SLOVENSKI STANDARD
SIST EN 50325-4:2003

01-julij-2003

Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces - Part 4: CANopen

Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces -- Part 4: CANopen

Industrielles Kommunikationssubsystem basierend auf ISO 11898 (CAN) -- Teil 4: CANopen

Sous-système de communications industriel basé sur l'ISO 11898 (CAN) pour les interfaces des dispositifs de commande -- Partie 4: CANopen

Ta slovenski standard je istoveten z: EN 50325-4:2002

ICS:

35.240.50	Uporabniške rešitve IT v industriji	IT applications in industry
43.040.15	Avtomobilska informatika. Vgrajeni računalniški sistemi	Car informatics. On board computer systems

SIST EN 50325-4:2003 en

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 50325-4

December 2002

ICS 43.180

English version

Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces Part 4: CANopen

Sous-système de communications industriel basé sur l'ISO 11898 (CAN) pour les interfaces des dispositifs de commande
Partie 4: CANopen

Industrielles Kommunikationssubsystem basierend auf ISO 11898 (CAN)
Teil 4: CANopen

This European Standard was approved by CENELEC on 2002-07-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in one official version (English). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Luxembourg, Malta, Netherlands, Norway, Portugal, Slovakia, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Ref. No. EN 50325-4:2002 E

Foreword

This European Standard was prepared by the Technical Committee CENELEC TC 65CX, Fieldbus.

The text of the draft was submitted to the Unique Acceptance Procedure and was approved by CENELEC as EN 50325-4 on 2002-07-01.

The following dates were fixed:

- latest date by which the EN has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2003-07-01
- latest date by which the national standards conflicting with the EN have to be withdrawn	(dow)	2005-07-01

Annexes designated "normative" are part of the body of the standard.

Annexes designated "informative" are given for information only.

In this standard, annexes A and B are normative and annexes C, D and E are informative.

This European standard is part of EN 50325 which consists of four parts:

Part 1	General requirements
Part 2	DeviceNet
Part 3	Smart Distributed System (SDS)
Part 4	CANopen

The specifications for DeviceNet,SDS and CANopen are based on ISO 11898 *Controller area network (CAN) for high-speed communication*, a broadcast-oriented communications protocol. However, ISO 11898 specifies only part of a complete communication system, and additional specifications are needed for other layers to ensure precise data exchange functionality and support of inter-operating devices.

General information on licensing and patents

Attention is drawn to the possibility that some of the elements of the European Standard EN 50325-4 may be the subject of patent rights. CENELEC shall not be held responsible for identifying any or all such patent rights

If during the application of those Standards Intellectual Property Rights may appear and will not be made available on reasonable and non discriminatory terms and conditions to anyone wishing to obtain such a license, applying the rules of CEN/CENELEC Memorandum 8, this fact shall be brought to the attention of CENELEC Central Secretariat for further action.

Contents

Introduction

CANopen is intended for use in, but is not limited to, industrial automation applications. These applications may include devices such as generic digital and analogue input/output modules, motion controllers, human machine interfaces, sensors, closed-loop controllers, encoders, hydraulic valves, and programmable controllers.

1 Scope

EN 50325-4 specifies the following particular requirements for CANopen:

- requirements for interfaces between programmable controllers and devices with input/output capabilities;
- normal service conditions for devices;
- constructional and performance requirements.

2 Normative references

EN 50081-2	1993	Electromagnetic compatibility (EMC) – Generic emission standard - Part 2: Industrial environment
EN 61000-6-2	1999	Electromagnetic compatibility (EMC) - Generic standards - Part 2: Immunity for industrial environments
EN 55011	1998	Industrial, scientific and medical (ISM) radio-frequency equipment – Radio disturbance characteristics - Limits and methods of measurement (CISPR 11: 1997, mod.)
EN 61000-4		Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques
EN 61131-3	1993	Programmable controllers – Part 3: Programming languages (IEC 61131-3:1993)
ISO 11898	1993	Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication
ISO 646	1991	Information technology - ISO 7-bit coded character set for information interchange
ISO 7498-1	1994	Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model
ISO 8859	1998	Information technology - 8-bit single-byte coded graphic character sets

3 Definitions

For the purpose of EN 50325-4 the definitions of EN 50325-1 and the following apply.

3.1
Automatic Repeat Request (ARQ)
scheme used to confirm the transmission of an SDO block

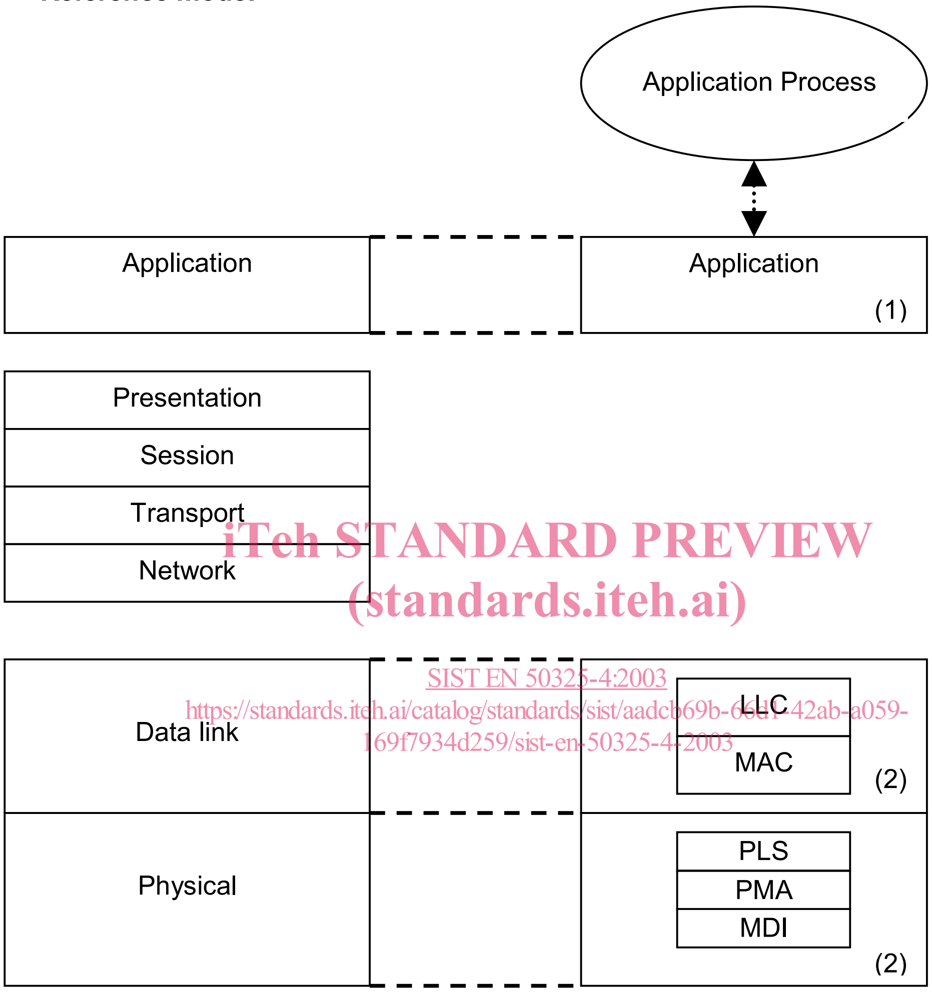
3.2
Node-ID
number, uniquely assigned to an NMT (NetworkManagemenT) slave; if 0, the NMT protocols address all NMT slaves

4 Classifications

4.1 General

Networks compliant to EN 50235-4 shall use the following reference model, device model, and communication model.

4.2 Reference model



(1) specified in this European Standard

(2) specified in ISO 11898

Figure 1 – Comparison with OSI reference model

The communication concept is described similar to the OSI reference model (left side of Figure 1).

4.2.1 Application layer:

The application layer comprises a concept to configure and communicate real-time-data as well as the mechanisms for synchronisation between devices. The functionality the application layer offers to an application is logically divided over different *service objects* in the application layer. A service object offers a specific functionality and all the related services. These services are described in the *Service Specification* of that service object.

Applications interact by invoking services of a service object in the application layer. To realise these services, this object exchanges data via the CAN network with (a) peer service object(s) via a protocol. This protocol is described in the *Protocol Specification* of that service object.

4.2.2 Service primitives:

Service primitives are the means by which the application and the application layer interact. There are four different primitives

- a *request* is issued by the application to the application layer to request a service,
- an *indication* is issued by the application layer to the application to report an internal event detected by the application layer or indicate that a service is requested,
- a *response* is issued by the application to the application layer to respond to a previous received indication,
- a *confirmation* is issued by the application layer to the application to report the result of a previously issued request.

4.2.3 Application layer service types

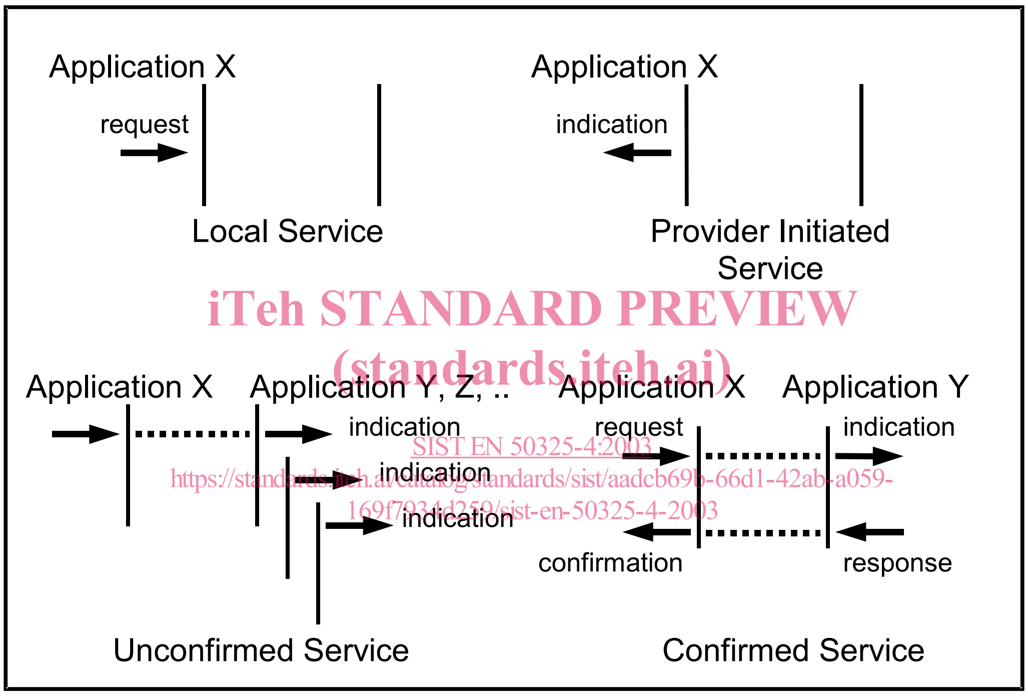


Figure 2 - Service types

A *service type* defines the primitives that are exchanged between the application layer and the co-operating applications for a particular service of a service object (see Figure 2).

- A *local service* involves only the local service object. The application issues a request to its local service object that executes the requested service without communicating with (a) peer service object(s).
- An *unconfirmed service* involves one or more peer service objects. The application issues a request to its local service object. This request is transferred to the peer service object(s). Each of them pass it to their application as an indication. The result is not confirmed back.
- A *confirmed service* may involve only one peer service object. The application issues a request to its local service object. This request is transferred to the peer service object that passes it to the other application as an indication. The other application issues a response that is transferred to the originating service object that passes it as a confirmation to the requesting application.
- A *provider initiated service* involves only the local service object. The service object (being the service provider) detects an event not solicited by a requested service. This event is then indicated to the application.

Unconfirmed and confirmed services are collectively called *remote services*.

4.3 Device model

4.3.1 General

A device is modelled as follows (see Figure 3):

- communication – This function unit provides the communication objects and the appropriate functionality to transport data items via the underlying network structure;
- object dictionary – The Object dictionary is a collection of all the data items which have an influence on the behaviour of the application objects, the communication objects and the state machine used on this device;
- application – The application comprises the functionality of the device with respect to the interaction with the process environment.

Thus the Object dictionary serves as an interface between the communication and the application. The complete description of a device's application with respect to the data items in the Object dictionary is named *device profile*.

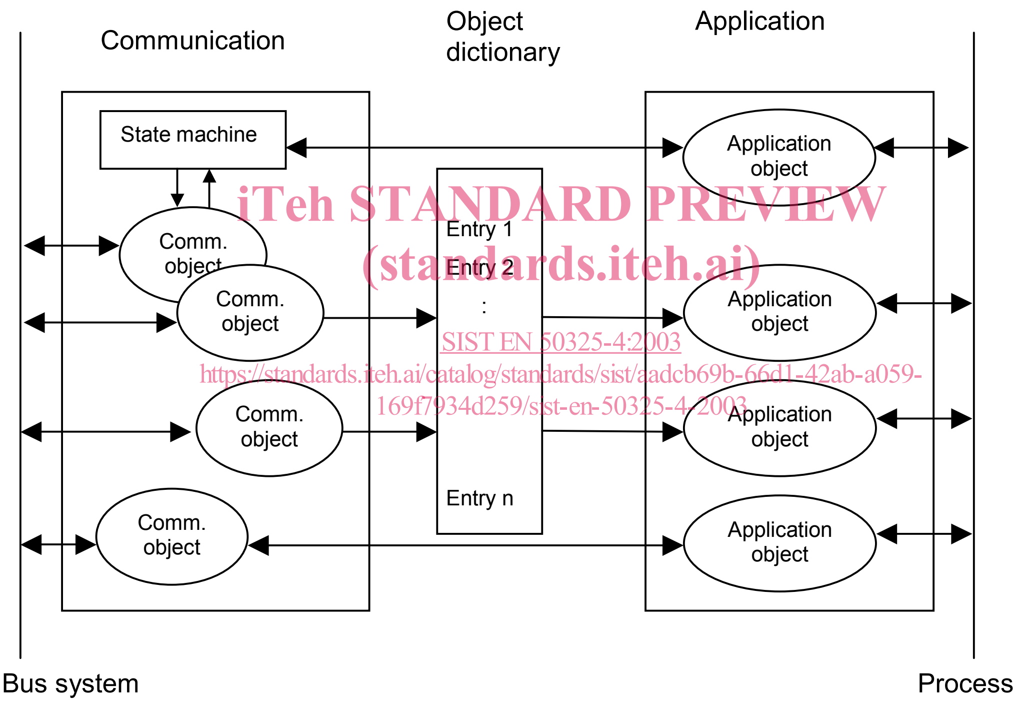


Figure 3 - Device model

4.3.2 Object dictionary

4.3.2.1 General

The most important part of a device profile is the Object dictionary description. The Object dictionary is essentially a grouping of objects accessible via the network in an ordered pre-defined fashion. Each object within the dictionary shall be addressed using a 16-bit index.

The overall layout of the standard Object dictionary is shown in Table 1. This layout closely conforms to other industrial serial bus system concepts.

Table 1 - Object dictionary structure

Index (hex)	Object
0000	Not used
0001-001F	Static data types
0020-003F	Complex data types
0040-005F	Manufacturer specific complex data types
0060-007F	Device profile specific static data types
0080-009F	Device profile specific complex data types
00A0-0FFF	Reserved for further use
1000-1FFF	Communication profile area
2000-5FFF	Manufacturer specific profile area
6000-9FFF	Standardised device profile area
A000-FFFF	Reserved for further use

The object dictionary may contain a maximum of 65536 entries which are addressed through a 16-bit index.

The Static Data Types at indices 0001h through 001Fh shall contain type definitions for simple data types like boolean, integer, floating point, string, etc. These entries are included for reference only, they shall not be read or written.

Complex Data Types at indices 0020h through 003Fh are pre-defined structures that are composed of simple data types and are common to all devices.

Manufacturer Specific Complex Data Types at indices 0040h through 005Fh are structures composed of standard data types but are specific to a particular device.

Device profiles may define additional data types specific to their device type. The static data types defined by the relevant device profile are listed at indices 0060h - 007Fh, the complex ones at indices 0080h - 009Fh.

A device may provide the structure of the supported complex data types (indices 0020h - 005Fh and 0080h - 009Fh) at read access to the corresponding index. Sub-index 0 then provides the number of entries at this index, and the following sub-indices contain the data type encoded as UNSIGNED16 according to Table B.4.

The Communication Profile Area at indices 1000h through 1FFFh shall contain the communication specific parameters for the CAN network. These entries shall be common to all devices.

The standardised device profile area at indices 6000h through 9FFFh contains all data objects common to a class of devices that may be read or written via the network. The device profiles may use entries from 6000h to 9FFFh to describe the device parameters and the device functionality. Within this range up to 8 different devices may be described. In such a case the devices are denominated Multiple Device Modules. Multiple Device Modules are composed of up to 8 device profile segments. By this feature it is possible to build devices with multiple functionality.

The object entries 6000h to 67FFh of each additional device profiles shall be shifted as follows:

-	6000h to 67FFh	for	device 0
-	6800h to 6FFFh	for	device 1
-	7000h to 77FFh	for	device 2
-	7800h to 7FFFh	for	device 3
-	8000h to 87FFh	for	device 4
-	8800h to 8FFFh	for	device 5
-	9000h to 97FFh	for	device 6
-	9800h to 9FFFh	for	device 7

The PDO distribution shall be used for every segment of a Multiple Device Module with an offset of 64, e.g. the first PDO of the second segment gets the number 65. In this way a system with a maximum of 8 segments is supported.

The object dictionary concept caters for optional device features which means a manufacturer does not have to provide certain extended functionality on his devices. However, if he implements this optional functionality, he shall be compliant to definitions given in this European Standard.

Space is left in the Object dictionary at indices 2000h through 5FFFh, which may be used for truly manufacturer-specific functionality.

4.3.2.2 Index and sub-Index usage

A 16-bit index shall be used to address all entries within the Object dictionary. In case of a simple variable the index references the value of this variable directly. In case of records and arrays, however, the index addresses the whole data structure.

To allow individual elements of structures of data to be accessed via the network a sub-index is defined. For single Object dictionary entries such as an UNSIGNED8, BOOLEAN, INTEGER32, etc. the value for the sub-index shall be always zero. For complex Object dictionary entries such as arrays or records with multiple data fields the sub-index references fields within a data-structure pointed to by the main index. The fields accessed by the sub-index may be of different data types.

4.4 Communication model

4.4.1 General

The communication model specifies the different communication objects and services and the available triggering modes of message transmission.

The communication model supports the transmission of synchronous and asynchronous messages. By means of synchronous message transmission a network wide co-ordinated data acquisition and data actuation is possible. The synchronous transmission of messages is supported by pre-defined communication objects (Sync message, time stamp message). Synchronous messages are transmitted with respect to the pre-defined Sync message, asynchronous messages may be transmitted at any time.

Due to the event-driven character of the underlying communication mechanism it is possible to define inhibit times for the communication. In order to guarantee that even communication objects with low priorities are transmitted in an appropriate time, communication objects may be assigned an inhibit-time. The inhibit-time of a communication object defines the minimum time that shall elapse between two consecutive invocations of a transmission service for that communication object. Inhibit-times may be assigned by the application.

With respect to their functionality, three types of communication relationships are distinguished

- master/slave relationship (Figure 4 and Figure 5),
- client/server relationship (Figure 6),
- producer/consumer relationship (Figure 7 and Figure 8).

4.4.2 Master/slave relationship

At any time there is exactly one device in the network serving as a master for a specific functionality. All other devices in the network are considered as slaves. The master issues a request and the addressed slave(s) respond(s) if the protocol requires this behaviour.

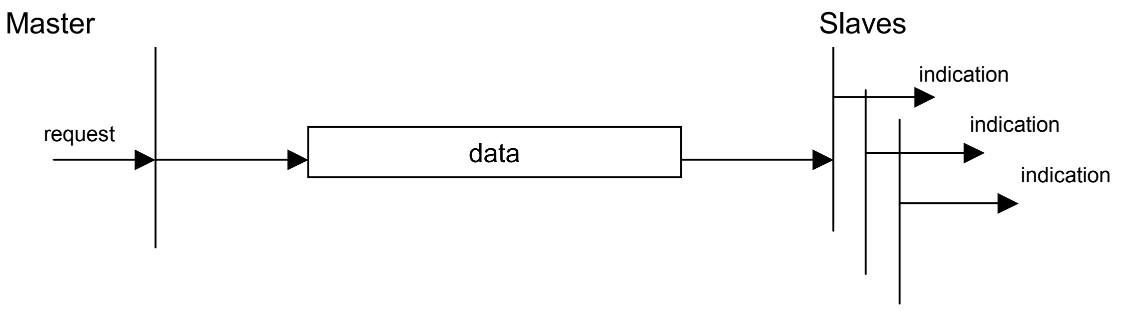


Figure 4 - Unconfirmed master/slave communication

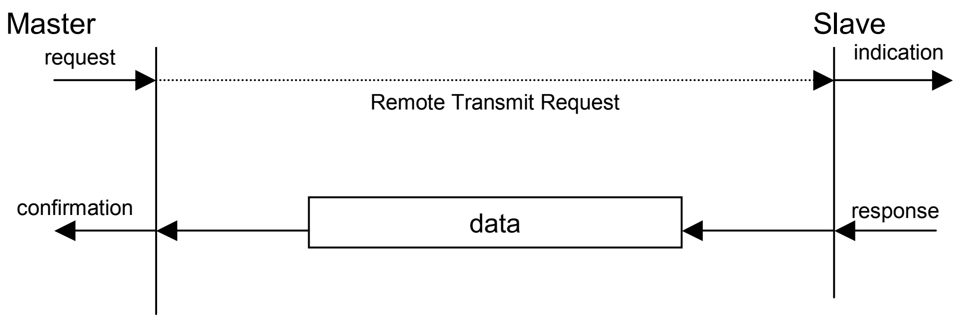


Figure 5 - Confirmed master/slave communication

4.4.3 Client/server relationship

This is a relationship between a single client and a single server. A client issues a request (upload/download) thus triggering the server to perform a certain task. After finishing the task the server answers the request.

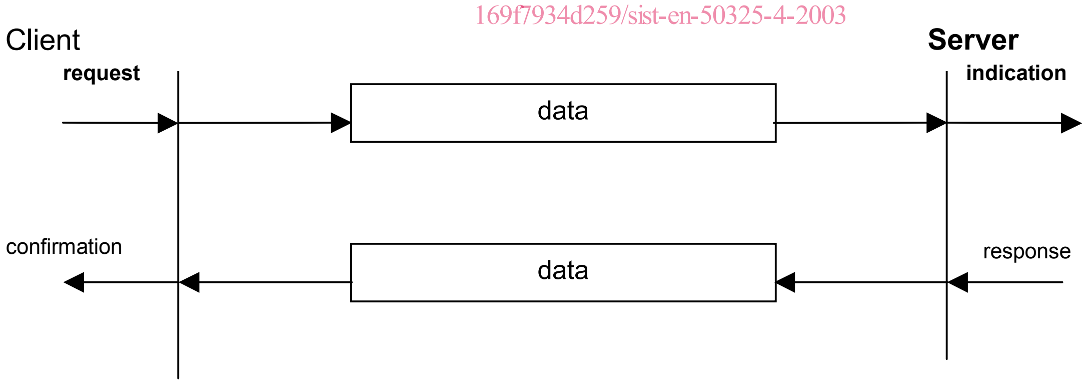


Figure 6 - Client/server communication

4.4.4 Producer/consumer relationship - Pull/push model

The producer/consumer relationship model involves a producer and zero or more consumer(s). The push model is characterised by an unconfirmed service requested by the producer. The pull model is characterised by a confirmed service requested by the consumer.

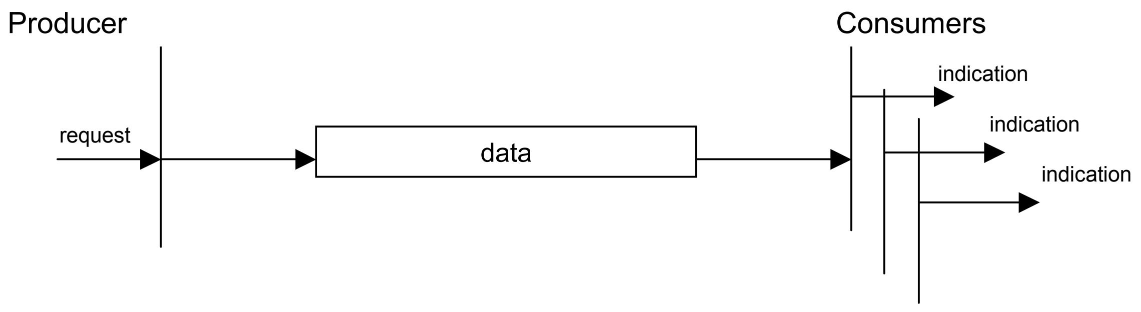


Figure 7 - Push model

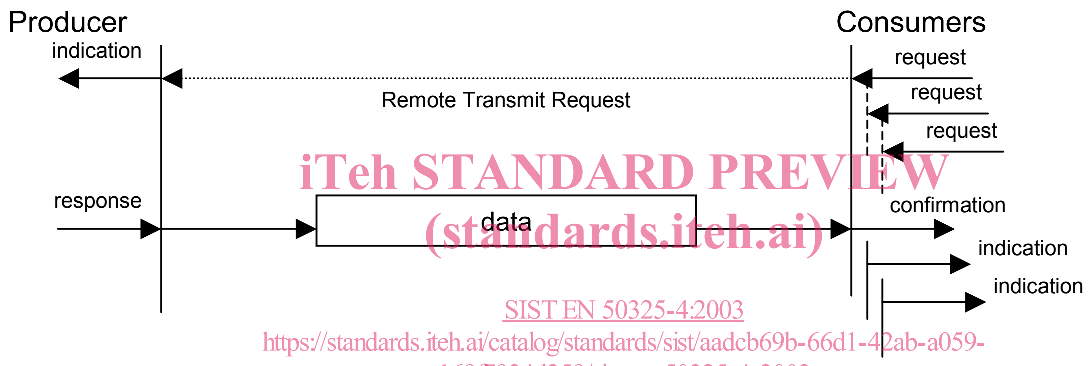


Figure 8 - Pull model

4.5 Data link layer

4.5.1 General

Networks compliant to EN 50235-4 shall be implemented on a data link layer and its sub-layers according to ISO 11898.

4.5.2 CAN frame type

This specification shall be implemented using CAN standard frames with 11-bit identifier field. It is not required to support the CAN extended frame with 29-bit identifier field.

However, as certain applications may require the usage of the extended frame with 29-bit identifier field the network may be operated in this mode, if all nodes support it.

5 Characteristics

5.1 Communication objects

5.1.1 General

The services and protocols describe the communication objects.

All services are described in a tabular form that contains the parameters of each service primitive that is defined for that service.

The primitives that are defined for a particular service determine the service type (e.g. unconfirmed, confirmed, etc.). How to interpret the tabular form and what service types exist is defined in 4.4 (Communication model).

All services assume that no failures occur in the data link and physical layer of the CAN network. These failures are resolved by the application and fall not in the scope of this European Standard.

5.1.2 Process Data Object (PDO)

5.1.2.1 General

The real-time data transfer is performed by means of Process Data Objects (PDO). The transfer of PDOs is performed with no protocol overhead.

The PDOs correspond to entries in the device Object dictionary and provide the interface to the application objects. Data type and mapping of application objects into a PDO is determined by a corresponding default PDO mapping structure within the Device Object dictionary.

If variable PDO-mapping is supported the number of PDOs and the mapping of application objects into a PDO may be transmitted to a device during the device configuration process (see Initialisation Procedure) by applying the SDO services to the corresponding entries of the Object dictionary.

Number and length of PDOs of a device is application specific and shall be specified within the device profile.

There are two kinds of use for PDOs. The first is data transmission and the second data reception. It is distinguished in Transmit-PDOs (TPDOs) and Receive-PDOs (RPDOs). Devices supporting TPDOs are PDO producers and devices which are able to receive PDOs are called PDO consumer. The PDO communication parameter (20h) and the PDO mapping parameter (21h) describe PDOs. The structure of these data types are explained in annex A. The PDO communication parameter describes the communication capabilities of the PDO. The PDO mapping parameter contains information about the contents of the PDOs (device variables). The indices of the corresponding Object dictionary entries shall be computed by the following formulas:

- RPDO communication parameter index = 1400h + RPDO-number –1;
- TPDO communication parameter index = 1800h + TPDO-number –1;
- RPDO mapping parameter index = 1600h + RPDO-number –1;
- TPDO mapping parameter index = 1A00h + TPDO-number –1.

For each PDO the pair of communication and mapping parameter shall be implemented. The entries mentioned above are described in annex B.

5.1.2.2 Transmission modes

The following PDO transmission modes are distinguished:

- Synchronous transmission
- Asynchronous transmission

In order to synchronise devices a synchronisation object (SYNC object) is transmitted periodically by a synchronisation application. The SYNC object is represented by a pre-defined communication object (see 5.1.4). In Figure 9 the principle of synchronous and asynchronous transmission is shown. Synchronous PDOs are transmitted within a pre-defined time-window immediately after the SYNC object. The principle of synchronous transmission is described in more detail in 5.2.

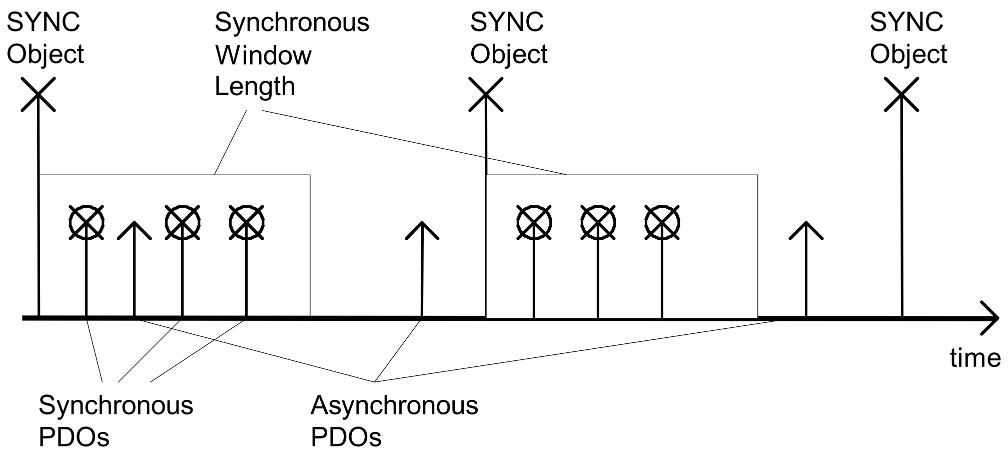


Figure 9 - Synchronous and asynchronous transmission

The transmission type parameter of a PDO specifies the transmission mode as well as the triggering mode.

For synchronous TPDOs the transmission type also specifies the transmission rate in form of a factor based on the basic SYNC-object transmission period. A transmission type of 0 means that the message shall be transmitted after occurrence of the SYNC but acyclic (not periodically), this is if an event occurred before the SYNC. A transmission type of 1 means that the message shall be transmitted with every SYNC object. A transmission type of n means that the message shall transmitted with every n-th SYNC object. Asynchronous TPDOs are transmitted without any relation to a SYNC.

The data of synchronous RPDOs received after the occurrence of a SYNC is passed to the application with the occurrence of the following SYNC, independent of the transmission rate specified by the transmission type. The data of asynchronous RPDOs is passed directly to the application.

5.1.2.3 Triggering modes

Three message triggering modes are distinguished:

- **Event driven**
  Message transmission is triggered by the occurrence of an object specific event. For synchronous PDOs this is the expiration of the specified transmission period, synchronised by the reception of the SYNC object.
  For acyclically transmitted synchronous PDOs and asynchronous PDOs the triggering of a message transmission is a device-specific event specified in the device profile.
- **Timer driven**
  Message transmission is either triggered by the occurrence of a device-specific event or if a specified time has elapsed without occurrence of an event.
- **Remotely requested**
  Transmission of an asynchronous PDO is initiated on receipt of a remote request initiated by any other device (PDO consumer).