cyclic messages as well as asynchronous data transfer. Since several companies have already adopted CANopen, an overview is given on ongoing implementations.

1 Introduction

Fig. 1 shows an example of a CAN based production cell. CAN is used as the communication system between automation components of the cell such as the controller, robot drives, operator interface, vision system, feeders and parts of a workpiece carrier transfer system.
Similar production units linked by material flow systems can be found on nearly every production line, especially in the electronic, electromechanical and mechanical device production areas. The cell of Fig. 1 is a typical example for multi-vendor production equipment.
The adaptation of production equipment to customer specific configurations requires enormous and ever increasing engineering and integration effort. This is due to decreasing product life cycles, extension of just-in-time delivery, increasing complexity and variety of products. For the example of a production cell given in Fig. 1, customer specific implementations of interfaces between control components and interface related software normally cause an engineering effort as high as the costs of the basic cell components.

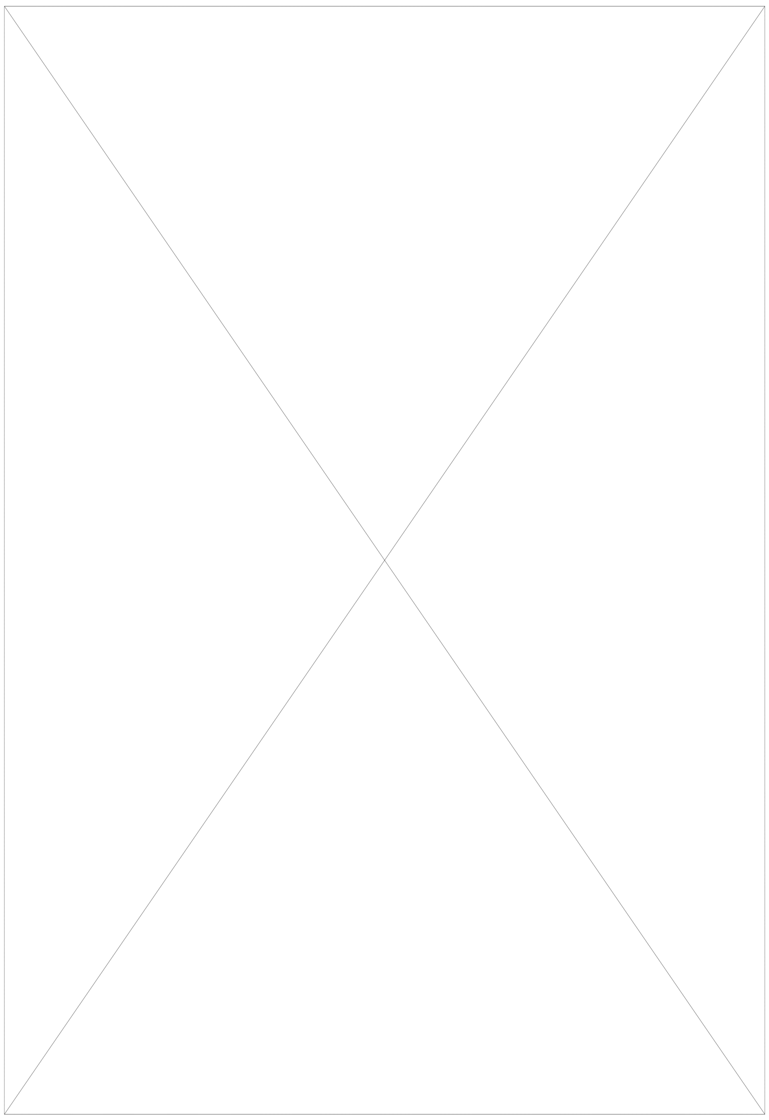


Figure 1: CAN based multi-vendor production cell (CANopen Pilot Cell)

Therefore the application of an open communication system should aim at two major problems: (1) interoperability between the basic functional elements of production cells has to be achieved, (2) traditional cell components usually directly connected to controllers do not support easy and fast cell set-up procedures due to missing installation and configuration flexibility.

2 Open communication with CAN

The use of CAN in automation components often implies the development of application specific parts of protocol software. When a new application is designed, at least a new proprietary "layer 8" specification (profile) is often invented. This might be a satisfactory solution for a certain period of time but the disadvantages become obvious. There is an enormous amount of effort to spend making CAN components completely interoperable. There are lots of protocols and more software versions which have to be supported and maintained. This creates costs which the customers won't pay for ever.

Fig. 2 shows the technical and commercial benefits which can be expected from open CAN communication using respective interoperable CAN based automation components. As a result, a feed forward effect will lead to an increasing number of manufactured components, to lower costs per part and a bigger market share. Additionally, it seems to be better for customers and manufacturers to protect the market share of a device by excellent functionality and a good price-performance-ratio rather than by the use of proprietary protocols.

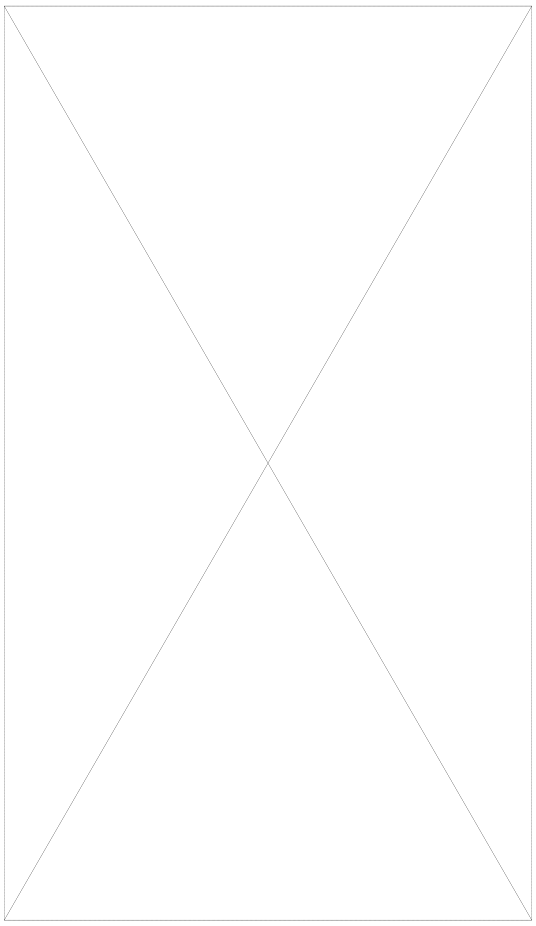


Fig. 2: Expected benefits achieved by open communication

To achieve the indicated benefits at least partially, it is not sufficient to just make public a set of specifications. A really open CAN communication system for automation applications should meet some more requirements. To achieve a high degree of interoperability, open possibilities of a layer 7 protocol like CAL are to be specified by profiles. Fig. 3 shows an extract of a requirement list containing both technical and strategic aspects.

A communication system is considered to be "open" if the *chances* to draw commercial benefit out of both master and slave implementations are similar to all. This is especially important for small and medium-sized enterprises as they have a big share in the CAN in automation market.

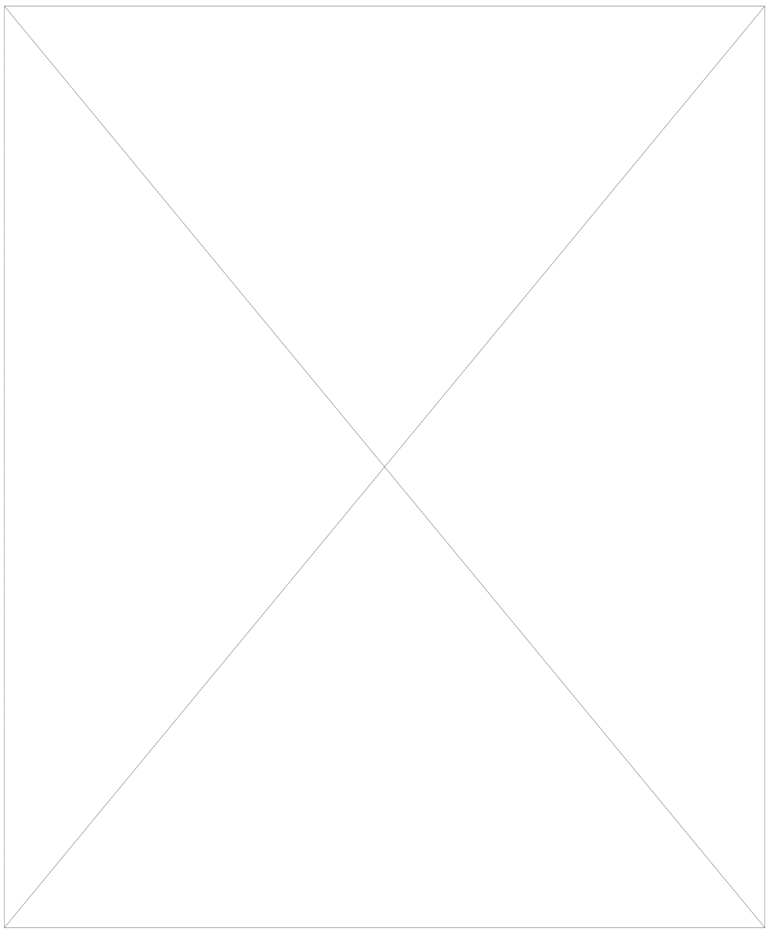


Fig. 3: Open communication is to meet technical and strategic requirements

3 CANopen Communication Profile

3.1 CANopen and CAL

CAN Application Layer (CAL) was the first available open application layer specification for CAN, and many users expected to get the benefits described above by simply using CAL. However, whilst CAL specifies a variety of data objects and services, it does not intend to specify the exact use of these services, but provides all elements for designing CAN communication applications.

One can compare CAL with a well equipped toolbox without a user manual that details which tool one has to use in order to solve a specific problem (see Fig. 4). If for example, a parameter set has to be downloaded to a device, the entire set can be transmitted using domain transfer services, or one can define each parameter to be a variable which is downloaded with a write_variable service. Alternatively, it is possible to use multiplexed variables with confirmed or unconfirmed services, NMT configuration control services, combine single parameters to structures with different access type, use various variable names and priorities, etc..

All possibilities are fully CAL compatible, but obviously not interoperable unless someone specifies which object and service type has to be used for which parameter, and how this parameter is to be interpreted.

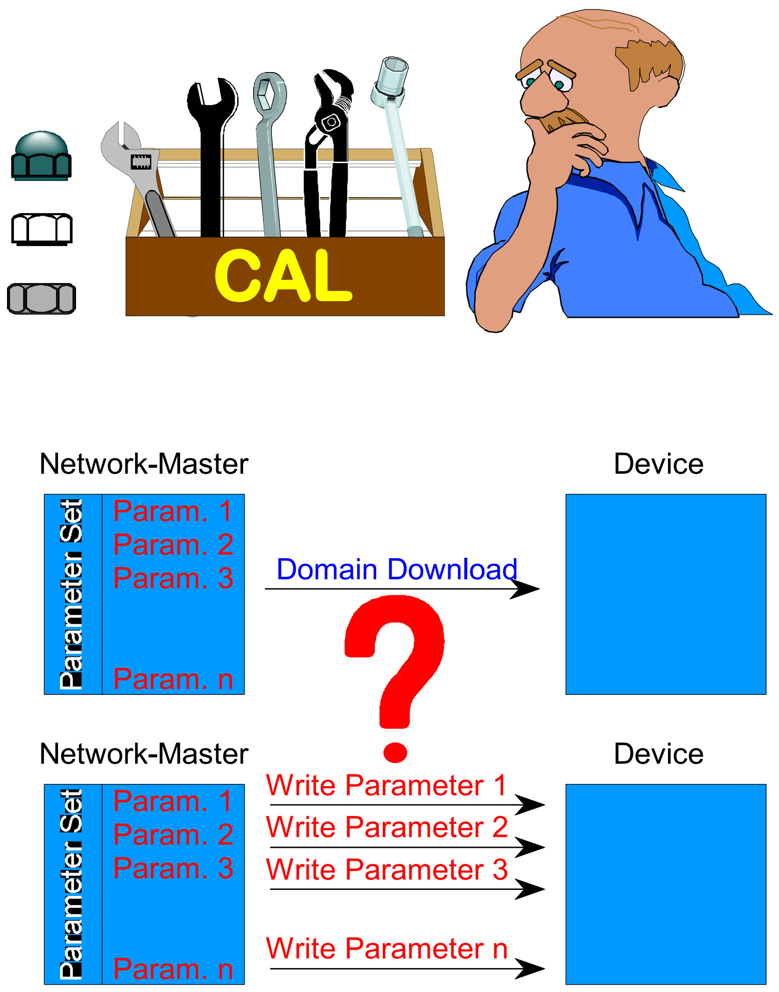


Figure 4: Purpose of communication profile

By defining the subset and use of CAL, the CANopen “CAL based communication profile for industrial systems” (CiA-DS 301) provides the missing user manual that is needed to establish open and interoperable communication with CAL. Or, in other words, CANopen reduces CALs degrees of freedom in order to achieve interoperability, lean implementations and superior performance. This is comparable with Profibus, where Profibus-DP represents a successful real-time subset of Profibus-FMS, which is too complex for most applications. However, although similar in leanness, CANopen provides significantly more functionality than Profibus-DP.

All devices following the CANopen communication profile can interact perfectly in the same physical network (if required together with generic CAL devices). Full interoperability regarding data content is achieved by employing the appropriate *device profile*. The communication profile describes *how* to communicate, the device profiles detail *what* to communicate for each type of device (see fig. 5).

3.2 SDO and PDO

Two data types with different characteristics are dominating in most automation system networks: one type is the real time data, which has to be transmitted quickly, preferably without any overhead, and with pre-defined structure. This process data is either transmitted in a cyclic, synchronous manner or asynchronously, event driven (taking advantage of CANs unique features: transmitting process image changes rather than the entire process image). Typical data content is I/O data or command/actual values for drives. An explicit confirmation of each process data is generally not required. A message of this data type is called Process Data Object (PDO), and CANopen uses the CAL event-service to transmit PDOs. The event service principally describes a CAN layer 2 message as it carries no overhead. CANs broadcasting features as well as its multi master features remain fully intact. PDOs get high priority identifiers in order to ensure their real time behaviour.

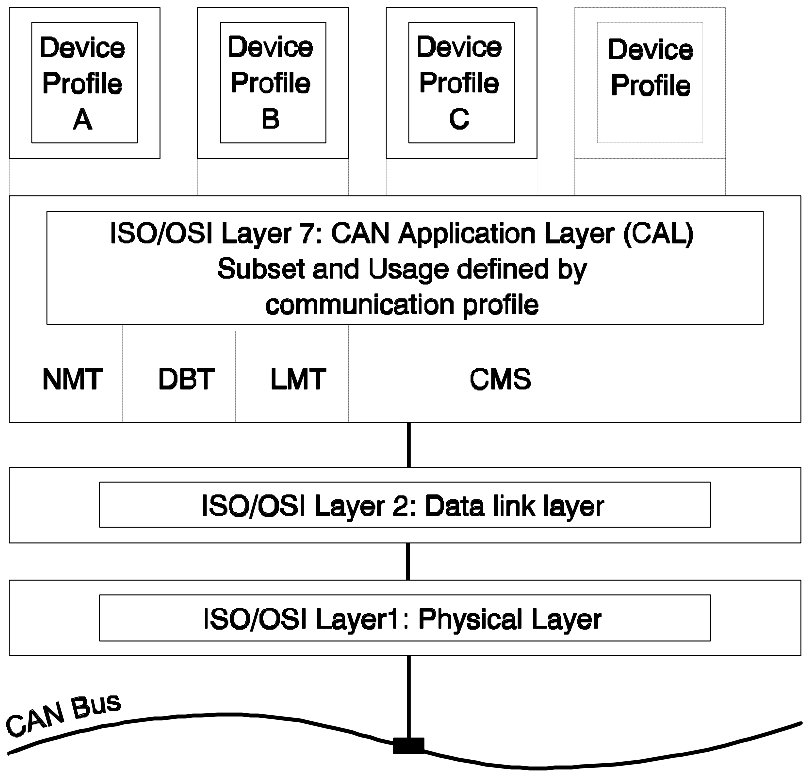

Device Profile A
Device Profile B
Device Profile C
Device Profile
ISO/OSI Layer 7: CAN Application Layer (CAL)
Subset and Usage defined by
communication profile
NMT
DBT
LMT
CMS
ISO/OSI Layer 2: Data link layer
ISO/OSI Layer1: Physical Layer
CAN Bus

Figure 2: CANopen Structure

Secondly there is parameter communication which has very different requirements: parameters have to be confirmed, they may consist of many bytes and then have to be split in several segments; parameters are typically transmitted asynchronously, and the requirements towards transmission times are moderate. It has to be possible to include address information in order to access a specific parameter out of a parameter list. CANopen introduces the Service Data Object (SDO) for such data and employs the CAL multiplexed domain service for transmission. The multiplexed domain protocol allows one to transmit parameters of up to 4 bytes with one handshake (protocol overview see Fig.6), including 3 bytes of address information. Most existing profiles (e.g. the Drivecom profiles) use 3 bytes addresses for parameters, and often limit the parameters to 4 bytes, so this expedited transfer covers all requirements for such devices. In case the service data length exceeds 4 bytes (e.g. an application program or a log file), a sequence of segmented messages follows to the initiate command. SDOs get low priority identifiers as they are not supposed to interfere with the PDO real time communication.

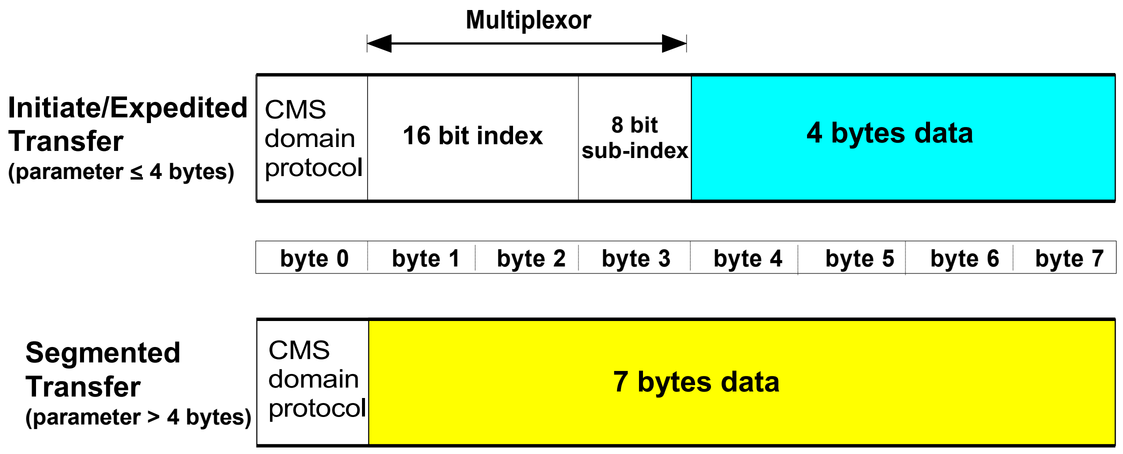


Figure 6: Service Data Object: Multiplexed Domain Protocol

3.3 Device Profiles and Object Dictionary

All device parameters are listed in an object dictionary. This object dictionary contains the description, data type and structure of the parameter as well as the address. The address being composed of a 16bit index and a 8bit sub-index guarantees compatibility with the object dictionaries of available device profiles (e.g. Drivecom). Therefore, only the bus specific entries have to be exchanged with CANopen entries. The object dictionary is organised in a communication profile specific part which contains the communication entries, and in a device specific part which contains the device entries. The device specific part is specified in the device profile, the communication entries form the common subset of all devices, therefore they are specified in the communication profile. There is a range of mandatory entries in the dictionary which ensure that all CANopen devices of a particular type show the same basic behaviour. The object dictionary concept caters for optional device features which means a manufacturer does not have to provide certain extended functionality on his device but if he wishes to do so he must do it in a pre-defined fashion. Additionally, there is sufficient address space for truly manufacturer specific functionality. This approach ensures that the CANopen device profiles are “future-proof”.

The CANopen device profiling provides a non-manufacturer specific path with upward compatibility. By defining mandatory device characteristics basic network operation is guaranteed. By defining optional device features a degree of defined flexibility can be built in. By leaving “hooks” for manufacturer specific functionality vendors will not be constrained to an out-of-date standard.

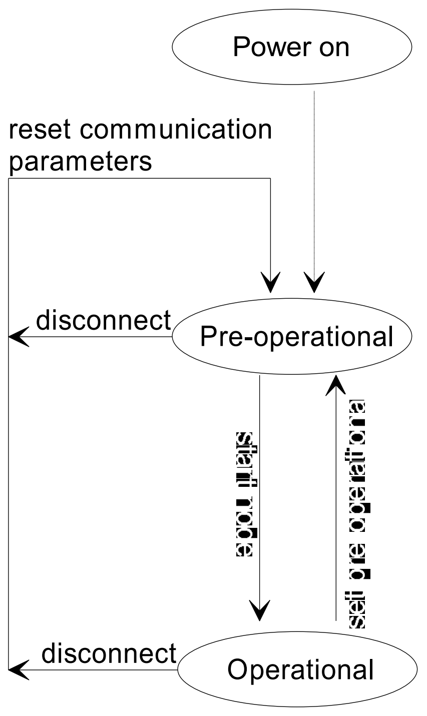
3.4 Boot-Up

The CANopen boot-up approach caters both for simple and sophisticated devices by defining a mandatory minimal boot-up procedure that can be optionally enhanced if additional features are required. The full version is equivalent to the standard CAL boot-up, ensuring that the whole range of CAL features is accessible. However, the minimal version already covers a wide range of applications.

The boot-up procedure assumes that by default the peripheral devices do not have to know what kind of application they are operating in. The network configuration takes place at one unit which can be the network management (NMT) master or a separate configuration tool called configuration master which remotely controls the NMT master. At the boot-up this master device can download the configuration data via service data objects to the configuration slaves. If the slaves are capable of storing this information, this only has to take place if the configuration changes.

CANopen defines a set of default identifiers which are derived from a node-ID, thus providing access via an SDO to the object dictionary and real-time master/slave communication via PDOs without any specific parameterisation. Of course this default identifier distribution can be modified either by changing the appropriate parameters in the object dictionary (SDO access), or by employing CAL DBT services, if present. However, applications that comprise one device that controls all others can operate sufficiently well with the default settings.

a) Minimal CANopen boot-up procedure



b) Full CANopen boot-up procedure

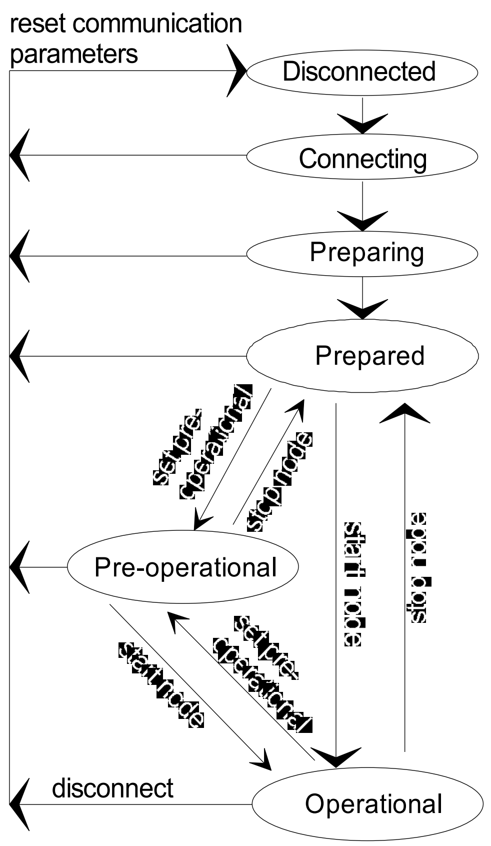


Figure 7: CANopen Boot-Up

The minimal boot-up covers only two states: pre-operational and operational (see Fig. 7a). After power-on, a device is pre-operational, thus giving read and write access to its object dictionary as the service communication is established using default identifiers. The devices can now be configured (including identifier distribution via object dictionary access) if the default settings are not satisfactory. With the standard CAL "start_remote_node" command then the devices are switched into "operational" in order to start PDO communication. PDO transmission can be stopped altogether if requested by switching the device back into pre-operational. By using the CAL command "disconnect_remote_node" all communication parameters are reset, default values (e.g. preset

identifiers) are valid again. All (NMT-) commands needed for this minimal boot-up use identifier 0 and are distinguished with the command specifier (cs) in the first data byte.
More sophisticated devices will support the full (CAL) boot-up (Fig 7.b) including DBT services which is started with a “disconnect” command, as all devices enter “pre-operational” after power-on. It is possible to have all combinations of devices in the same network, as the full boot-up can be performed separately with each device supporting it whilst the minimal boot-up is performed with the other devices. If the network master only supports minimal boot-up, all slaves behave like minimal slaves.
This boot-up concept ensures that very lean implementations are possible as all parameterisation (including most of the network configuration) can be done via one single CMS service, the multiplexed-domain protocol of the service data object. If the default settings are sufficient or if the devices are capable of storing their configuration data, the boot-up is reduced to one single two-byte message: “start all nodes”.

3.5 Bus Timing

Besides the cyclic exchange of data many real time applications demand synchronisation between different bus nodes. I.e. axis of a kinematic have to be synchronised or I/O modules have to set outputs or read inputs simultaneously like a PLC. Synchronised drives expect commanded positions and send actual positions in pre-defined time windows. CANopen meets these requirements by introducing an optional synchronisation telegram with a high priority, which divides the time axis in equidistant communication cycles (see fig. 8). The synch-message does not contain data and can be used as an interrupt by I/O modules to then set outputs or read inputs. Intelligent devices like drives can synchronise e.g. using the PLL method. In the report window right after the synchronisation telegram the drives send their actuals and the I/O modules send their input values. Afterwards, in the command window, the commands and the output values are transmitted, which are then set valid at the next synch-signal. As the report window directly follows on the synch-signal it can be hit even by simple components without using timers. Bandwidth not used inside the windows and the time between the command window and the synch telegram is available for low-priority SDO messages.
As the synchronisation telegrams are optional, it is also possible to operate CANopen networks in totally asynchronous manner if desired. However, bus traffic and processor loading are much more predictable if bus synchronisation is used.
For applications that require optimal synchronisation (the synch-message may jitter slightly due to bus traffic at the synch transmission time), an optional high resolution synchronisation method has been specified which uses time stamping of synch messages. This enhanced synchronisation is especially useful for low speed networks with hard synchronisation requirements. However, it has been shown that the standard synchronisation method perfectly good at operating robot kinematics.

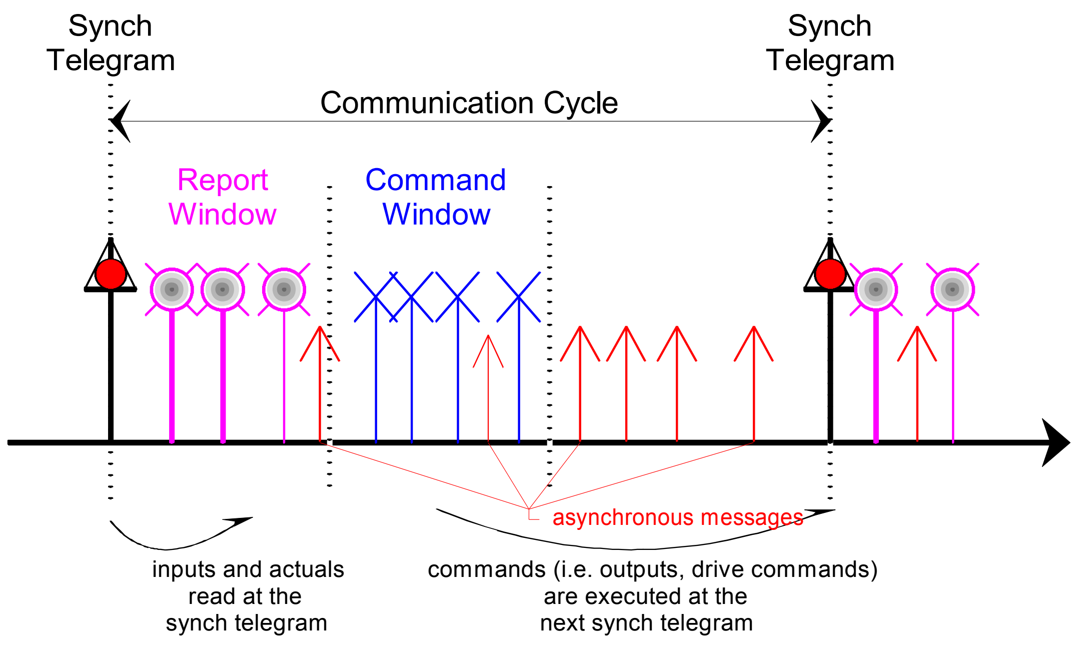

Synch Telegram
Communication Cycle
Synch Telegram
Report Window
Command Window
asynchronous messages
inputs and actuals read at the synch telegram
commands (i.e. outputs, drive commands) are executed at the next synch telegram

Figure 3: CANopen Bus Timing

4 Implementations

CANopen emerged from a joint European research project. In this project, several pilot networks featuring CANopen prototypes have been set up, thus ensuring that implementation experience accompanies the final profile specifications. A multi-vendor network similar to the one shown in Fig. 1 was displayed at the Hanover Industrial Fair 1995.

Although the CANopen communication profile specification was finalised only recently, there are already a number of CANopen implementations from several companies, many of them supported, some performed by STA Reutlingen. Experience with these implementations shows, that the CANopen approach allows generation of very lean code. It was possible to program a CANopen drive interface in C-language on an 8051-type controller with CANopen boot-up, dynamic identifier distribution, two PDOs, two SDOs, synch and emergency message support and full object dictionary access with less than 80 bytes of RAM and less than 6 Kbytes of ROM. The interrupt routine for the protocol is executed in less than 100 µs. A full implementation for a digital I/O module (including all features) requires about 10 Kbytes ROM, a down-sized version can be implemented with less than 4 Kbytes.

5 Test and Certification

The advantages of open systems are only achieved if the protocol implementations are in exact agreement with the specification. Therefore suitable protocol test methods are being developed. Both in conformity testing (ISO/OSI-layers) as in interoperability testing (profiles) the device under test is stimulated with extensive test message sequences. These are combined manually or in automatic mode randomly out of a large number of small test strings in order to achieve maximal variance of test states. The test evaluation is performed automatically as well. In doing so long time tests are possible with changing characteristics.

Apart from the peer to peer test set-up there are multi vendor test beds available. Special care is taken to ensure that the devices not only work in one specific set-up, but in various environments with differing complexity, e.g. minimal slave implementations together with a full master implementation and vice versa.
The test will be performed by independent and acknowledged institutions, certification based on the test results will be done by the CANopen organisation.

6 CANopen: the Open Communication Standard

The CANopen development started in 1992, first prototype implementations have been available since beginning of 1994. CANopen so far has mainly focused on the technical aspects of the protocol development, ensuring that the resulting specifications meet the technical demands of a wide range of applications. Now that the specifications are available, the promotion and marketing of this protocol family is becoming more active. There is no large dominating company pushing this protocol with a large marketing budget, but there are many medium sized companies involved reflecting the structure of the European automation market, especially of the CAN market.
First products are already available, others are currently developed. Major drive and I/O component manufacturers have decided to use CANopen as their CAN protocol, several master implementations on PLCs and PLC interfaces are under way. The manufacturers have understood that CANopen has the following advantages:

- it is an open protocol, that is independent of a specific manufacturer
- the real-time capabilities of CAN are not restricted, but easily accessible
- it is modular, one only has to implement the required features
- it is interoperable
- it covers a wide range of applications, from robot control and PLC applications to networks with distributed intelligence, building automation applications etc.
- the profile structure is similar to existing profiles from Interbus-S, Profibus and others

Process data communication with CANopen is “pure CAN” without protocol overhead.

CANopen supports:

- auto-configuration of the network
- comfortable access to all device parameters
- device and network synchronisation
- cyclic and event driven process data transmission
- simultaneous reading of inputs
- simultaneous setting of outputs

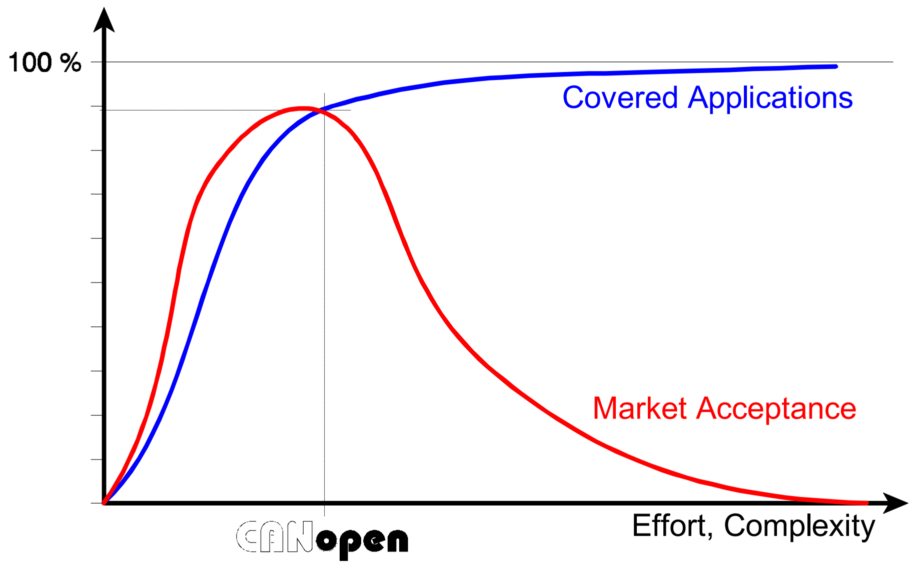


Figure 4: Market Acceptance versus Effort

Carefully the balance between the range of covered applications and the related effort and complexity has been restored (see fig.9), as market acceptance decreases significantly (due to costs) when the protocol gets too complex. Experience with the CANopen approach in various pilot applications has proven the usability of this protocol family.

Literature

[1] ISO 11898: 'Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high speed communications'.1993.

[2] CiA Standard 'CAN Physical Layer for Industrial Applications', CiA-DS 102, 1994.

[3] CiA Standard 'CAN Application Layer', CiA/DS 201-207, 1994.

[4] CiA Standard 'CAL based Communication Profile for Industrial Applications', CiA/DS 301, 1995.

[5] Gruhler, G.; Rostan, M.: ‘A New Generation of Control Systems for Production Cells’, Proceedings of the 5th FAIM Conference. Begell House, New York 1995.

[6] Tischer, M.; Pisarz, J.:'Open Communication for Drives', Proceedings PCIM 1995.

[7] Rostan, M.; Gruhler, G., ’CAN Real Time Communication Profile’, Proceedings 1st International CAN Conference 1994, pp. 6-36...6-43, Erlangen 1994.

[8] Rostan, M., Gruhler, G.; 'Echtzeitkommunikation mit CAN/CAL unter Nutzung von Geräteprofilen', Vortragsband iNet 1994, pp. 250...256, Hagenburg: Network 1994.

[9] Gruhler, G.; Jantzer, M.: 'Anforderungen an einen Feldbus für Montagezellen'. Vortragsband iNet 1993, pp. 77...82, München: Drebinger 1993.

[10] Jantzer, M.; Rostan M., ’Requirements on an Installation Bus Concept for Production Units’, CIM-Europe Conference, Amsterdam 1993.

[11] Gruhler, G.; Rostan, M.: Auswahl eines Feldbussystems für Produktionszellen. Elektronik plus 6/93, S. 77...82, München 1993.

[12] Einführende Darstellung und detaillierter Vergleich von Feldbussystemen. 1993/1995. Available from: STA Reutlingen, phone +49 7121 271-327.

[13] ASPIC Consortium: Specification of CAN Real Time Communication Profile. 1994.

[14] Ratcliff, K.; Booth, R.; Farsi, M.: Principles of ASPIC Device Communication. University of Newcastle upon Tyne, United Kingdom 1994.