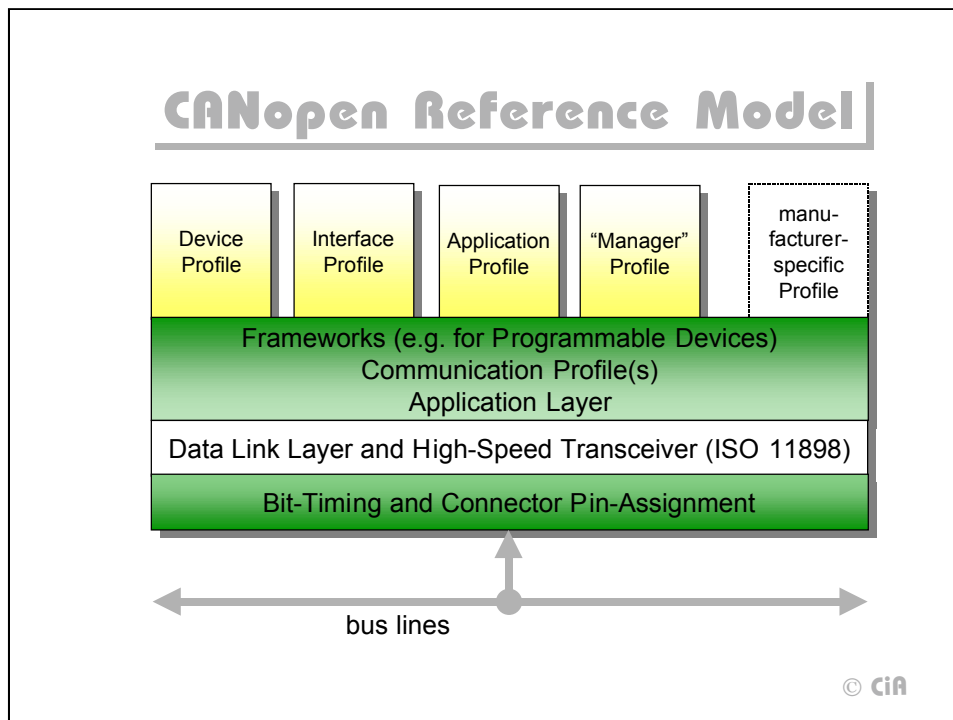


CANopen is a networking system based on the CAN serial bus. CANopen assumes that the device's hardware has a CAN transceiver and CAN controller as specified in ISO 11898.

CANopen profile family specifies standardized communication mechanisms and device functionality. The profile family is available and maintained by CAN in Automation (CiA), the international users' and manufacturers' group and may be implemented license-free.

The CANopen specifications cover application layer and communication profile (CiA DS-301) as well as a framework for programmable devices (CiA DSP-302), recommendations for cables and connectors (CiA DRP-303-1) and SI units and prefix representations (CiA DRP-303-2). Additional application-specific frameworks may apply in parallel. CANopen is supplemented by a number of standardized device profiles, interface profiles as well as application profiles (CiA DS-4XX).

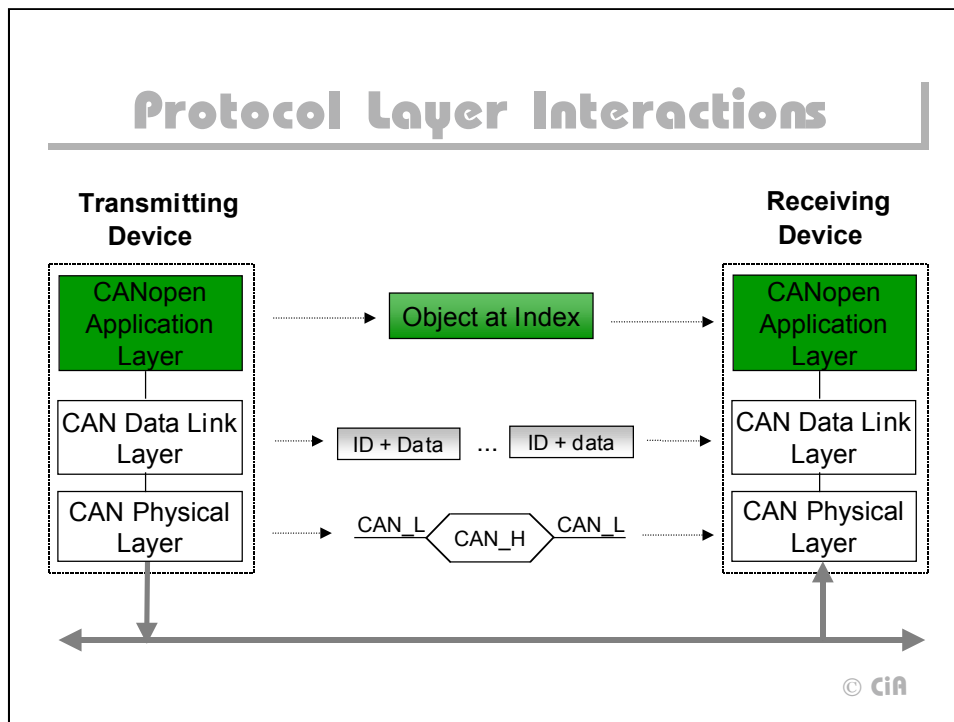
CANopen was originally designed for motion-oriented industrial control systems, such as handling systems. But CANopen networks are also used in other application fields, e.g. public transportation, off-road vehicles, medical equipment, maritime electronics, and building automation.



The CANopen communication concept can be described similar to the ISO Open Systems Interconnection (OSI) Reference Model. CANopen represents a standardized application layer and communication profile. The optional framework for programmable devices specifies additional communication functionality.

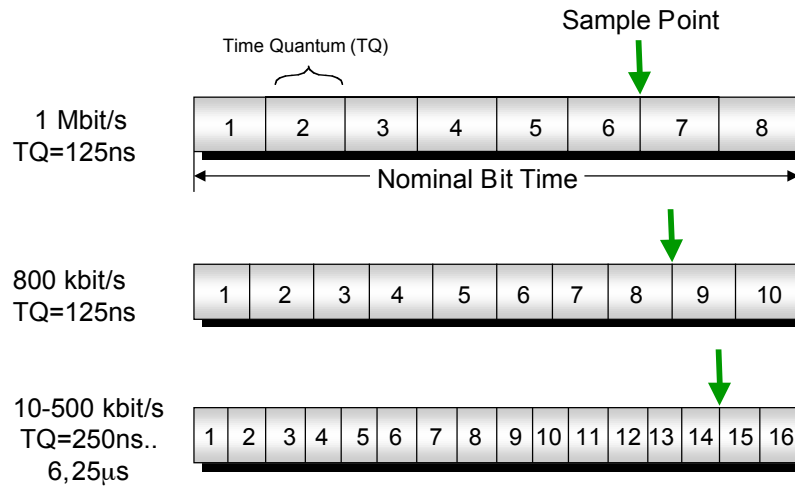
CANopen is based on the CAN data link layer and high-speed transceiver as specified in ISO 11898, part 1 and part 2. In addition, CANopen specifies bit-timing and recommends pin-assignments for some connectors.

The standardized device profiles, interface profiles, and application profiles describes the default behavior and the optional functionality of devices, interfaces, and applications.



The protocol layer interactions describe the communication on the different layers. On the CANopen application layer the devices exchange communication and application objects. All these objects are accessible via a 16-bit index and 8-bit sub-index. These communication objects (COB) are mapped to one or more CAN frames with pre-defined or configured Identifiers. The CAN physical layer specifies the bit level including the bit-timing.

Bit-Timing Specification



© **cia**

At a bitrate of 1 Mbit/s the bit consists out of 8 time quanta, at 800 kbit/s out of 10 time quanta and from 500 kbit/s to 10 kbit/s out of 16 time quanta. CANopen uses single sampling mode only.

Bit-Timing (2)

Bit rate Bus length ⁽¹⁾	Nominal bit time t_b	Number of time quanta per bit	Length of time quantum t_q	Location of sample point	Index
1 Mbit/s 25 m	1 μ s	8	125 ns	6 t_q (750 ns)	0
800 kbit/s 50 m	1.25 μ s	10	125 ns	8 t_q (1 μ s)	1
500 kbit/s 100 m	2 μ s	16	125 ns	14 t_q (1.75 μ s)	2
250 kbit/s 250 m ⁽²⁾	4 μ s	16	250 ns	14 t_q (3.5 μ s)	3
125 kbit/s 500 m ⁽²⁾	8 μ s	16	500 ns	14 t_q (7 μ s)	4
50 kbit/s 1000 m ⁽³⁾	20 μ s	16	1.25 μ s	14 t_q (17.5 μ s)	5
20 kbit/s 2500 m ⁽³⁾	50 μ s	16	3.125 μ s	14 t_q (43.75 μ s)	6
10 kbit/s 5000 m ⁽³⁾	100 μ s	16	6.25 μ s	14 t_q (87.5 μ s)	7

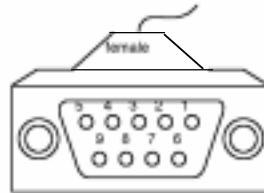
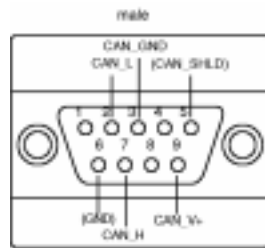
© CIA

Every module has to support one of the specified bit rates. The rounded bus length estimation (worst case) on basis 5 ns/m propagation delay and a total effective device-internal in-out delay is as follows:

1M - 800 kbit/s:	210 ns
500 - 200 kbit/s:	300 ns (includes 2 x 40 ns for optocouplers)
125 kbit/s:	450 ns (includes 2 x 100 ns for optocouplers)
50 - 10 kbit/s:	1,5 time quanta; effective delay = delay recessive to dominant plus dominant to recessive divided by two.

For bus length greater than about 200 m the use of optocouplers is recommended. For bus length greater than about 1 km bridge or repeater devices may be needed.

Pin Assignment



9-pin D-Sub: DIN 41652

Pin	Signal	Description
1	-	Reserved
2	CAN_L	CAN_L bus line dominant low
3	CAN_GND	CAN Ground
4	-	Reserved
5	(CAN_SHLD)	Optional CAN Shield
6	GND	Optional Ground
7	CAN_H	CAN_H bus line dominant high
8	-	Reserved
9	(CAN_V+)	Optional CAN external positive supply

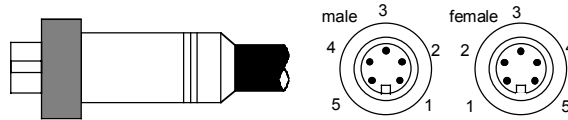
© CiA

The physical medium for CANopen devices is a differentially driven two-wire bus line with common return according to ISO 11898.

The CiA DRP-303-1 CANopen recommendation defines the pinning of the 9-pin D-sub connector (DIN 41652 or corresponding international standard), the 5-pin mini style connector, the open style connector, the multi-pole connector, and other connectors. The 9-pin D-Sub connector pinning complies to CiA DS-102.

Mini Style Connector

5-pin Mini Style Connector : ANSI/B93.55M-1981



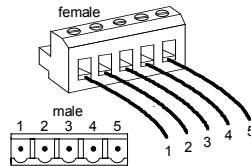
If 5-pin Mini Style Connectors are used the following pinning applies:

Pin	Signal	Description
1	(CAN_SHLD)	Optional CAN Shield
2	(CAN_V+)	Optional CAN external positive supply (dedicated for supply of transceiver and optocouplers, if galvanic isolation of the bus node applies)
3	CAN_GND	Ground / 0V / V-
4	CAN_H	CAN_H bus line (dominant high)
5	CAN_L	CAN_L bus line (dominant low)

© **ciA**

Open Style Connector

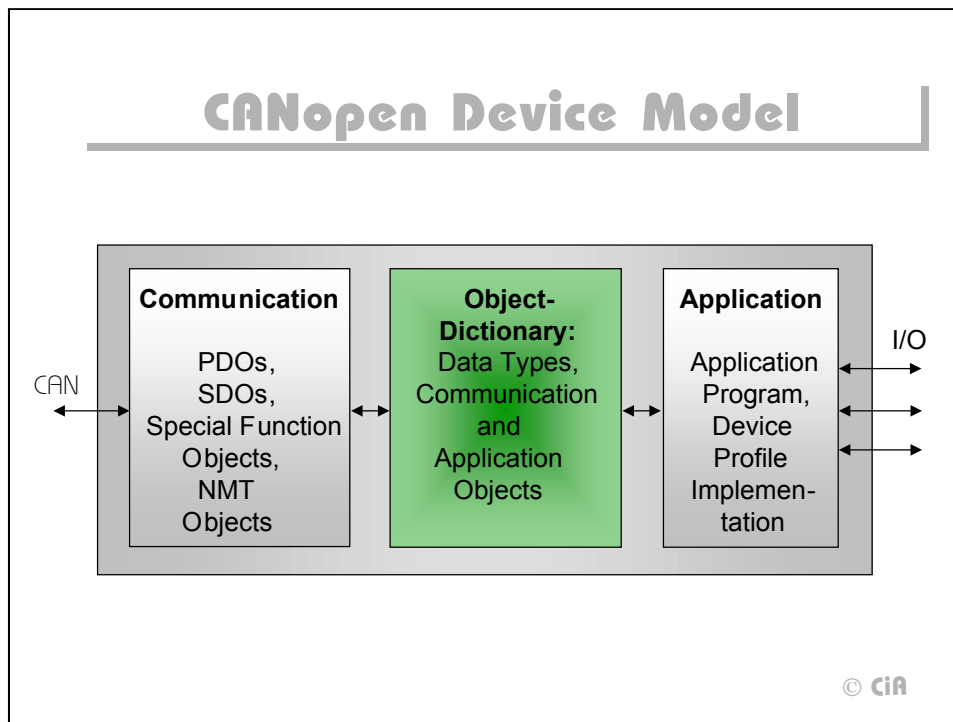
Open Style Connector



If Open Style Connectors are used the following pinning is recommended:

Pin	Signal	Description
1	CAN_GND	Ground / 0 V / V-
2	CAN_L	CAN_L bus line (dominant low)
3	(CAN_SHLD)	Optional CAN Shield
4	CAN_H	CAN_H bus line (dominant high)
5	(CAN_V+)	Optional CAN external positive supply (dedicated for supply of transceiver and optocouplers, if galvanic isolation of the bus node applies)

© **ciA**



A CANopen device can be divided into three parts:

- communication interface and protocol software
- object dictionary
- process interface and application program

The communication interface and protocol software provide services to transmit and to receive communication objects over the bus. The object dictionary describes all data types, communication objects and application objects used in this device. It is the interface to the application software. The application program provides the internal control functionality as well as the interface to the process hardware interfaces.

Object Dictionary Layout

Index (hex)	Object
0000	Reserved
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific 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	Standardized Device Profile Area
A000-FFFF	Reserved for further use

© **cia**

The most important part of a CANopen device is the object dictionary. The object dictionary is essentially a grouping of objects accessible via the network in an ordered pre-defined fashion. Each object within the dictionary is addressed using a 16-bit index and an 8-bit sub-index.

The overall layout of the standard object dictionary conforms with other industrial fieldbus concepts.

The object dictionary concept caters for optional device features, which means a manufacturer does not have to provide certain extended functionality on his devices but if he wishes to do so he must do it in a pre-defined fashion.

By defining object dictionary entries for anticipated increased functionality in an optional category manufacturers wishing to implement enhanced functionality will all do so in the same way.

Data Types (1)

0001	DEFTYPE	Boolean
0002	DEFTYPE	Integer8
0003	DEFTYPE	Integer16
0004	DEFTYPE	Integer32
0005	DEFTYPE	Unsigned8
0006	DEFTYPE	Unsigned16
0007	DEFTYPE	Unsigned32
0008	DEFTYPE	Real32
0009	DEFTYPE	Visible_String
000A	DEFTYPE	Octet_String
000B	DEFTYPE	Unicode_String
000C	DEFTYPE	Time_Of_Day
000D	DEFTYPE	Time_Difference
000E	DEFTYPE	Bit_String
000F	DEFTYPE	Domain
0010	DEFTYPE	Integer24
0011	DEFTYPE	Real64
0012	DEFTYPE	Integer40
0013	DEFTYPE	Integer48
0014	DEFTYPE	Integer56
0015	DEFTYPE	Integer64
0016	DEFTYPE	Unsigned24
0017		reserved
0018	DEFTYPE	Unsigned40
0019	DEFTYPE	Unsigned48
001A	DEFTYPE	Unsigned56
001B	DEFTYPE	Unsigned64
001C-001F		reserved

© CIA

The static data types are placed in the object dictionary for definition purposes. Data of basic type BOOLEAN attains the values TRUE or FALSE. Data of basic type INTEGER n has values in the integers. The value range is $-2^{n-1}, \dots, 2^{n-1}-1$ (bit sequences of length n). Data of basic type UNSIGNED n has values in the non-negative integers. The value range is $0, \dots, 2^{n-1}-1$ (bit sequences of length n). Data of basic type FLOAT has values in the real numbers.

The data type VISIBLE STRING is described in the following syntax of data and data type definitions: Unsigned8 - Visible Char, Array of Visible Char - Visible String. The admissible values of data of type Visible Char are 0h and the range from 20h to 7Eh.

The data type OCTET STRING is described in the following syntax of data and data type definitions: Array of Unsigned8 - Octet String.

The data type DATE is defined as bit sequences of length 56 including ms, min, hour, standard or summer time, day of month, day of week, month, year and some reserved values.

The data type TIME OF DAY represents absolute time including time in ms after midnight and number of days since January 1st, 1984.

The data type TIME DIFFERENCES represents a time difference as sum of days and ms.

Data Types (2)

0020	DEFSTRUCT	PDO_Communication_Parameter
0021	DEFSTRUCT	PDO_Mapping
0022	DEFSTRUCT	SDO_Parameter
0023	DEFSTRUCT	Identity
0024-003F		reserved
0040-005F	DEFSTRUCT	Manufacturer Specific Complex Data Types
0060-007F	DEFTYPE	Device Profile (0) Specific Standard Data Types
0080-009F	DEFSTRUCT	Device Profile (0) Specific Complex Data Types
00A0-00BF	DEFTYPE	Device Profile 1 Specific Standard Data Types
00C0-00DF	DEFSTRUCT	Device Profile 1 Specific Complex Data Types
00E0-00FF	DEFTYPE	Device Profile 2 Specific Standard Data Types
0100-011F	DEFSTRUCT	Device Profile 2 Specific Complex Data Types
0120-013F	DEFTYPE	Device Profile 3 Specific Standard Data Types
0140-015F	DEFSTRUCT	Device Profile 3 Specific Complex Data Types
0160-017F	DEFTYPE	Device Profile 4 Specific Standard Data Types
0180-019F	DEFSTRUCT	Device Profile 4 Specific Complex Data Types
01A0-01BF	DEFTYPE	Device Profile 5 Specific Standard Data Types
01C0-01DF	DEFSTRUCT	Device Profile 5 Specific Complex Data Types
01E0-01FF	DEFTYPE	Device Profile 6 Specific Standard Data Types
0200-021F	DEFSTRUCT	Device Profile 6 Specific Complex Data Types
0220-023F	DEFTYPE	Device Profile 7 Specific Standard Data Types
0240-025F	DEFSTRUCT	Device Profile 7 Specific Complex Data Types

© **cia**

CANopen specifies some predefined complex data types for PDO and SDO parameters. In addition, the object dictionary reserves entries for device-specific standard and complex data types. For devices or device profiles that provide Multiple Device Modules like multiple axis controllers each virtual device may use its own data types.

Object Description

Name	short description of usage
Object Code	Variable, Array, Record etc.
Data Type	Unsigned8, Boolean, Integer16, etc.
Category	optional (o) or mandatory (m) or conditional (c)

© **cia**

The Object Code must be one of those defined by the CANopen specification. For simple variables the attribute description appears once without the sub-index field and entry category. For complex data types the attribute description must be defined for each element (sub-index).

Entry Description

Description*	Descriptive name of the Sub-Index
Data Type*	Unsigned8, Boolean, Integer16, etc.
Entry Category*	optional (o), mandatory (m), conditional (c)
Access	read only (ro), write only (wo), read write (rw) or constant
PDO Mapping	no, optional, default
Value Range	the allowed value range for this object
Default Value	value of this object after device initialization
Substitute Value	value of this object if it is not implemented

* only for sub-objects in Arrays and Structures

© CiA

Object Dictionary Entry

Index	Subindex	Variable Accessed	Data Type
6092	0	Number of Entries	Unsigned8
6092	1	Baud Rate	Unsigned16
6092	2	Number of Data Bits	Unsigned8
6092	3	Number of Stop Bits	Unsigned8
6092	4	Parity	Unsigned8

C-Structure Equivalent

```
typedef struct {  
    UNSIGNED8    NumberOfEntries;  
    UNSIGNED16   BaudRate;  
    UNSIGNED8    NumberOfDataBits;  
    UNSIGNED8    NumberOfStopBits;  
    UNSIGNED8    Parity;  
} RS232_T;
```

© **cia**

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

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. For example on a single channel RS-232 interface module there may exist a data-structure at index 6092h which defines the communication parameters for that module. This structure may contain fields for baud-rate, number of data bits, number of stop bits and parity type. The sub-index concept can be used to access these individual fields as shown above. To allow individual elements of structures of data to be accessed via the network a sub-index has been defined. The value for the sub-index is always zero.

Communication Profile

Index (hex)	Object	Name	Type	Acc.	M/O
1000	VAR	device_type	Unsigned32	ro	M
1001	VAR	error_register	Unsigned8	ro	M
1002	VAR	manufacturer_status_register	Unsigned32	ro	O
1003	ARRAY	pre-defined_error_field	Unsigned32	ro	O
1004	Reserved for compatibility reasons				
1005	VAR	COB-ID_SYNC-message	Unsigned32	rw	O
1006	VAR	communication_cycle_period	Unsigned32	rw	O
1007	VAR	synchronous_window_length	Unsigned32	rw	O
1008	VAR	manufacturer_device_name	Vis-String	c	O
1009	VAR	manufacturer_hardware_version	Vis-String	c	O
100A	VAR	manufacturer_software_version	Vis-String	c	O
100B	Reserved for compatibility reasons				
100C	VAR	guard_time	Unsigned32	rw	O
100D	VAR	life_time_factor	Unsigned32	rw	O
100E	Reserved for compatibility reasons				
100F	Reserved for compatibility reasons				
1010	VAR	store_parameters	Unsigned32	rw	O
1011	VAR	restore_default_parameters	Unsigned32	rw	O
1012	VAR	COB-ID_time_stamp	Unsigned32	rw	O
1013	VAR	high_resolution_time_stamp	Unsigned32	rw	O
1014	VAR	COB-ID_Emergency	Unsigned32	rw	O
1015	VAR	Inhibit_Time_Emergency	Unsigned16	rw	O
1016	ARRAY	Consumer_Heartbeat_Time	Unsigned32	rw	O
1017	VAR	Producer_Heartbeat_Time	Unsigned16	rw	O
1018	RECORD	identity_object	Identity	ro	M
.....
11FF	reserved				

© CIA

Error Register

0	1	2	3	4	5	6	7
Generic Error	Current	Voltage	Temperature	Communication Error	Device profile-specific	Reserved (0)	Manufacturer-specific
M	O	O	O	O	O	O	O

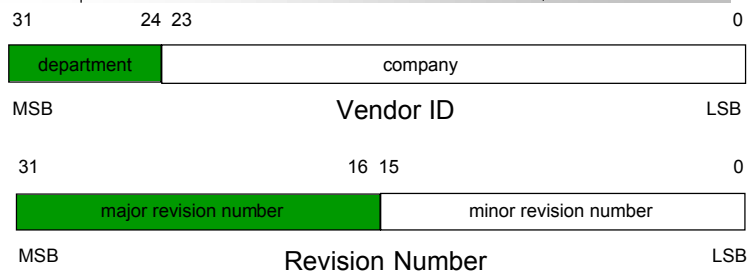
If a bit is set to 1 the specified error has occurred. The only mandatory error that has to be signaled is the generic error. The generic error is signaled at any error situation.

© CiA

The object 1001h is an error register for the device. The device can map internal errors in this byte. This entry is mandatory for all devices. It is a part of the Emergency object.

Identity Object

Index	Subindex	Description	Data Type
1018h	0	Number of Entries	Unsigned8
	1	Vendor ID	Unsigned32
	2	Product Code	Unsigned32
	3	Revision Number	Unsigned32
	4	Serial Number	Unsigned32



© CiA

The mandatory Identity Object at index 1018h contains general information about the virtual device. The Vendor ID is a numeric value of type Unsigned32 and will consist of a unique number for each registered company and may be a unique number for each department of that company (only if required). The allocation of the Vendor ID is handled by CiA Headquarters. Both parts of the Vendor ID must be registered by CiA and will cause an administrative costs of 128 EUR + German VAT. For CiA Members this service is free of charge.

The manufacturer-specific Product code identifies a specific device version. The manufacturer-specific Revision number consists of a major revision number (identifies a specific CANopen behavior) and a minor revision number (identifies different versions of the same CANopen behavior). If the CANopen functionality is expanded, the major revision number has to be increased.

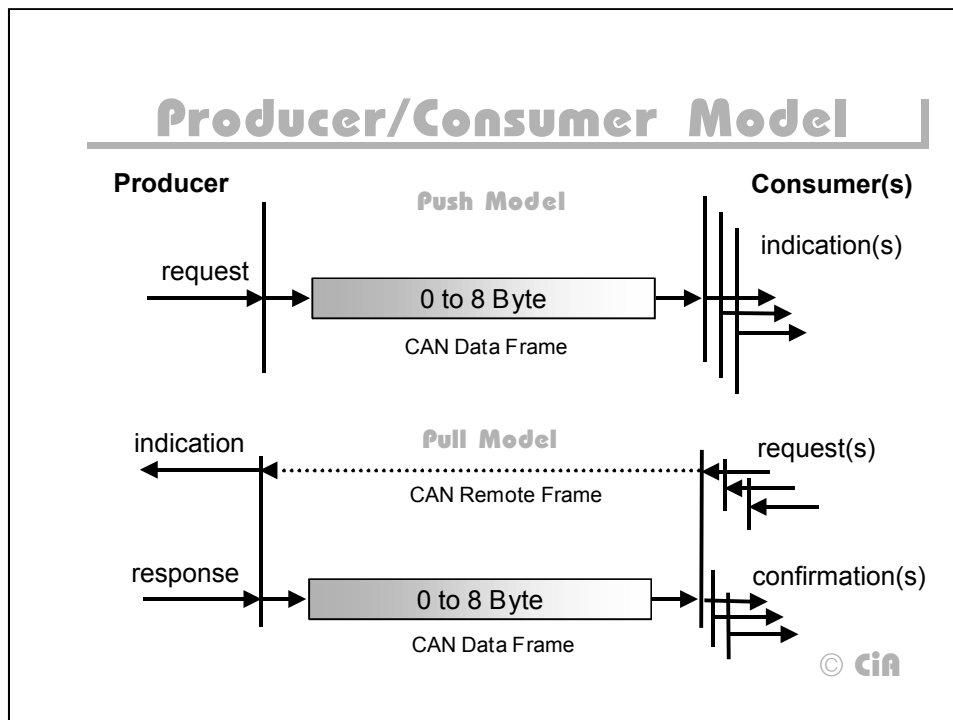
Communication Objects

- ◆ Process Data Objects (PDO)
- ◆ Service Data Object (SDO)
- ◆ Special Function Objects:
 - Synchronization Object (SYNC)
 - Time Stamp Object
 - Emergency Object (EMCY)
- ◆ Network Management Objects:
 - NMT Message
 - Boot-Up Object
 - Error Control Object

© **cia**

CANopen communication objects described by the services and protocols. They are classified as follows:

- The real-time data transfer is performed by means of Process Data Objects (PDOs).
- With Service Data Objects (SDOs) the read and write access to entries of a device object dictionary is provided.
- Special Function Objects provide application-specific network synchronization, time stamping and emergency messages.
- The Network Management (NMT) Objects provide services for network initialization, error control and device status control.

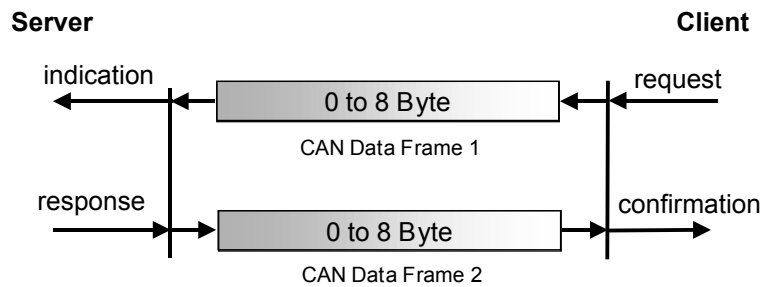


The Producer/Consumer model describes perfectly the CAN broadcast communication capability. Each station of the network can listen to the messages of the transmitting station. After receiving the message it is the task of every node to decide if the message has to be accepted or not. So Acceptance Filtering has to be implemented in a CAN node.

The CAN broadcast communication is similar with a radio station transmitting information about traffic jam for vehicle drivers. Every driver has to decide if the messages are important for him depending on the direction he wants to go.

The Producer/Consumer model allows to services: to transmit a messages (push model) or to request a message (pull model).

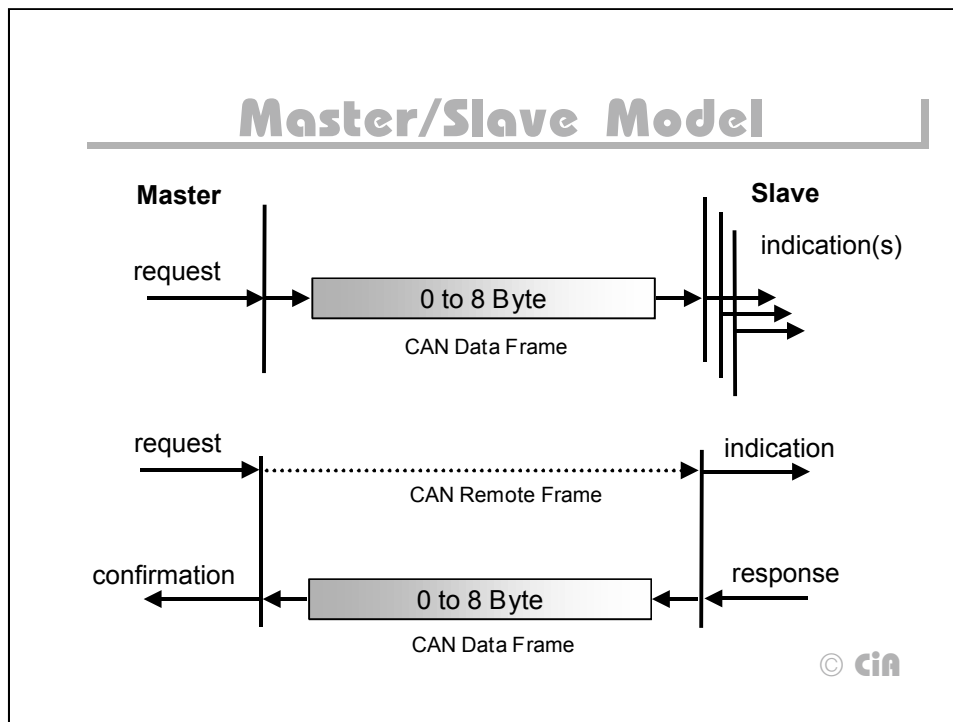
Client/Server Model



© **cia**

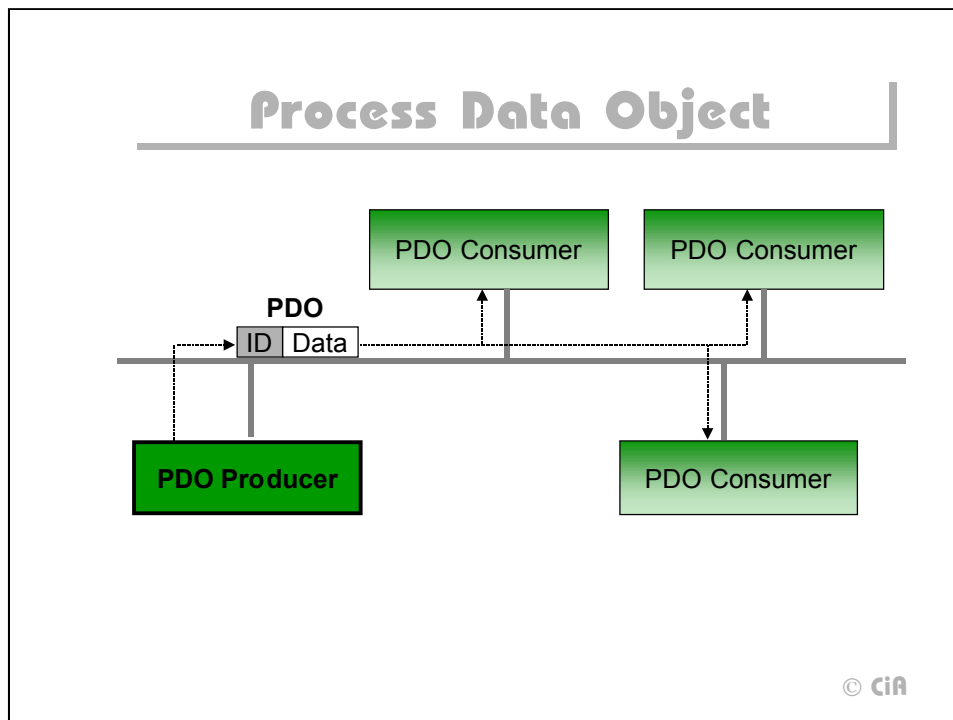
In the classical Client/Server model the client transmits a message, which will be responded by the server, so that the client gets a confirmation. That is like to give an order, which must be confirmed, so that you know the order was understood.

The Client/Server model is used to transfer data longer than 8 bytes. Therefore the original data to be transmitted is segmented and sent segment for segment on the same identifier. Each or a group of segments or the total message will be confirmed by the receiver. So this is a peer-to-peer communication. The services on this client/server transmission may include up- and download as well as transfer abort.



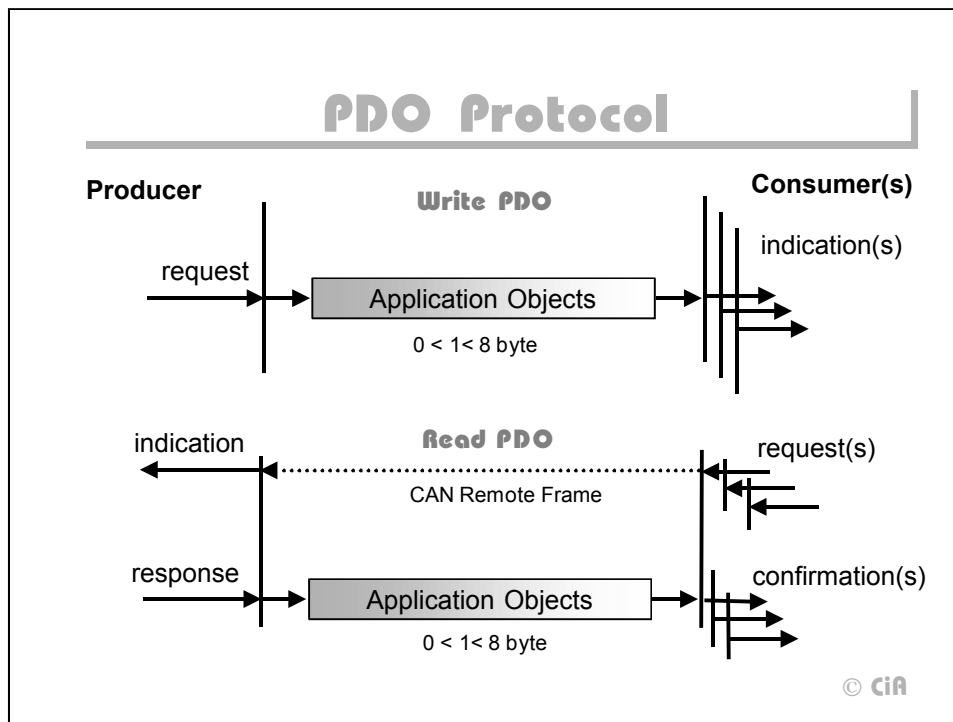
The Master/Slave model allows only a communication initiated by the master, The slave has always to wait for the master's communication requests. This is like in the army: the soldier will get orders and he has only to speak after permission.

In CAN-based networks master/slave communication can be implemented by an appropriate identifier allocation. Unconfirmed Master/slave communication allows also broadcasting.



PDO communication can be described by the producer/consumer model. Process data can be transmitted from one device (producer) to one another device (consumer) or to many other devices (broadcasting). PDOs are transmitted in a non-confirmed mode.

The producer sends a Transmit-PDO (T_P DO) with a specific identifier, which corresponds to the identifier of the Receive-PDO (R_PDO) of one or more consumers.



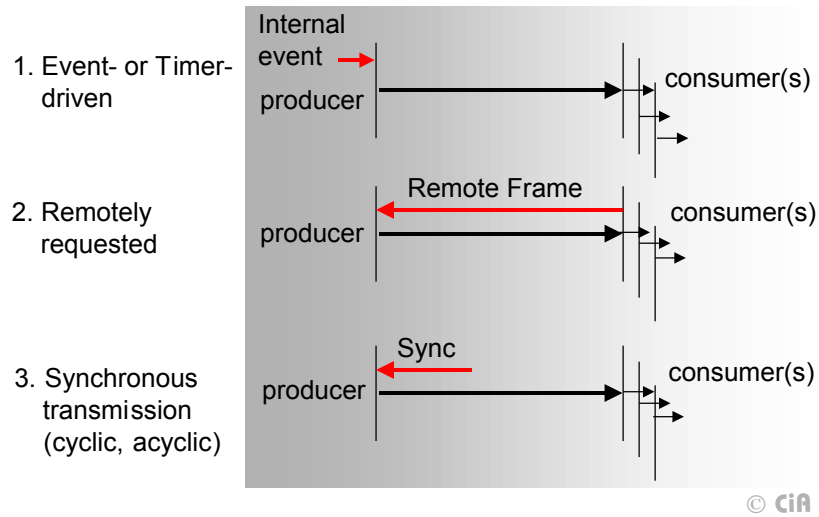
There are two PDO services: to write a PDO and to read a PDO.

The Write-PDO is mapped to a single CAN Data frame. The Read-PDO is mapped to CAN Remote Frame, which will be responded by the corresponding CAN Data Frame. Read-PDOs are optional and depending on the device capability.

The complete data field of up to 8 byte may contain process data. Number and length of PDOs of a device are application-specific and have to be specified in the device profile.

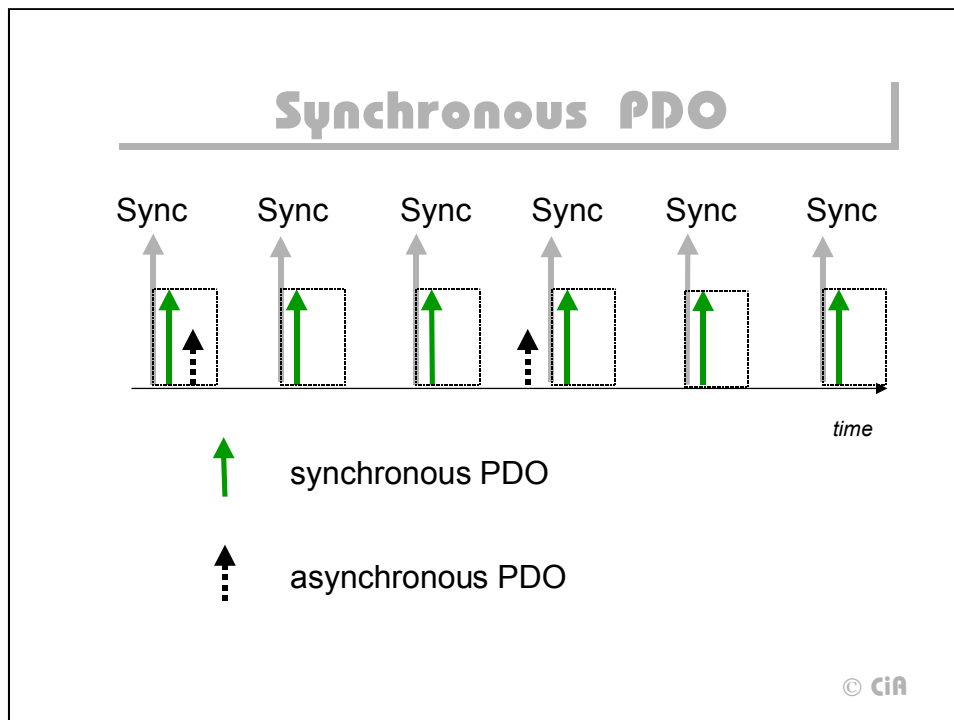
The PDOs correspond to entries in the Object Dictionary and provide the interface to application objects. Data type and mapping of application objects into a PDO is determined by a corresponding default PDO mapping structure within the Object Dictionary. This structure is defined in the entries 1600h for the 1st R_PDO and 1A00h for the first T_PDO. In one CANopen network there can be used up to 512 T_PDOs and 512 R_PDOs.

PDO Communication Modes



The CANopen communication profile distinguishes three message triggering modes:

1. Message transmission is triggered by the occurrence of an object specific-event specified in the device profile. Periodically transmitting nodes are triggered additionally by elapsed timer even if no event has occurred.
2. The transmission of asynchronous PDOs may be initiated on receipt of a remote request initiated by another device.
3. Synchronous PDOs are triggered by the expiration of a specified transmission period synchronized by the reception of the SYNC object.

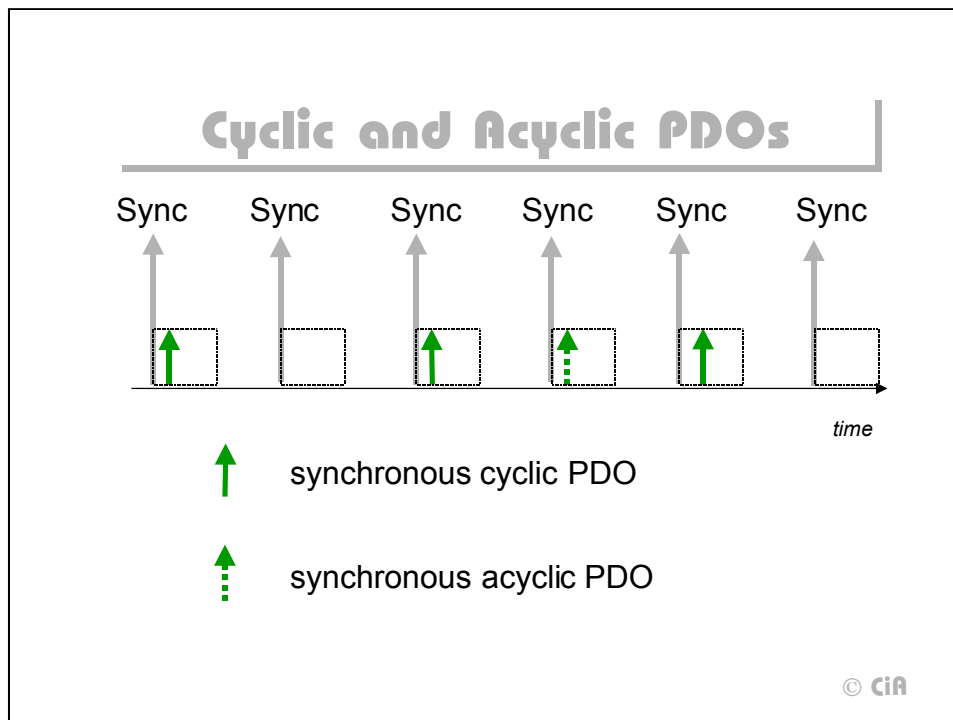


CANopen distinguishes the following transmission modes:

- synchronous transmission,
- asynchronous transmission.

Synchronous PDOs are transmitted within the synchronous window after the Sync Object. The priority of synchronous PDOs should be higher than the priority of asynchronous PDOs.

Asynchronous PDOs and SDOs can be transmitted at every time with respect to their priority. So they could also be transmitted within the synchronous window.



The transmission of synchronous PDOs can be subdivided into cyclic and acyclic transmission mode.

Synchronous cyclic PDOs are transmitted within the synchronous window. The number of the transmission type (1 to 240) indicates the number of Sync objects between two PDO transmissions.

Acyclically transmitted synchronous PDOs are triggered by an application specific event. The message shall be transmitted synchronously with the Sync object but not periodically.

PDO Transmission Types

Type No.	cyclic	acyclic	synchronous	asynchronous	RTR only
0		x	x		
1-240 ¹	x		x		
241-251			reserved		
252 ²			x		x
253 ³				x	x
254 ⁴				x	
255 ⁵				x	

¹ the type indicates the number of SYNC objects between two PDO transmissions

² data is updated (but not sent) immediately after reception of the SYNC

³ data is updated at the reception of the RTR

⁴ application event is device-specific

⁵ application event is defined in the device profile

© **cia**

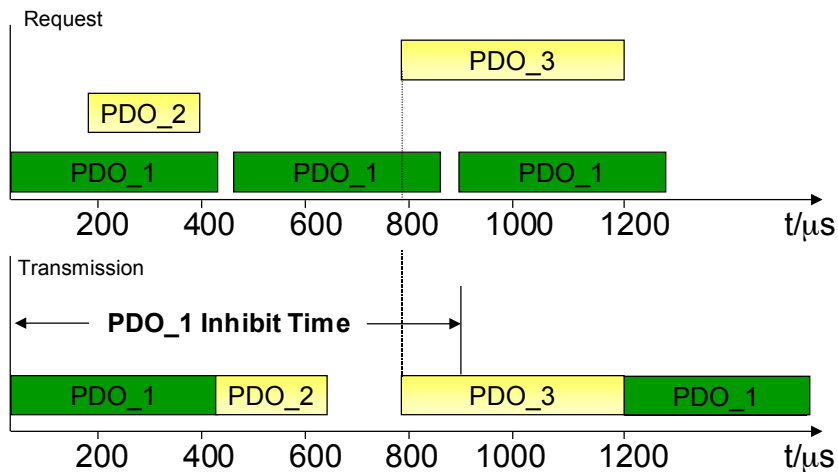
Synchronous (transmission types 0-240 and 252) means that the transmission of the PDO shall be related to the SYNC object.

Acyclic (transmission type 0) means that the message shall be transmitted synchronously with the SYNC object but not periodically.

Preferably the devices use the SYNC as a trigger to output or actuate based on the previous synchronous Receive-PDO respectively to update the data transmitted at the following synchronous Transmit-PDO. Details of this mechanism depend on the device type and are defined in the device profile if applicable.

The transmission type of a PDO is defined in the PDO communication parameter index (1400h for the 1st R_PDO or 1800h for the 1st T_PDO).

PDO Parameter 'Inhibit Time'



© **cia**

To guarantee that no starvation on the network occurs for communication objects with low priorities, PDOs can be assigned an inhibit time. The inhibit-time defines the minimum time that has to elapse between two consecutive invocations of a PDOs service.

In the example above PDO 1 has got a higher priority than PDO 2 and PDO 3. The inhibit-time allows the second and the third PDO to get bus access although having lower priority than the first PDO.

PDO Parameter

Receive PDO Communication Parameter (20H)					
1400	RECORD	1 st receive PDO Parameter	PDOCommPar	rw	M/O*
1401	RECORD	2 nd receive PDO Parameter	PDOCommPar	rw	M/O*
.....
15FF	RECORD	512 th receive PDO Parameter	PDOCommPar	rw	M/O*
Receive PDO Mapping Parameter (21H)					
1600	ARRAY	1 st receive PDO mapping	PDOMapping	rw	M/O*
1601	ARRAY	2 nd receive PDO mapping	PDOMapping	rw	M/O*
.....
17FF	ARRAY	512 th receive PDO mapping	PDOMapping	rw	M/O*
Transmit PDO Communication Parameter (20H)					
1800	RECORD	1 st transmit PDO Parameter	PDOCommPar	rw	M/O*
1801	RECORD	2 nd transmit PDO Parameter	PDOCommPar	rw	M/O*
.....
19FF	RECORD	512 th transmit PDO Parameter	PDOCommPar	rw	M/O*
Transmit PDO Mapping Parameter (21H)					
1A00	ARRAY	1 st transmit PDO mapping	PDOMapping	rw	M/O*
1A01	ARRAY	2 nd transmit PDO mapping	PDOMapping	rw	M/O*
.....
1BFF	ARRAY	512 th transmit PDO mapping	PDOMapping	rw	M/O*

* If a device supports PDOs, the according PDO communication parameter and PDO mapping entries in the object dictionary are mandatory. These may be read_only.

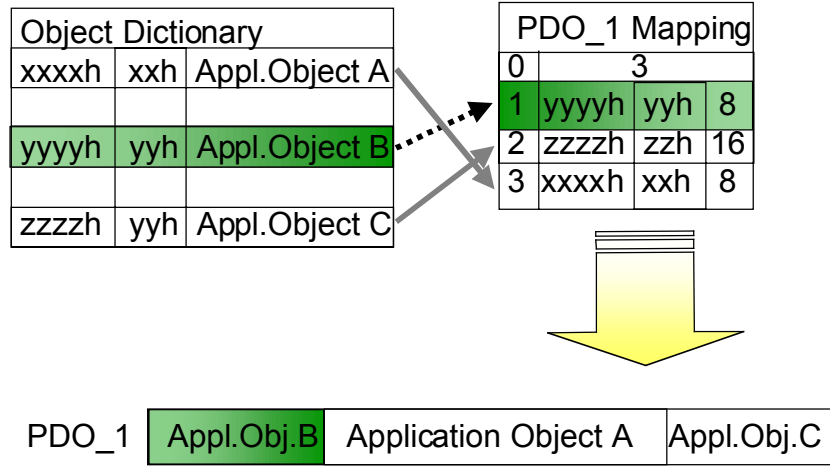
Communication Parameter

Index	Sub-Index	Description	Data Type
0020h	0h	number of entries	Unsigned8
	1h	COB-ID	Unsigned32
	2h	transmission type	Unsigned8
	3h	inhibit time	Unsigned16
	4h	reserved	Unsigned8
	5h	event timer	Unsigned16

© **cia**

The communication parameters describe the communication behavior of a PDO. The communication parameter set is defined by index 20h in the object dictionary.

PDO Mapping



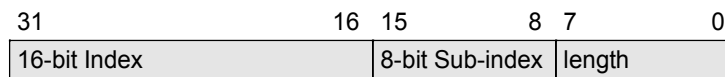
© ciA

The PDO Mapping Parameter defines the content of the Process Data Objects. In the CANopen Device Profiles may be specified a default PDO mapping. Optionally a variable PDO mapping may be supported. If a device supports variable mapping PDO transfer can be optimized application-specificly.

Mapping Parameter

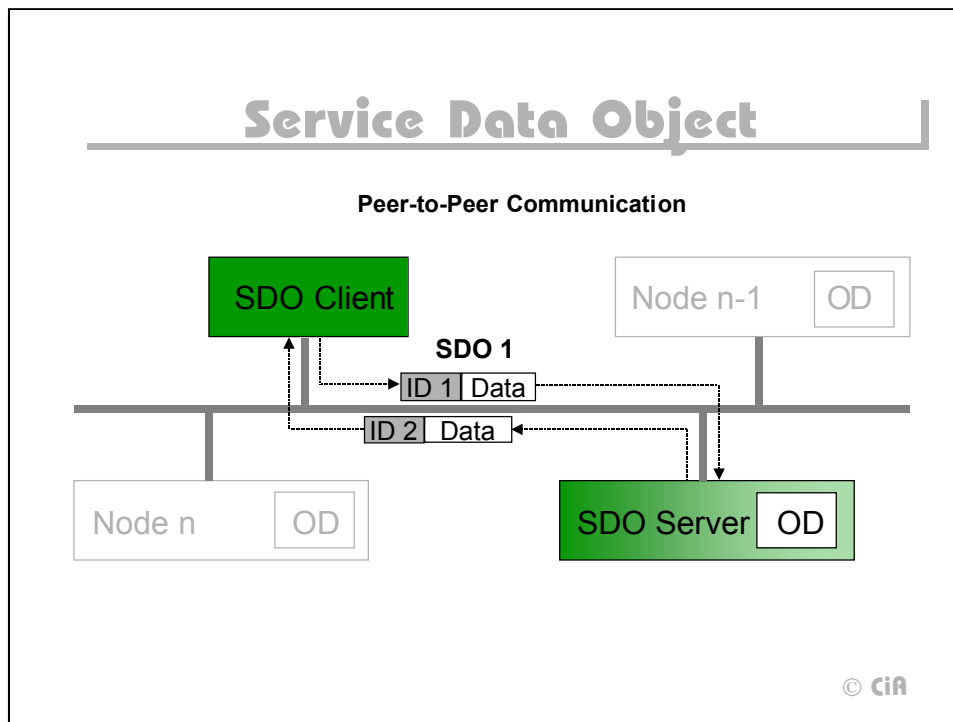
Index	Sub-Index	Description	Data Type
0021h	0h	number of mapped	Unsigned8
	1h	1st object	Unsigned32
	2h	2nd object	Unsigned32

	40h	64th object	Unsigned32

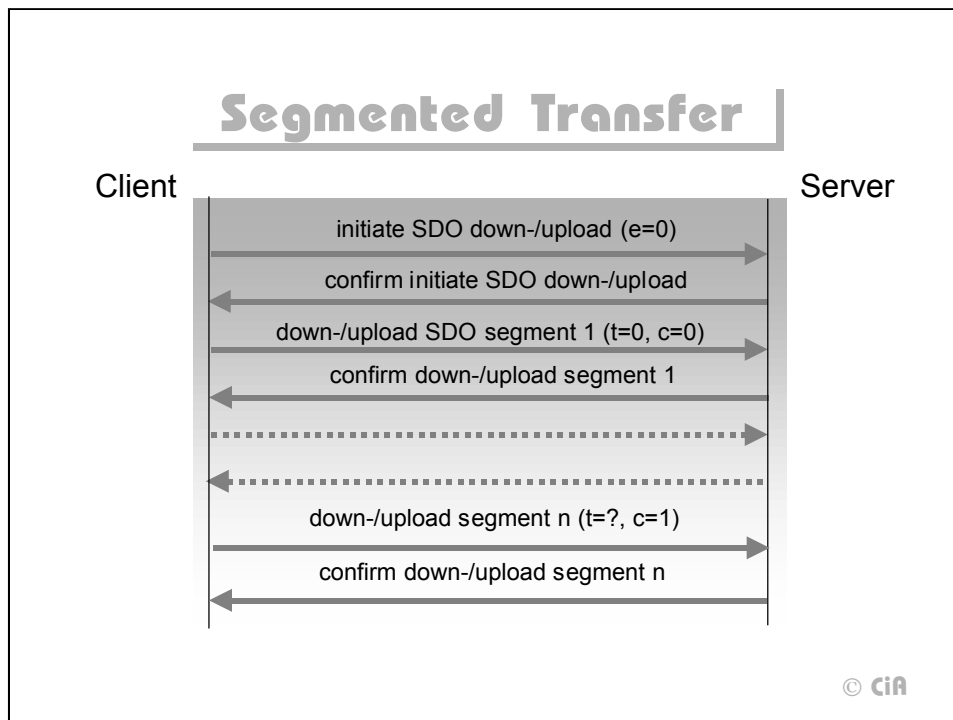


© **cia**

The mapping parameter set described in 21h specifies, which application objects are mapped into a PDO. In maximum, there may be mapped up to 64 objects.



With Service Data Objects (SDO) the access to entries of a device object dictionary is provided. One SDO uses two CAN Data Frames with different identifiers, because the communication is confirmed. By means of a SDO a peer-to-peer communication channel between two devices may be established. The owner of the accessed object dictionary is the server of the SDO. A device may support more than one SDO. One supported SDO is mandatory and the default case.

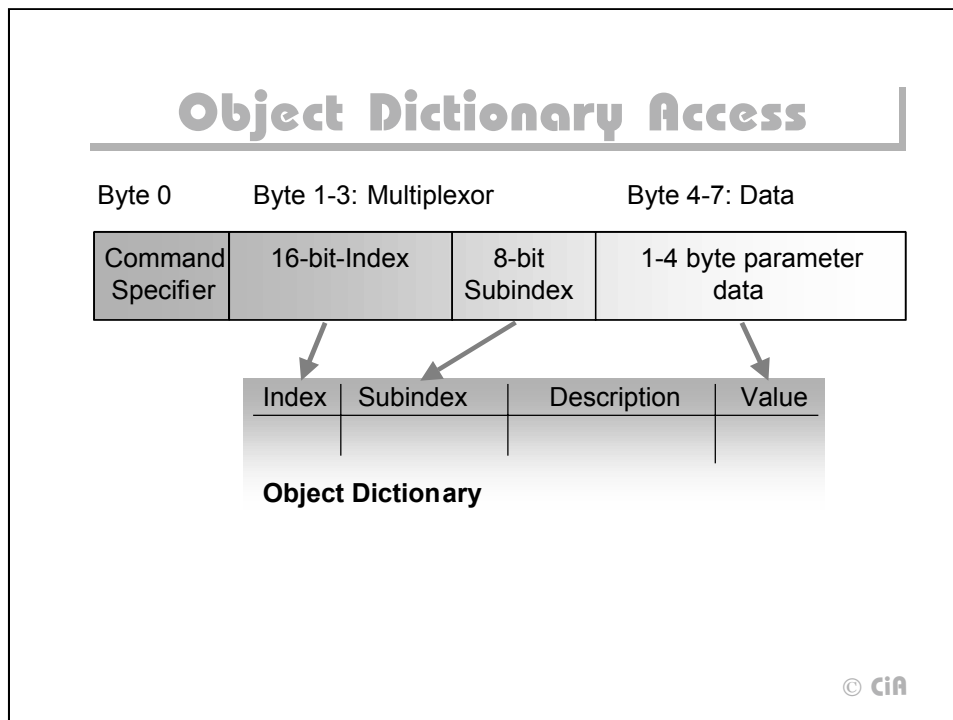


SDOs allow to transfer data of any size. They are transferred as a sequence of segments. For transmission of messages with data length less than 5 bytes an 'expedited' transfer may be performed by means of the 'Initiate Domain Down/Upload' protocols. With these protocols the data transmission is performed within the initiate protocol.

The transfer of messages of more than 4 data bytes has to be performed by means of the segmented transfer. This permits transfer of data of any length since the data can be split up over several CAN messages. All segments after the SDO's first CAN message may each contain seven bytes of useful data. The last segment may contain an end indicator.

An SDO is transferred in 'confirmed' mode that is the reception of each segment is acknowledged by a corresponding CAN message. It is always the client that takes the initiative for a transfer. The owner of the accessed Object Dictionary is the server of the SDO. Both client and server can take the initiative to abort the transfer of a domain.

Optionally a SDO can be transferred as a sequence of blocks. Each block is a sequence of up to 127 segments containing a sequence number and the data, but is confirmed by only one message.

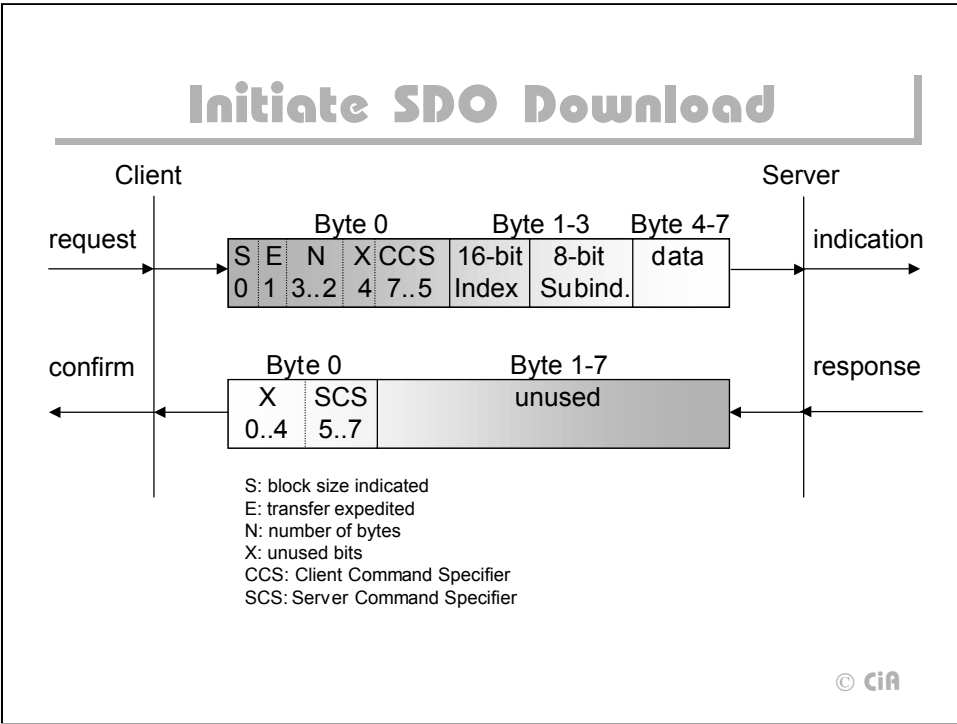


Read and write access to the CANopen object dictionary is performed by SDOs. The Client/Server Command Specifier contains the following information:

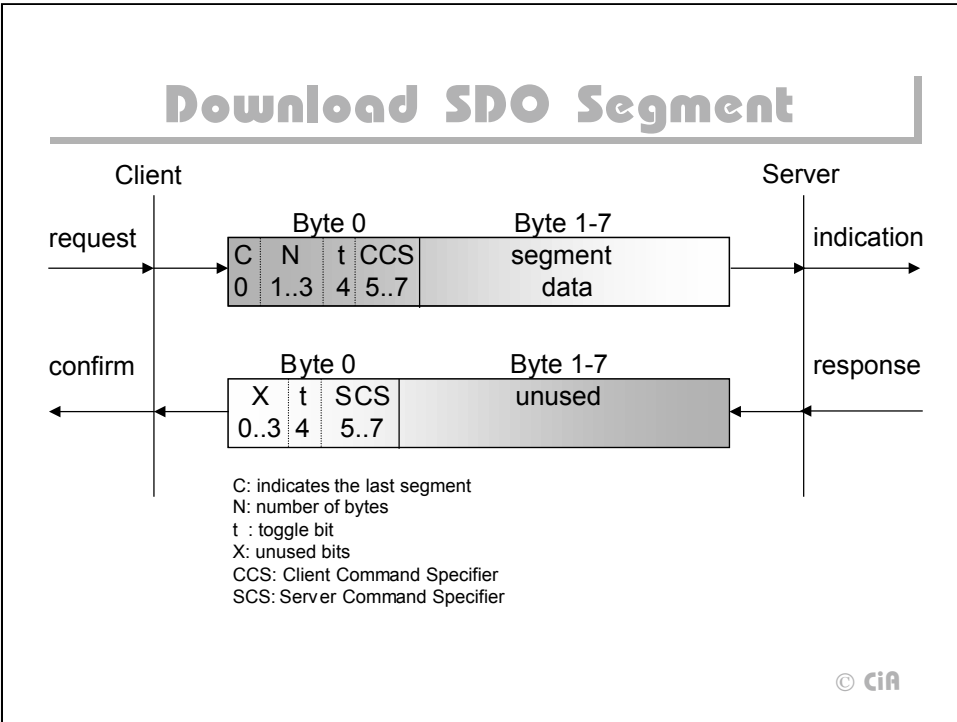
- download/upload,
- request/response,
- segmented/block/expedited transfer,
- number of data bytes,
- end indicator,
- alternating toggle bit for each subsequent segment.

SDOs are described by the communication parameter. The default Server-SDO (S_SDO) is defined in the entry 1200h, and the first Client-SDO is specified in the entry 1280h.

In one CANopen network there may be used up to 256 SDO channels requiring two CAN identifiers each.



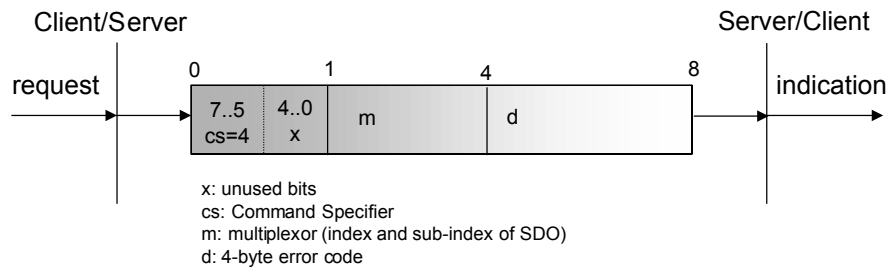
When reading or writing to the Object Dictionary of the Server device, the service Initiate SDO Down-/Upload is the first to be invoked. It is important to note that the data bytes are transmitted with the most significant bit first.



For each SDO segment up-/downloaded, two CAN Data Frames are exchanged between Client and Server. The request carries the segment of data, as well as segmentation control information. The response acknowledges the reception of each segment.

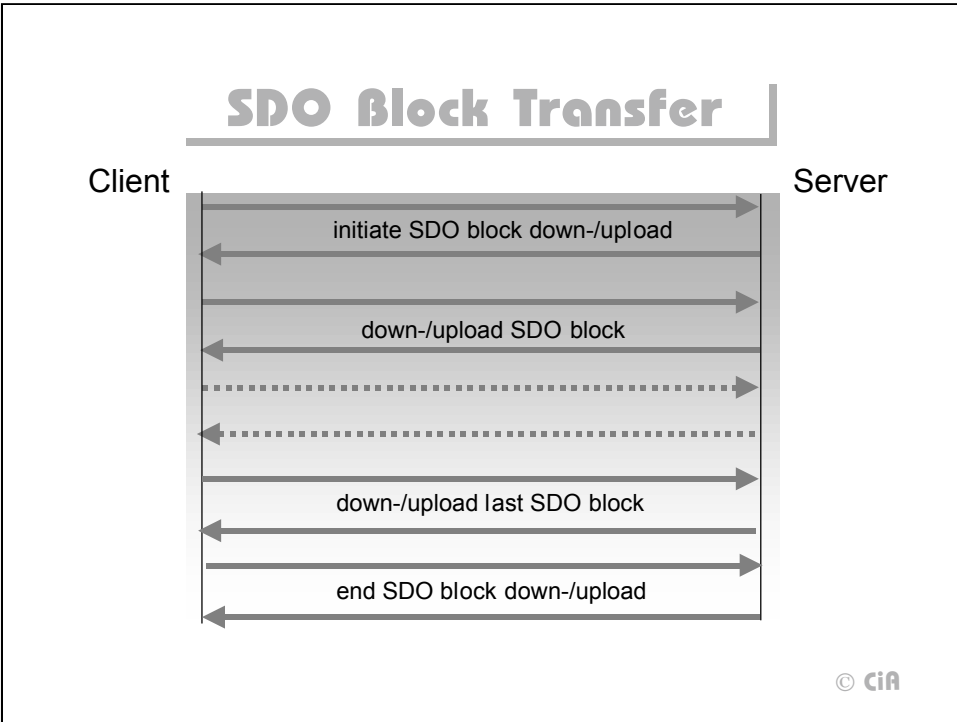
Bit t is a toggle bit and it is toggled on successive Domain Segment telegrams. For the first segment it has an initial value of zero.

Abort SDO Transfer

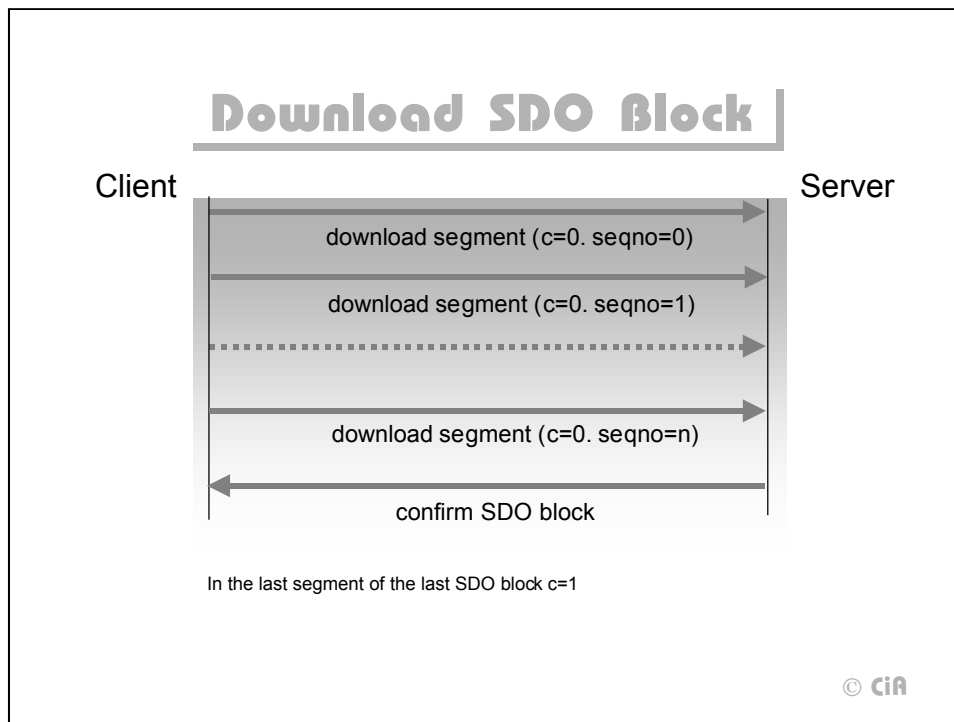


© CiA

The Abort SDO Transfer is used to notify either the Client or the Server of SDO transfer errors. The reason for the invocation of this service is indicated as a set of error codes, stored in the fields Error Class, Error Code and Additional Code.

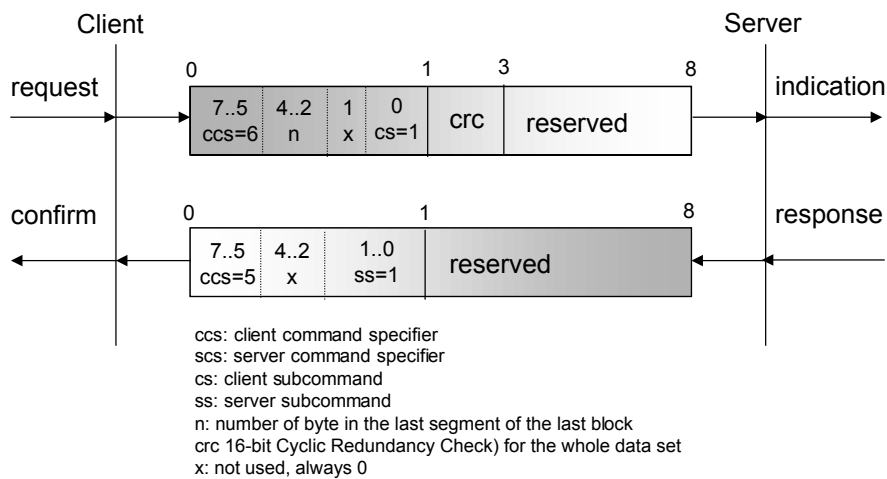


The SDO Down-/Upload Block Protocol is initiated by the Initiate SDO Block Down-/Upload messages followed by a SDO Down-/Upload Block sequence. The protocol is terminated by the End SDO Block Down-/Upload messages.



The Download SDO Block is a sequence of up to 127 segments confirmed by one message. If in the confirmation message the c-bit is set to 1, the block download sequence was received successfully. Unsuccessful completion is indicated by an Abort SDO Transfer request/indication.

End SDO Block Download



© CiA

To verify the correctness of a SDO Block down-/upload client and server calculating a CRC which is exchanged and verified during the End SDO Block Down-/Upload messages. The check polynomial has the formula $x^{16} + x^{15} + x^5 + 1$. The CRC generator has to be initialized to 0 before calculating the checksum. The checksum can be calculated or a polynome table can be used.

SDO Parameter

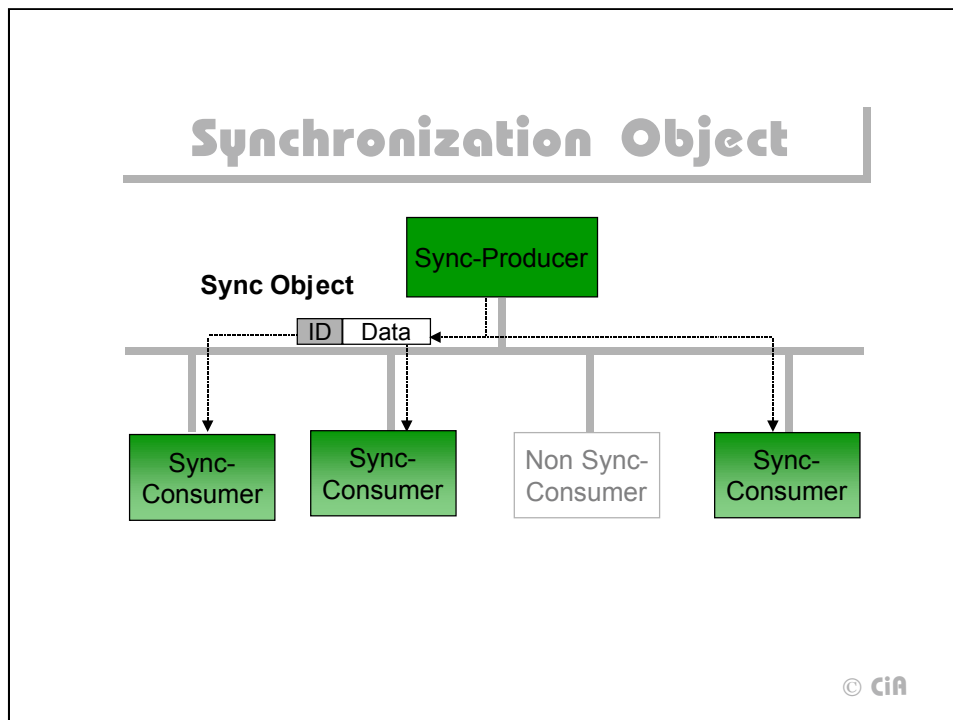
Server SDO Parameter (22H)					
1200	RECORD	1 st Server SDO parameter	SDOParameter	ro	0
1201	RECORD	2 nd Server SDO parameter	SDOParameter	rw	M/O**
.....
127F	RECORD	128 th Server SDO parameter	SDOParameter	rw	M/O**
Client SDO Parameter (22H)					
1280	RECORD	1 st Client SDO parameter	SDOParameter	rw	M/O**
1281	RECORD	2 nd Client SDO parameter	SDOParameter	rw	M/O**
.....
12FF	RECORD	128 th Server SDO parameter	SDOParameter	rw	M/O**
1300		reserved			
.....
13FF		reserved			

** If a device supports SDOs, the according SDO parameters in the Object Dictionary are mandatory

SDO Parameter Record

Index	Sub-Index	Description	Data Type
0022h	0h	number of entries	Unsigned8
	1h	COB-ID client -> server	Unsigned32
	2h	COB-ID client <- server	Unsigned32
	3h	node ID	Unsigned8

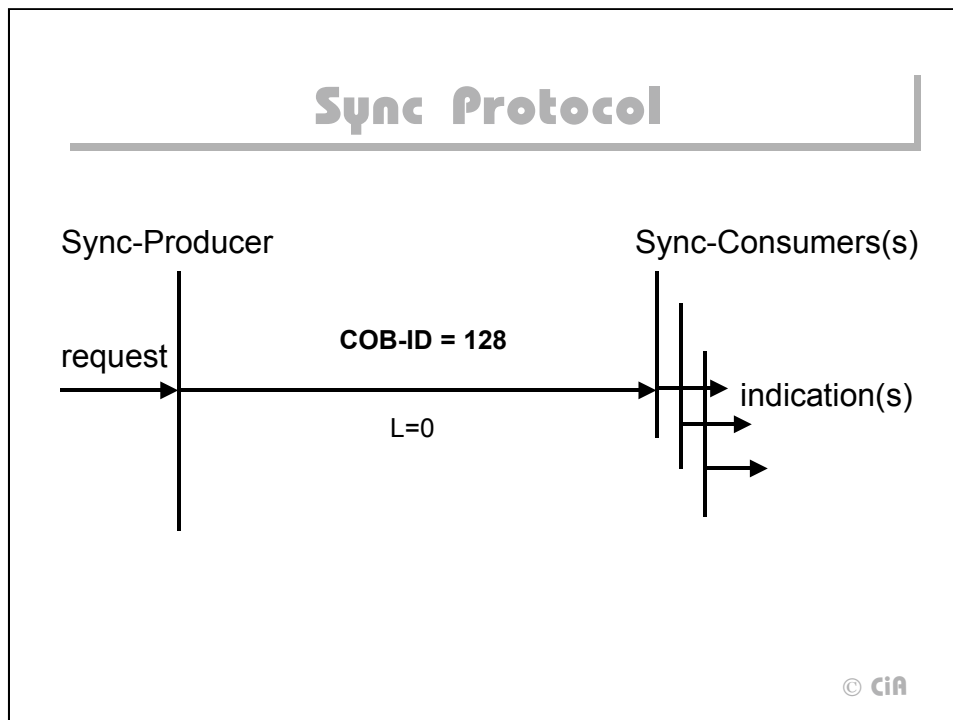
© **cia**



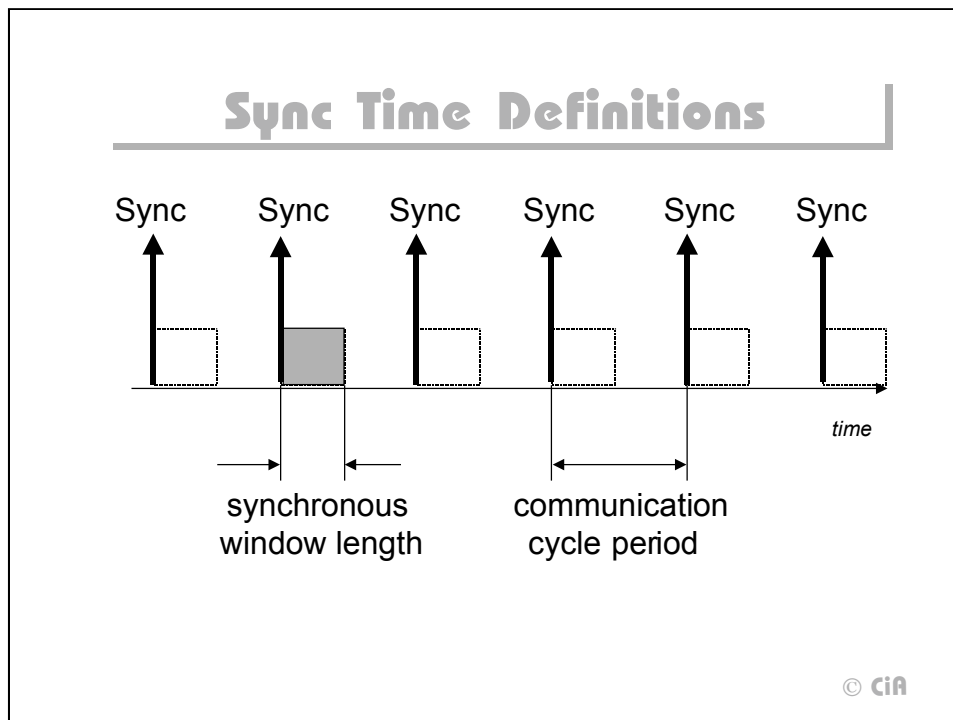
The Sync-Producer provides the synchronization-signal for the Sync-Consumer. When the Sync-Consumer receive the signal they start carrying out their synchronous tasks.

In general the fixing of the transmission time of synchronous PDO messages coupled with the periodicity of transmission of the Sync Object guarantees that sensor devices may arrange to sample process variables and that actuator devices may apply their actuation in a coordinated fashion.

The identifier of the Sync Object is available at index 1005h.

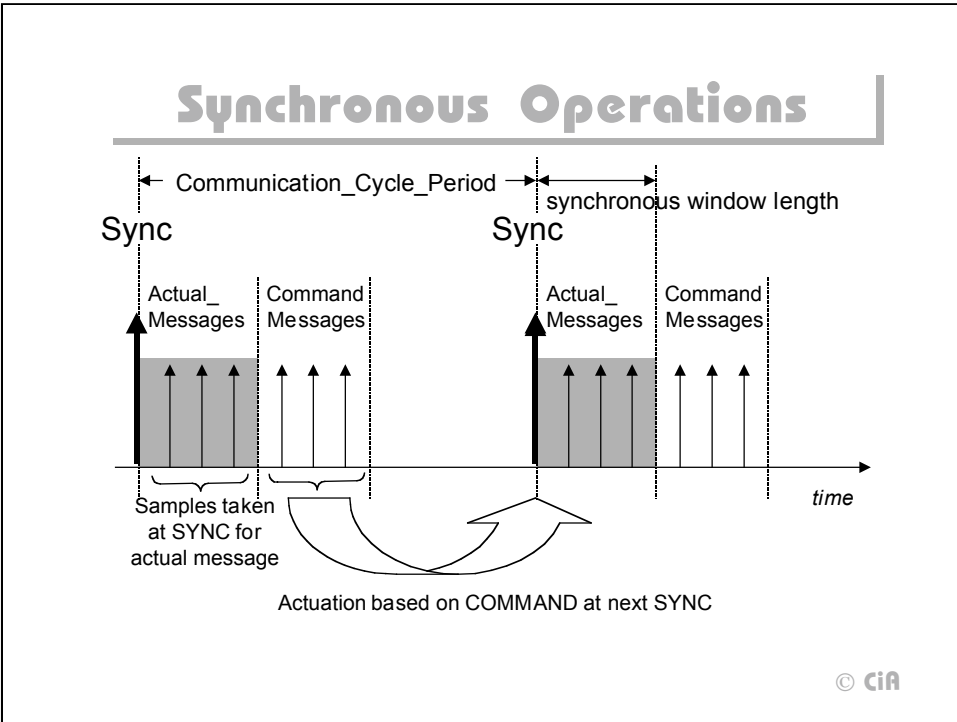


The Sync Object is implemented with the synchronization device acting as the producer of that object. In order to guarantee timely access to the CAN bus the Sync Object is given a very high priority identifier. CANopen suggests the identifier 128 which is a value of the highest priority group used in the pre-defined connection set. Within the Sync object no application data are transported. By default, the Sync Object does not carry any data.

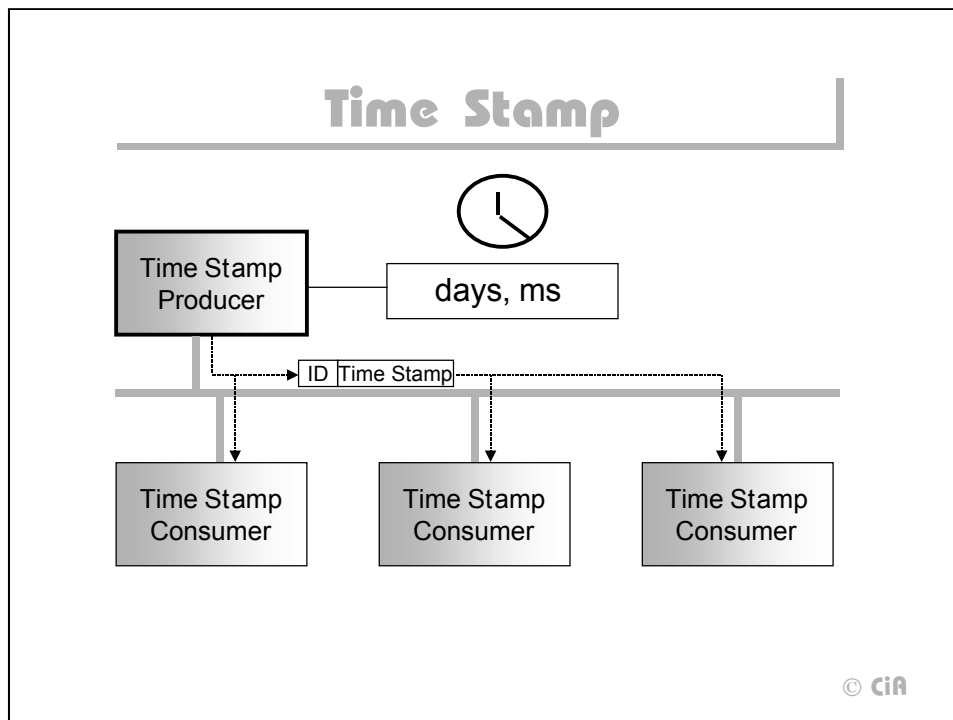


Synchronous transmission of a PDO means that the transmission is fixed in time with respect to the transmission of the Sync Object. The synchronous PDO is transmitted within a given time window 'synchronous window length' with respect to the Sync transmission, and at most once for every period of the Sync. The time period between the Sync objects is specified by the parameter 'communication cycle period'. The 'synchronous window length' (1007h) and the 'communication cycle period' (1006h) are specified in the Object Dictionary, which may be written by a configuration tool to the application devices during the boot-up process.

There can be a time jitter in transmission by the Sync-Producer corresponding approximately to the latency due to another message being transmitted just before the Sync Object. You also have to consider the case, that the Sync is transmitted twice.



The cyclically transmitted Sync telegram may cause the CANopen devices that support this function to store the actual values of the inputs quasi-simultaneously. They are then transferred to all interested devices in the following time frame. In the following time frame, the CANopen devices transmit the output values to the actuators. However, these values are not valid until the next Sync message is received.

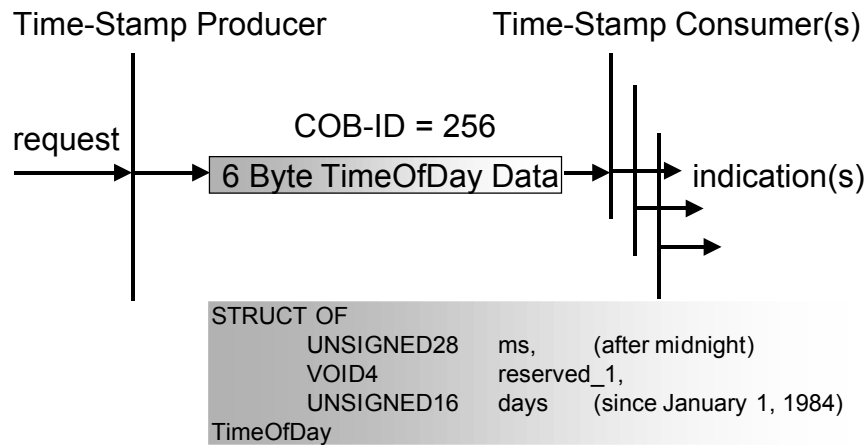


Usually the Time-Stamp object represents an absolute time in ms after midnight and the number of days since January 1, 1984. This is a bit sequence of length 48 (6 byte).

Some time critical applications especially in large networks with reduced transmission rates require very accurate synchronization; it may be necessary to synchronize the local clocks with an accuracy in the order of microseconds. This is achieved by using the optional high resolution synchronization protocol which employs a special form of time stamp message to adjust the inevitable drift of the local clocks.

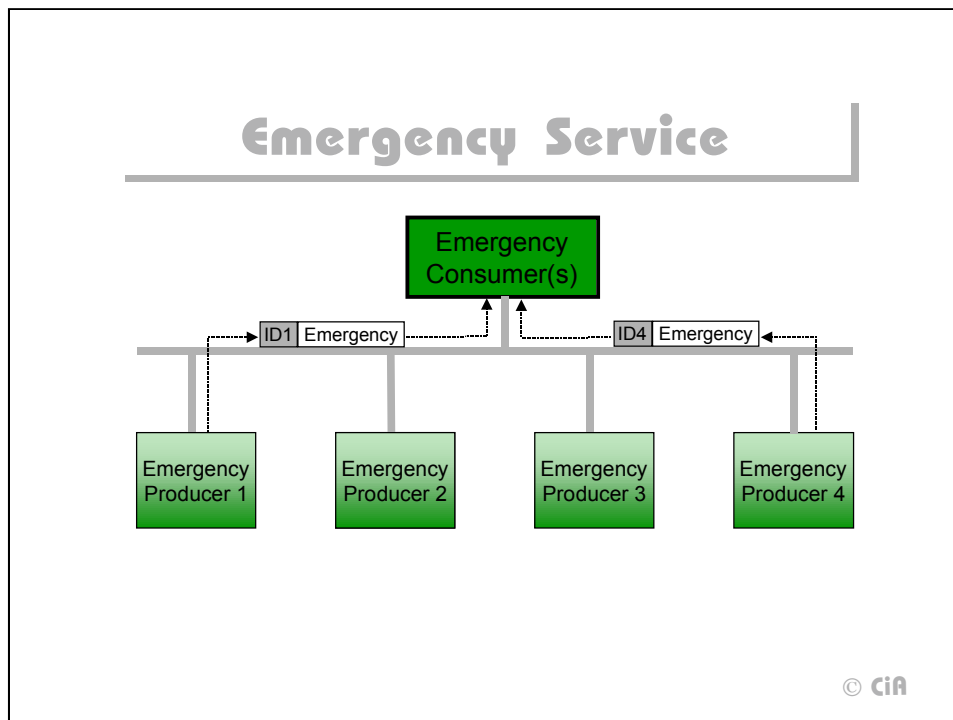
The high-resolution time-stamp is encoded as unsigned32 with a resolution of 1 microsecond which means that the time counter restarts every 72 minutes. It is configured by mapping the high resolution time-stamp (object 1013h) into a PDO.

Time-Stamp Object



© CiA

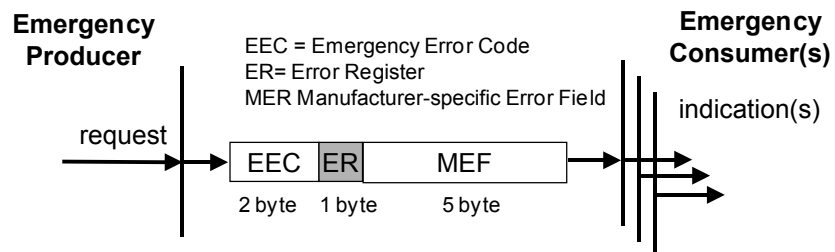
The Time-Stamp object is implemented with the transmitting device acting as the producer and the receiving devices acting as consumers of the event. For the Time Stamp Message identifier 256 is suggested.



Emergency messages are triggered by the occurrence of a device internal fatal error situation and are transmitted from the concerned application device to the other devices with high priority. This makes them suitable for interrupt type error alerts. An Emergency Telegram may be sent only once per 'error event', i.e. the emergency messages must not be repeated. As long as no new errors occur on a device no further emergency message must be sent.

By means of CANopen Communication Profile defined emergency error codes, the error register and device specific additional information are specified in the device profiles.

Emergency Object



Name: EMCY000
Error Register: Object 1001h
Function Code: 0001
Pre-defined COB-ID: 129 to 255

© **cia**

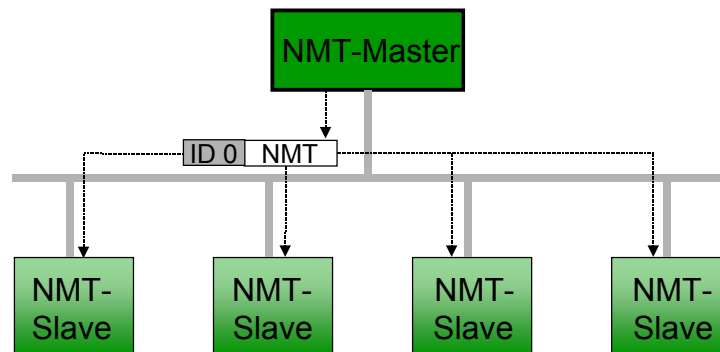
The Emergency Object is optional. If a device supports this object, it has to support at least the two error codes 00xx (error reset or no error) and 11xx (generic error). The Emergency Object contains 8 data bytes and is confirmed transmitted.

Emergency Error Code

00xx	Error Reset or No Error	60xx	Device Software
10xx	Generic Error	61xx	internal
20xx	Current	62xx	user
21xx	device input side	63xx	data set
22xx	inside of device	70xx	Additional Modules
23xx	device output side	80xx	Monitoring
30xx	Voltage	81xx	communication
31xx	main	8110	CAN overrun
32xx	inside of device	8120	Error Passive (EP)
33xx	output	8130	Life Guard Error
40xx	Temperature	8140	recovered from Bus-off
41xx	ambient	82xx	Protocol Error
42xx	device	8210	PDO not processed
50xx	Device Hardware	8220	length exceeded
		90xx	External Error
		F0xx	Additional Functions
		FFxx	Device Specific

© CiA

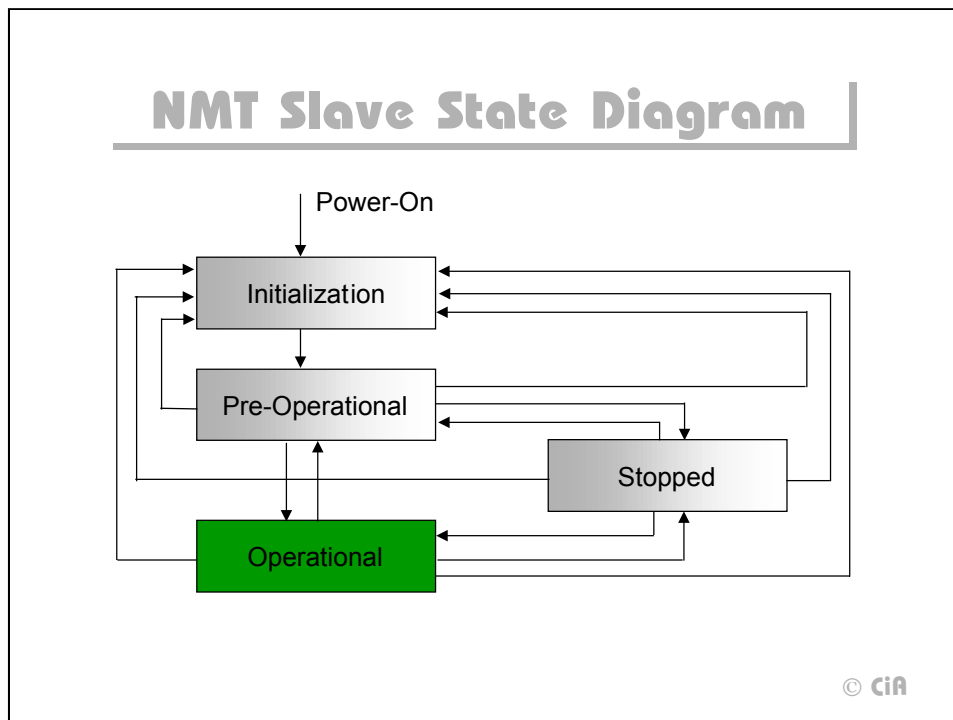
Network Management (NMT)



© ciA

The CANopen network management is node-oriented and follows a master/slave structure. It requires one device in the network, which fulfills the function of the NMT Master. The other nodes are NMT Slaves. The network management provides the following functionality groups: Module Control Services for initialization of NMT Slaves that want to take part in the distributed application, Error Control Services for supervision of the nodes and networks communication status, Configuration Control Services for up- and downloading of configuration data from respectively to a module of the network.

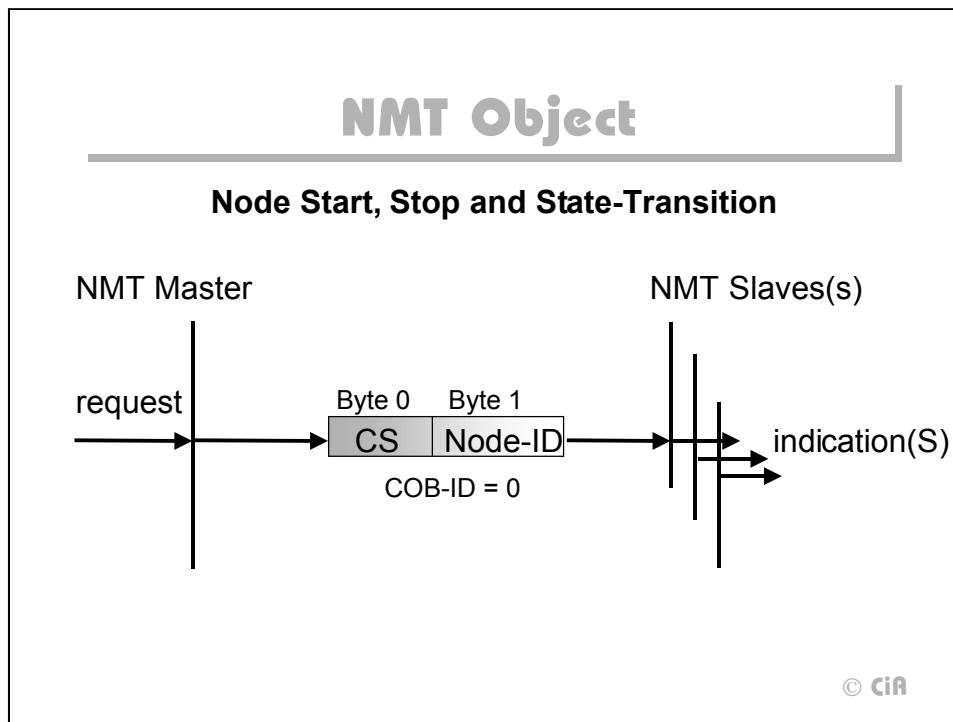
A NMT Slave represents that part of a node, which is responsible for the node's NMT functionality. A NMT Slave is uniquely identified by its module ID.



The CANopen NMT Slave devices implement a state machine, which brings automatically after power-on and internal initialization every device in Pre-operational state. In this state the node may be configured and parameterized via SDO (e.g. using a configuration tool), no PDO communication is allowed.

The NMT Master device may switch all nodes or a single node to Operational state and vice versa. In Operational state PDO transfer is allowed. By switching a device into the Stopped state it is forced to stop PDO and SDO communication. Furthermore, this state can be used to achieve certain application behavior. The definition of this behavior falls into the scope of device profiles.

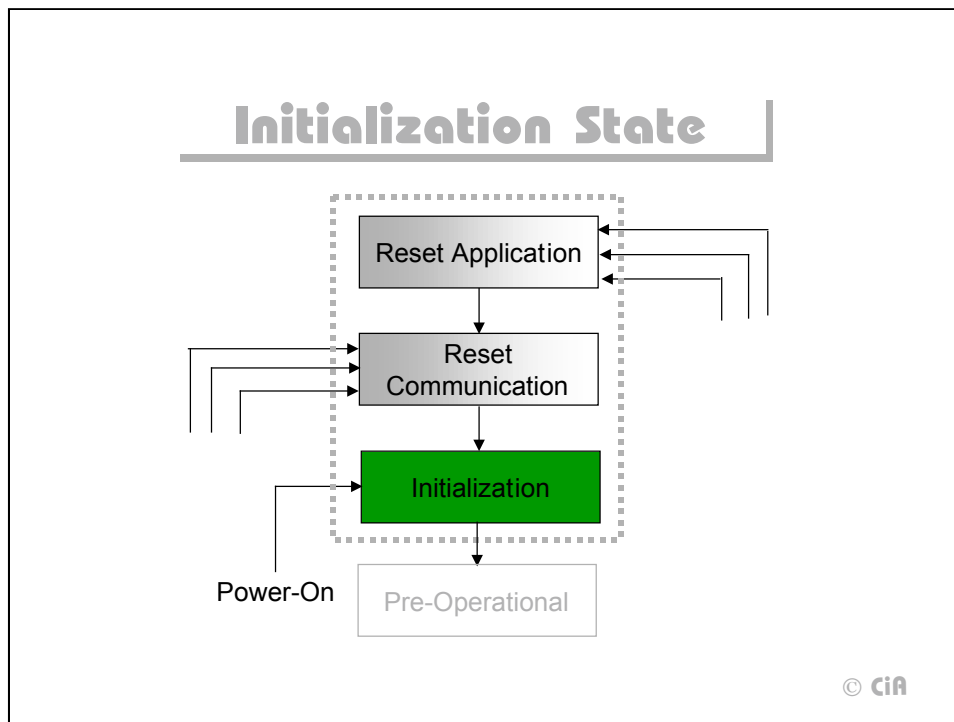
In the Operational state all communication objects are active. Object Dictionary access via SDO is possible. Implementation aspects or the application state machine however may require to switch off or to read only certain application objects whilst being operational (e.g. an object may contain the application program, which cannot be changed during execution).



CANopen Network Management provides the following five services, which can be distinguished by the Command Specifier (CS):

- Start Remote Node (CS=1),
- Stop Remote Node (CS=2),
- Enter Pre-Operational (CS=128),
- Reset Node (CS=129) and
- Reset Communication (CS=130).

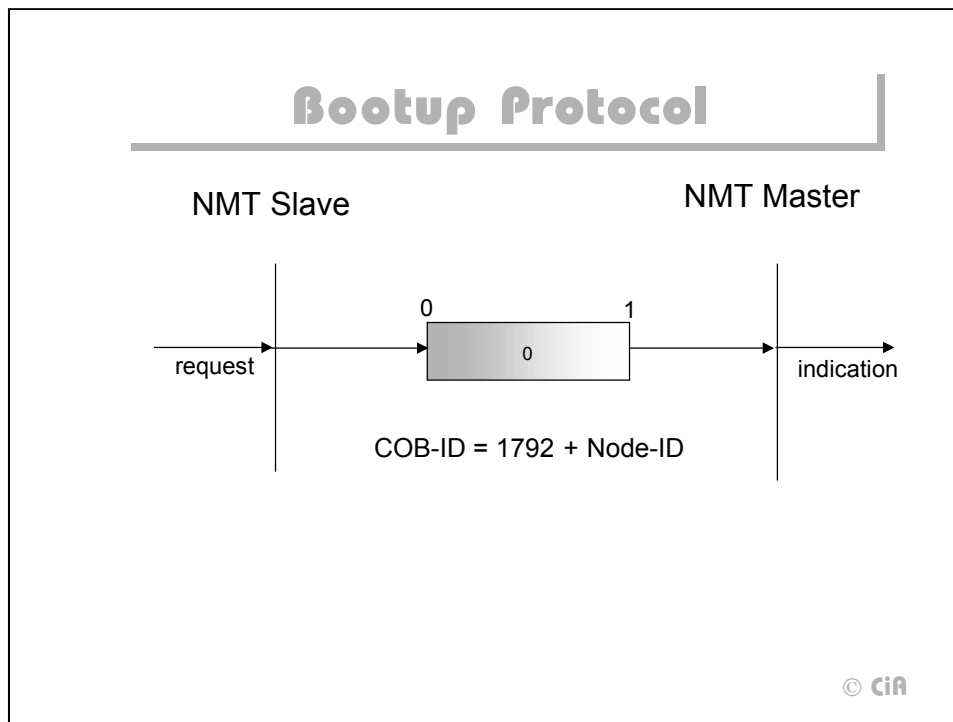
The communication object has got the identifier=0 and consists out of two bytes. The Node-ID defines the destination of the message. If it is zero the protocol addresses all NMT slaves.



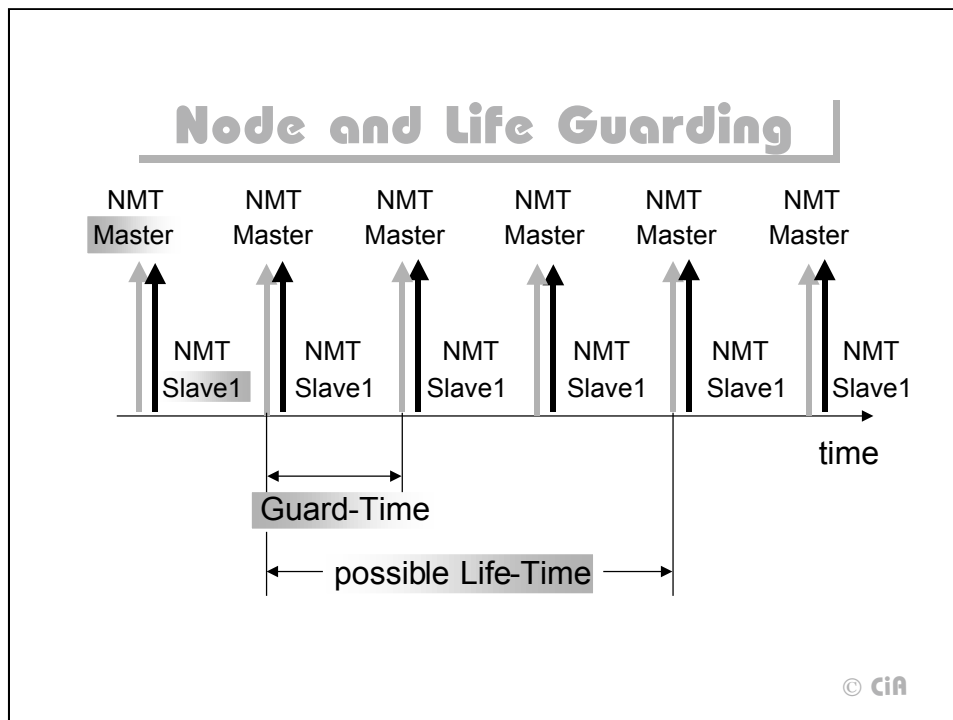
The Initialization state is divided into three sub-states in order to enable a complete or partial reset of a node. In the Reset Application sub-state the parameters of the manufacturer-specific profile area and of the standardized device profile area are set to their default values. After setting of the power-on values the Reset Communication sub-state is entered autonomously.

In the Reset Communication sub-state the parameters of the communication profile are set to their power-on values. After this state the Initializing sub-state is entered. In this sub-state the basic device initialization is executed. Before entering the Pre-Operational the standardized Boot-up Object (Version 4.0) is transmitted.

Power-on values are the last stored parameters. If no parameter storing is supported or if the reset was preceded by a `restore_default` command (Object 101h), the power-on values are the default values according to the communication and device profile.



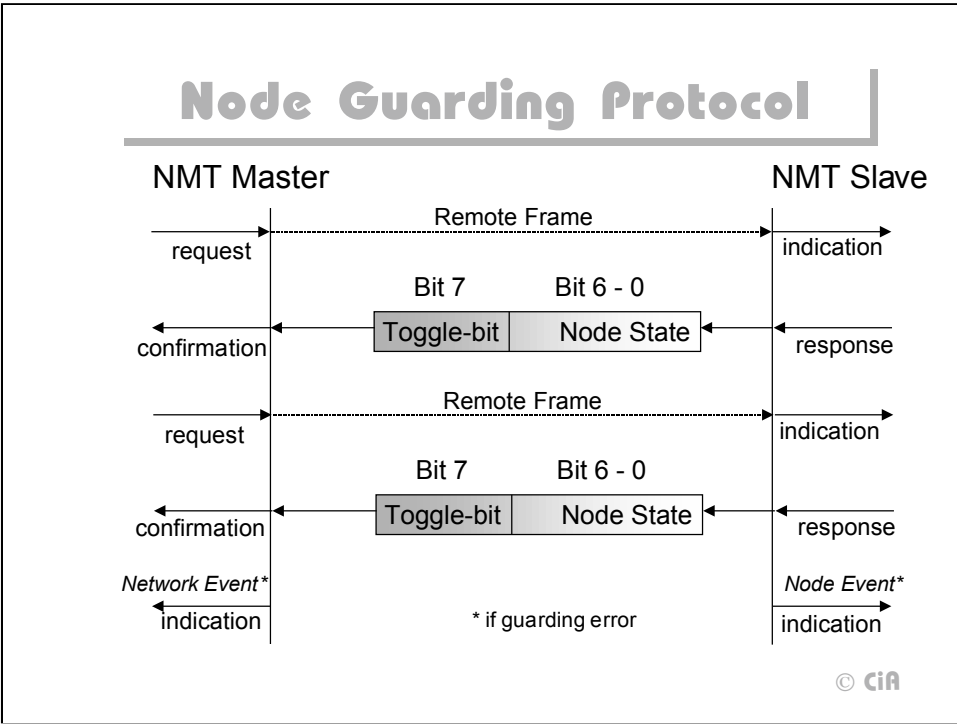
This protocol is used to signal that a NMT slave has entered the node state Pre-Operational after the state Initializing. The protocol uses the same identifier as the error control protocols. The boot-up messages is transmitted also after reset-communication, reset-application and recovering from power-off. The 1-byte data field has a fixed value of zero.



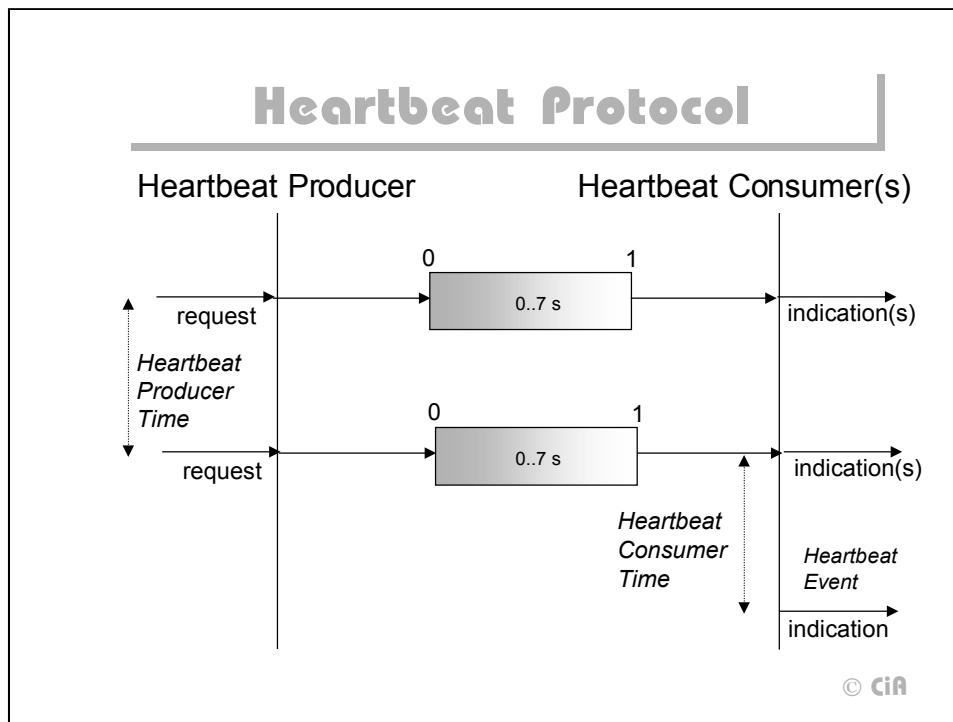
To detect absent devices (e.g. because of bus-off) that do not transmit PDOs regularly, the NMT Master can manage a database, where besides other information the expected states of all connected devices are recorded, which is known as Node Guarding. With cyclic node guarding the NMT master regularly polls its NMT slaves.

To detect the absence of the NMT master, the slaves test internally, whether the Node Guarding is taking place in the defined time interval (Life Guarding).

The Node Guarding is initiated by the NMT Master in Pre-Operational state of the slave by transmitting a Remote Frame. Node Guarding is also in the Stopped Mode active.



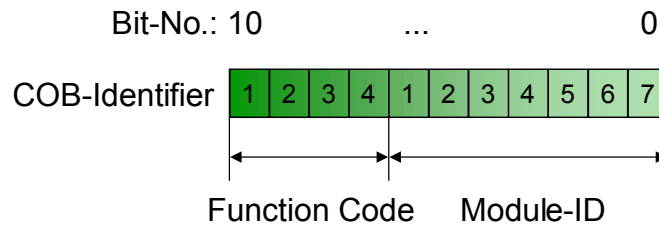
The NMT Master regularly retrieves the actual states of all devices on the network by a Remote Frame and compares them to the states recorded in the network database. Mismatches are indicated first locally on the NMT Master through the Network Event Service. Consequently the application must take appropriate actions to ensure that all devices on the bus will get to a save state.



The optional heartbeat protocol should substitute the life/node guarding protocol. It is highly recommend to implement for new device designs the heartbeat protocol.

A Heartbeat Producer transmits the Heartbeat message cyclically with the frequency defined in Heartbeat producer time object. One or more Heartbeat Consumer may receive the indication. The relationship between producer and consumer is configurable via Object Dictionary entries. The Heartbeat Consumer guards the reception of the Heartbeat within the Heartbeat consumer time. If the Heartbeat is not received within this time a Heartbeat Event will be generated.

Identifier Allocation Scheme



© **cia**

In order to reduce configuration effort for simple networks CANopen defines a mandatory default identifier allocation scheme. These pre-defined identifiers are available in the Pre-Operational state directly after initialization (if no modifications have been stored).

The default ID-allocation scheme consists of a functional part (Function Code) and a Module-ID part, which allows to distinguish between devices. The Module-ID may be assigned by DIP switches, serial interface, or LMT services (Layer Management).

Predefined Connection Set

object	function code (binary)	resulting COB-ID	Communication Parameters at Index
NMT	0000	0	-
SYNC	0001	128 (80h)	1005h, 1006h, 1007h
TIME STAMP	0010	256 (100h)	1012h, 1013h
EMERGENCY	0001	129 (81h) – 255 (FFh)	1014h, 1015h
PDO1 (tx)	0011	385 (181h) – 511 (1FFh)	1800h
PDO1 (rx)	0100	513 (201h) – 639 (639h)	1400h
PDO 2 (tx)	0101	641 (281h) – 767 (2FFh)	1801h
PDO2 (rx)	0110	769 (301h) – 895 (37Fh)	1401h
PDO3 (tx)	0111	897 (381h) – 1023 (3FFh)	1802h
PDO3 (rx)	1000	1025 (401h) – 1151 (47Fh)	1402h
PDO4 (tx)	1001	1153 (481h) – 1279 (4FFh)	1803h
PDO4 (rx)	1010	1281 (501h) – 1407 (57Fh)	1403h
SDO (tx)	1011	1409 (581h) – 1535 (5FFh)	1200h
SDO (rx)	1100	1537 (601h) – 1663 (67Fh)	1200h
NMT Error Control	1110	1793 (701h) – 1919 (77Fh)	1016h, 1017h

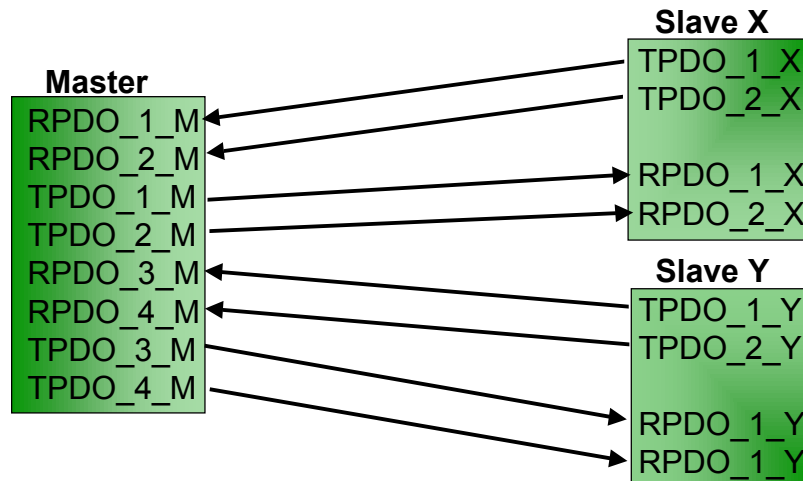
© **cia**

This ID allocation scheme allows a peer-to-peer communication between a single master device and up to 127 slave devices. It also supports the broadcasting of non-confirmed NMT-services, SYNC- and Time-Stamp-objects and node guarding.

The pre-defined master/slave connection set supports one emergency object, one SDO, and at maximum 4 Receive-PDOs and 4 Transmit-PDOs and the node guarding object.

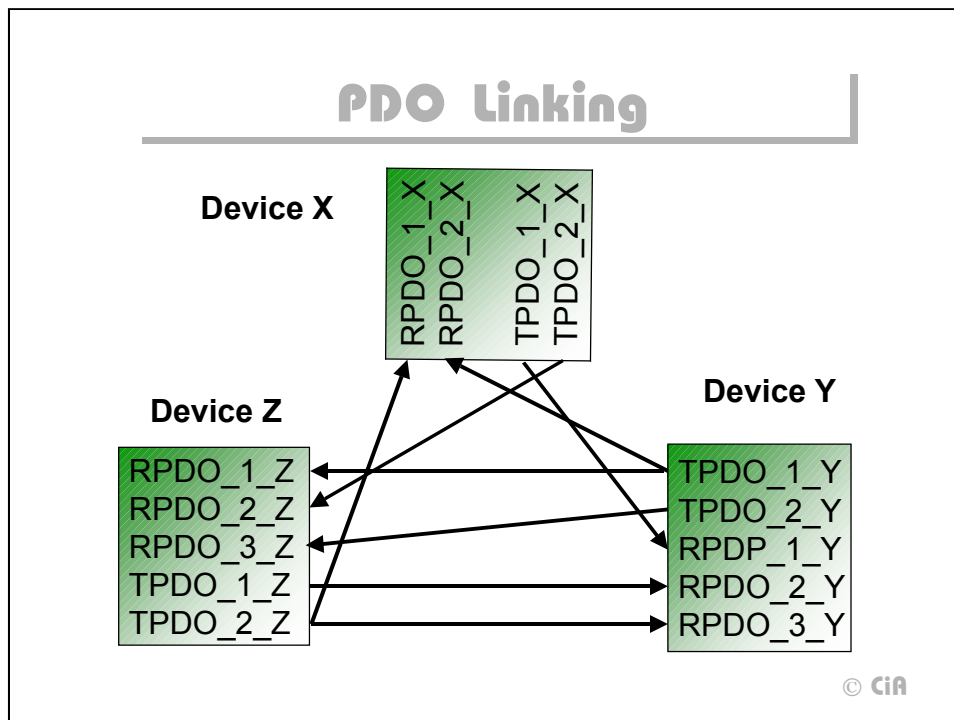
The Predefined Master/Slave Connection Set predefines some CAN Identifiers, others are free. If the connected devices support PDO Linking and variable identifier allocation, the system integrator can assign to the communication objects any identifier values. There are only the identifiers for NMT service (0), Default SDOs (1405 .. 1535 and 1537 ... 1663), NMT Error Control Messages (1793 .. 1919) fix assigned and can't be changed. The identifiers 2015 .. 2031 are reserved for NMT, LMT and DBT services and should not be used for other purposes. Also the identifier range from 257 .. 384 are reserved for Safety-relevant Data Objects (SRDO) specified in the CANopen framework for safety-relevant communication.

Predefined PDO Connections



© ciA

The predefined Master/Slave connection set requires a master device for distribution of the slaves' PDOs. For performing this service the slaves' Transmit PDOs (TPDOs) are connected to the master's Receive PDOs (RPDOs) and the master's Transmit PDOs (TPDOs) are connected to the slaves' Receive PDOs (RPDOs).



PDO Linking enables to establish flexible Multi-Master connections. These connections can be realized during Pre-Operational State by using comfortable configuration tools. The changes in the object dictionaries will be done automatically by the tools using SDOs. In the example above Device Z and Device Y can communicate directly without any master device because the Receive- and Transmit-PDOs are linked.

I/O Object Linking

	Module 1 Output 1	Module 1 Output 2	Module 2 Output 1	Module 2 Output 1	Module 3 Output 1
Module 1 Input 1					
Module 4 Input 1					
Module 4 Input 2			linked		
Module 5 Input 1					
Module 6 Input 1					

© CiA

If users do not want to optimize PDO communication, they can use tools providing I/O object linking capability. The tool configures the related CANopen devices via SDO communication by updating the PDP Mapping Parameters.

CANopen Frameworks

- **CiA DSP-302**
Framework for Programmable Devices
- **CiA DSP-304**
Framework for Safety-Relevant Communication
- **CiA DSP-30X**
Framework for Maritime Electronics
- **CiA DSP-30X**
Framework for Integrated Operating Theaters

© CiA

The DSP-302 framework for programmable CANopen devices is already published and describes features, which have been leaven open in the CANopen communication profile.

The DSP-304 Framework for safety-relevant communication specifies Safety-Relevant Data Objects (SDRO) and a Global Emergency Switch-Off message in compliance with the European approval organizations.

The CANopen framework for maritime electronics will define implementation guidelines for redundant networks as well as physical layer requirements and general requirements of CANopen networks to be used in maritime applications.

The CANopen framework for integrated operating theaters will specify an off-the-shelf plug-and-play capability and interchangeability capability of medical instruments and systems used in operating theaters or other medical applications.

CANopen Manager

- **NMT Master**

Usually the NMT master application is part of the application master

- **SDO Manager**

Optional function to handle dynamic SDO connections, it must reside together with the NMT Master on the CANopen Manager

- **Configuration Manager**

Optional function to configure nodes during boot-up, it must reside together with the NMT Master and SDO Manager on the CANopen Manager

© CiA

The CANopen Application Layer and Communication Profile CiA DS-301 defines the basic communication mechanism for exchanging data. For the description and operation of programmable devices further mechanisms are necessary, which are specified in CiA DSP-302. This specification has to be regarded as a framework for the definition of device profiles for intelligent or programmable devices in form of an extension to the communication profile. The additional mechanisms specified in the framework are useful especially for intelligent devices such as PLCs, HMIs or CANopen tools.

The framework introduces the terms CANopen Manager, which combines NMT Master, SDO Manager, and Configuration Manager functionality. In a CANopen system all these functions must reside on the same device.

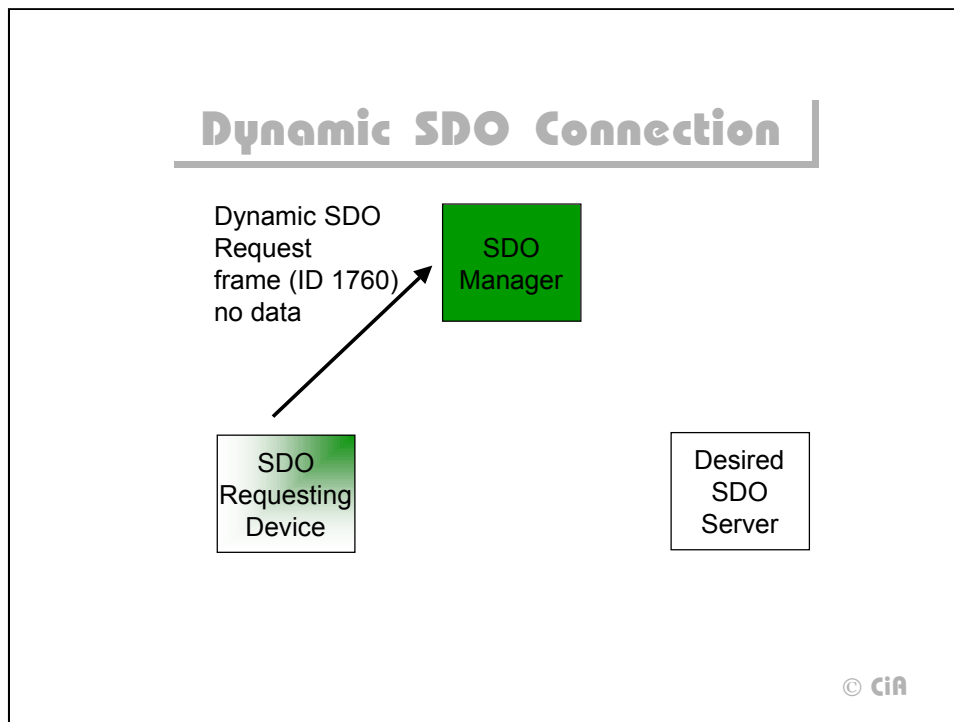
Programmable Devices

- dynamic establishment of SDO connections
- CANopen Manager as network controlling device
- dynamically allocated entries in an object dictionary
- general mechanism for downloading program data
- detecting and configuring of unconfigured nodes during system boot-up
- debugging mechanism in form of an OS command
- multiplexed PDOs

© CiA

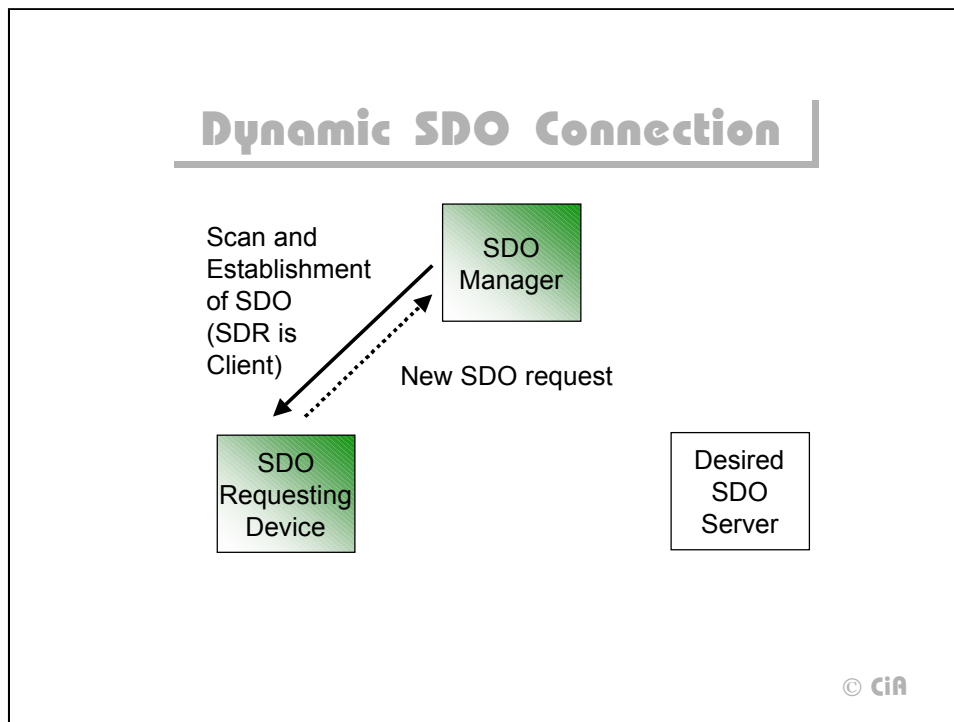
Within a distributed control system the application process is divided into several parts running on different nodes. From the application's point of view usually one node is responsible for system control. This node is called Application Master (e.g. PLC or PC-based controller).

From the network's point of view there are several additional functionality, which not deal with the application but provide application supporting functions. These functions include the above mentioned topics, which are specified in the Framework for Programmable CANopen Devices (CiA DSP-302).



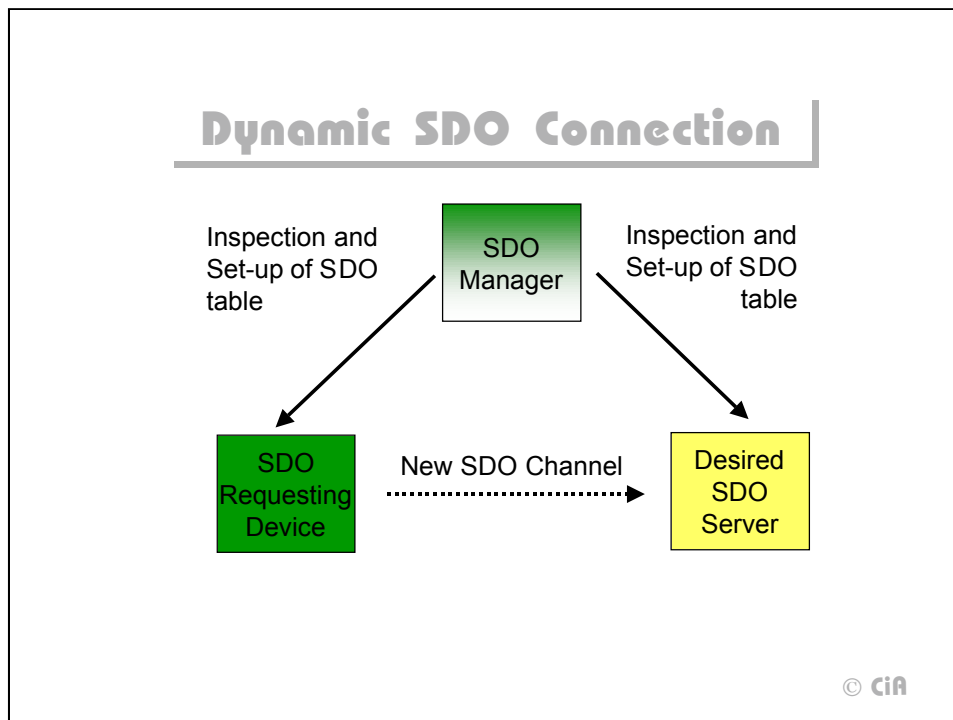
CANopen provides communication mechanism between devices via SDO. Static SDO channels as specified in the CANopen Communication Profile are always established between two nodes. For accessing a device the first time at least one SDO per device is required. This is the default SDO, and only the SDO Manager has the right to access that SDO. Each CANopen device may support additional SDOs, which are disabled by default.

Dynamic establishment of SDO nodes is described in the DSP-302 specification. The SDO Manager manages also dynamic SDOs. To establish SDO channel between a “SDO Requesting Device (SDR)” and a “Desired SDO Server”, the SDR first has to become registered. This is done with the service “Dynamic SDO Request”. This is a CAN data frame with the ID 1760 and no data.

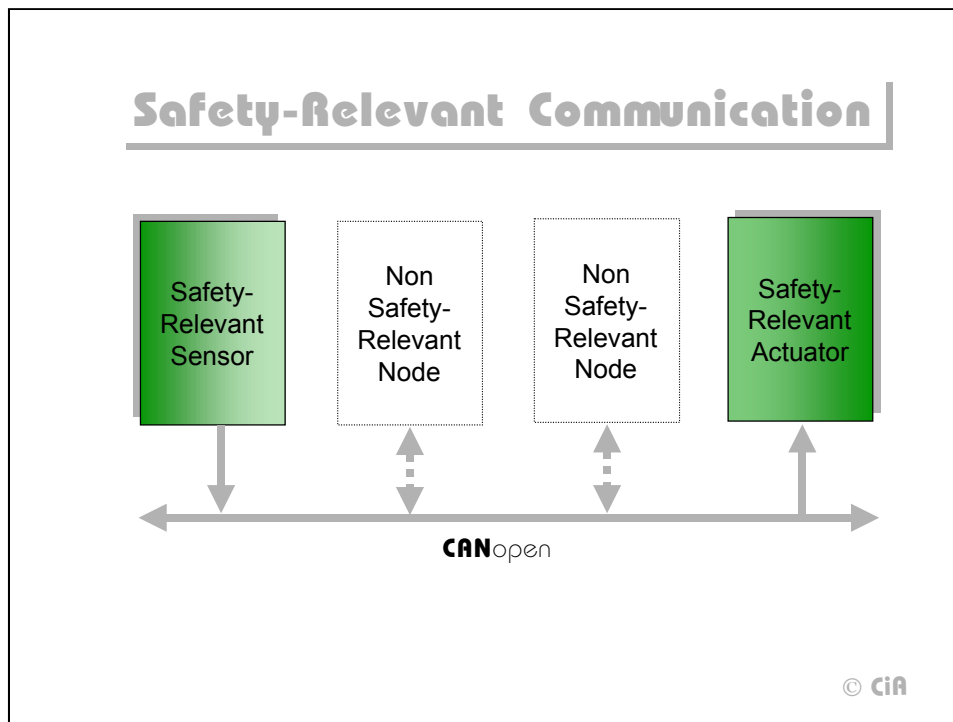


In the next step the SDO manager scans for the requesting device and establishes a connection between SRD as SDO Client and the SDO Manager as SDO Server.

Hereafter the SDR can perform requests to the SDO Manager via the new SDO channel. It will use this to request a connection to the “Desired SDO Server”.



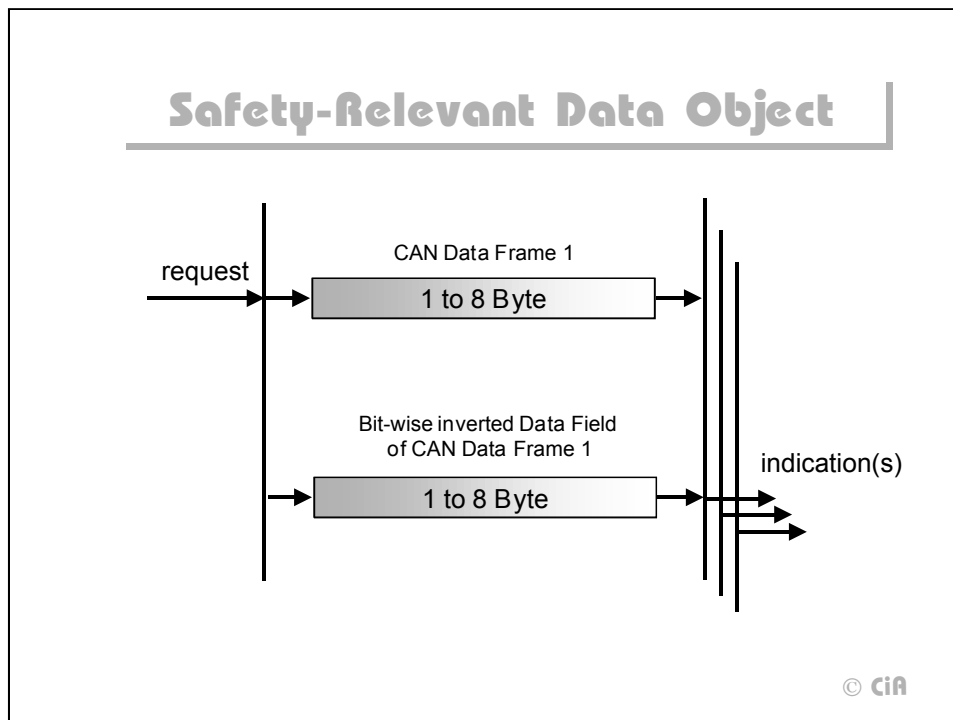
The SDO Manager checks its internal table to determine whether the default SDO of the “Desired SDO Server DSS)” is free. If this is occupied it will check the CANopen Object Dictionary of the DSS for free additional SDOs. It then establishes the connection by writing into the DSS object dictionary. If it decides to use the Default SDO, it does not need to write to the DSS object dictionary, in that case it will up-date only its internal table.



CANopen users require transmission of standard communication objects and transmission of safety-relevant data on the very same physical network. The CANopen communication profile for safety-relevant data transmission is compatible with the CiA DS-301 Version 4.0 CANopen application layer and communication profile. It is intended, that the additional safety-relevant communication is not affecting the standard operation and services on a CANopen network. Safety-relevant communication is not related to a special class of devices, so no specific device profiles have to be used.

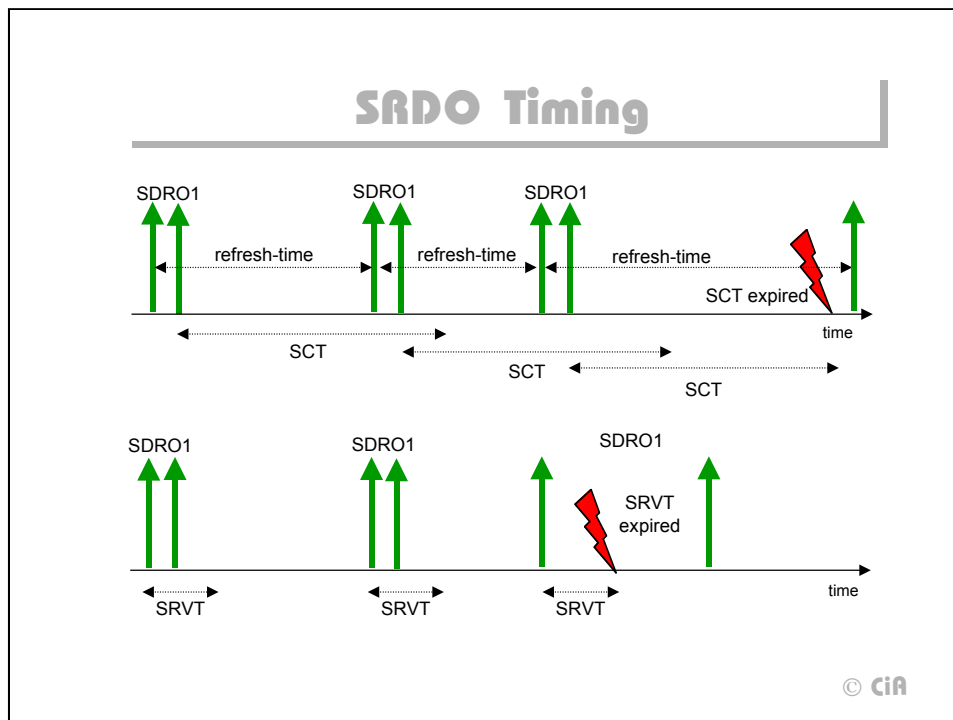
To ensure compatibility, the usage of identifiers and pre-defined objects are coordinated with CiA DS-301. As there is no use of data bits in the safe communication method, it is compatible with existing device profiles.

In CANopen network the data interface to the application program within a certain node is only via the object dictionary accessible. Therefore, the application itself has no influence to the data sequence and the time behavior of the CAN communication. The safety tests due to timing conformance has to be done in the SafetyCAN interface.



Safety relevant data must be distributed by SRDOs (safety relevant data objects). A standard PDO or SDO is not sufficient for hard safety requirements. So with the SRDOs different measures (e.g. redundancy, cyclic transmission etc.) are taken to ensure safety. A identifier range not currently in use for CANopen has been used for the SRDO transmissions.

An SRDO consists of two CAN data frames with different identifiers. The user data in both transmissions is redundant, i.e. the meaning of the data is the same, but the data on the 2nd transmission is inverted bit by bit.



SRDOs must be transmitted periodically in order to test the correct function of the safety components on the CAN bus, the periodic time (SCT) is to be defined. It must be supervised by the safety master.

The period of the SRDOs represents the test scheme of conventional safety circuitry, i.e. is possibly higher than the required reaction time in case of a safety relevant event.

A second test determines, if there is sufficient network capacity for a safety system. Both frames of an SRDO must be received correctly within a given time (SRVT). Normally both frames are transmitted with minimum delay. If the 2nd frame is not being received within SRVT, the bus system has reduced transmission capabilities. The reaction time on a safety relevant event could be enlarged.

Pre-Defined Identifier

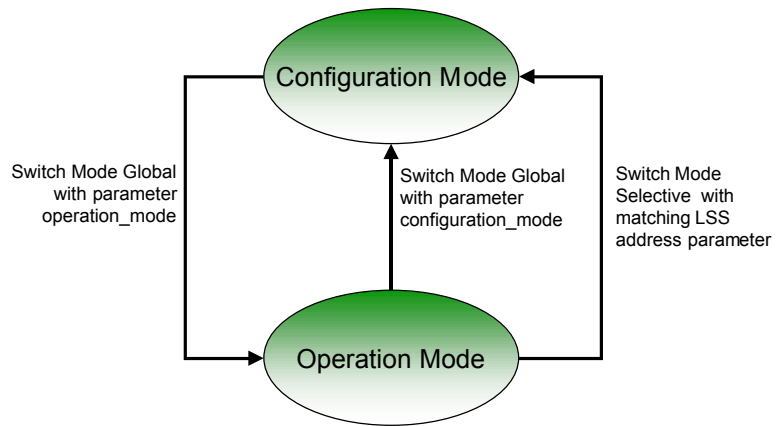
Node ID	SRDO	Object Entry	SRDO Identifiers		Direction
			Data	Complement	
1	1	1301h	257	258	Tx
2	2	1302h	259	260	Tx
...	Tx
32	32	1320h	319	320	Tx
33	33	1321h	321	322	Rx
...
63	63	133Fh	381	382	Rx
64	64	1340h	383	384	Rx

© **cia**

With the predefined connection set the property of a node is linked to the CANopen "Node ID". To enable safety function to a node, it must have access to one SRDO at least - writing or receiving. Assuming, that a safety node is either a producer or a consumer of safety-relevant information, the following matrix of distributing SRDOs together with the corresponding channel directions can be used.

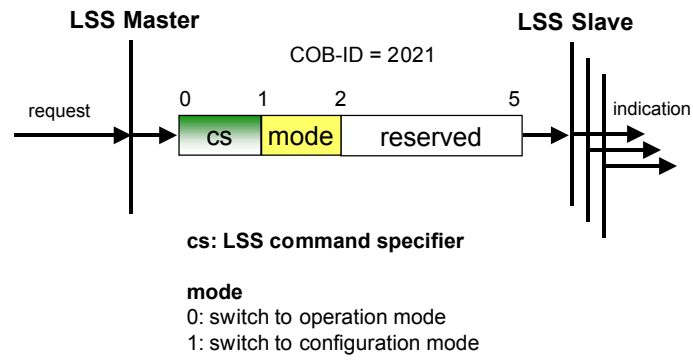
All other communication relationships (one producer - many user, many producer - one user or mixed) are possible, but must be configured accordingly to the application.

Layer Setting Services



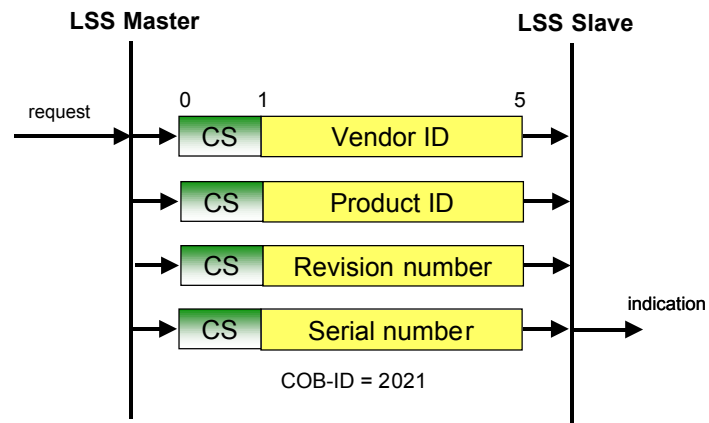
© CIA

Switch Mode Global



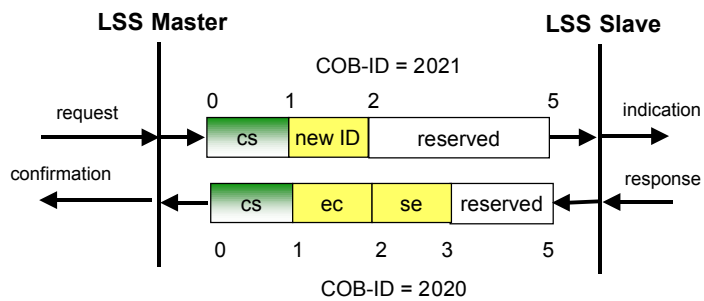
© CIA

Switch Mode Selective



© CIA

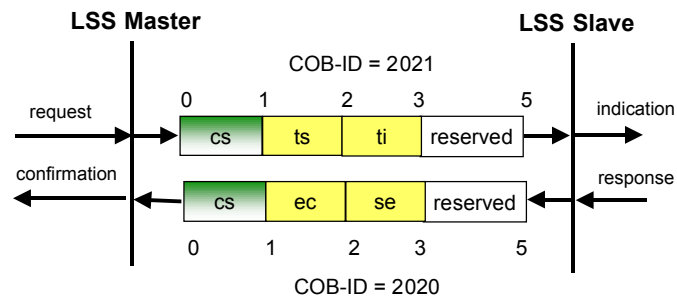
Configure Node ID



new ID: Node ID to be configured
cs: command specifier
ec: error code
se: specific error

© CIA

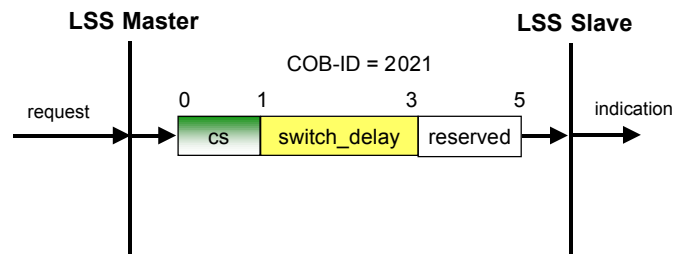
Configure Bit Timing



ts: table selector
ti: table index
ec: error code
se: specific error

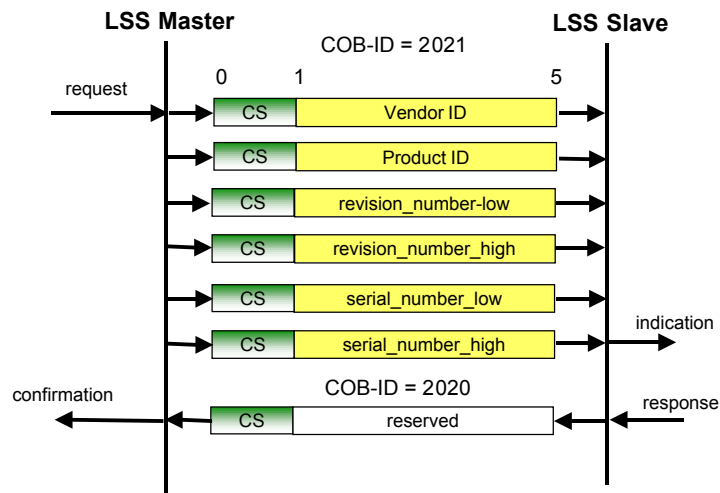
© CIA

Activate Bit Timing



© CIA

Identify Slave



© CIA

Standardized Profiles

Device Profile Specification

- CiA DSP-401: **I/O Modules**
- CiA DSP-402: **Drives and Motion Control**
- CiA DSP-403: **Human Machine Interface**
- CiA WD-404: **Measuring Devices and Closed-Loop Controllers**
- CiA DSP-406: **Encoders**
- CiA WD-408: **Proportional Hydraulic Valves**
- CiA WD-409: **Door Control (Railways)**
- CiA WDP-4XX: **Brake Control (Railways)**
- CiA WDP-4XX: **Train Bus Gateways**

Under development are device profiles for diesel engines, maritime-specific modules and medical-specific systems.

Interface Profile Specification

- CiA DSP-405: **IEC 1131 Programmable Devices**

Application Framework Specification

- CiA WD-407: **Public Transportation**

© CiA

Device Profiles specify supported application objects, additional error codes, and default PDO mappings. There are mandatory objects, optional objects and manufacturer-specific objects. Device Profiles standardized by CiA use the Object Dictionary entries from 6000h to 9FFFh.

CANopen Interface Profiles specify application objects, data type mappings and CANopen object access from other interfaces. The first Interface Profile developed by the CiA is the IEC-1131 Interface Profile (CiA DSP-405).

CANopen Application Profiles describes a specific application including all devices and optionally some additional specifications of the physical layer.

Device Profile Content

- Detailed Specification of Object 1000h
- PDO Transmission Parameters
- PDO Mapping Parameter
- Additional Emergency Codes
- Additional Data Types
- Application Object Descriptions

© CiA

Device Type Object

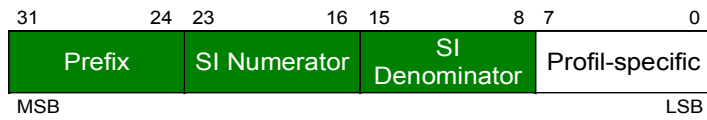
Object 1000h



For multiple device modules the additional information contains FFFFh and the device profile number referenced by this object is the device profile of the first device in the Object Dictionary. All other devices of a multiple device module identify their profiles at objects $67FF + x * 800$ (with x = internal number of the device). These entries describe the device type of the preceding device.

© CiA

SI Unit Object



Example: OBJECT DESCRIPTION

INDEX	6XXXh
Name	Vehicle Velocity
Object Code	Variable
Data Type	Unsigned32
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	Unsigned32
Default Value	03 01 48 00h <10 ³ m/h>
Substitute Value	03 01 48 00h <10 ³ m/h>

© CiA

The CiA DRP-303-2 CANopen recommendation describes the representation of SI units and prefixes.

I/O Module Profile

Object 1000h: Device Type

MSB		LSB
Additional Information		Device Profile Number
1st Bit	digital input	401d
2nd Bit	digital output	
3rd Bit	analog input	
4th Bit	analog output	
Rest	reserved	

© CiA

The object at index 1000h describes the type of device and its functionality. It is composed of a 16-bit field, which describes the device profile that is used and a second 16-bit field, which gives additional information about optional functionality of the device. The Additional Information parameter is specified in the appropriate device profile.

The CANopen Device Profile for I/O Modules defines remote I/O devices and programmable I/O modules. They can receive configuration information via the SDO.

Default I/O Mapping

1st T_PDO	8-bit in	8-bit in	8-bit in	8-bit in	8-bit in	8-bit in	8-bit in	8-bit in
	8 x 8-bit digital inputs transmitted asynchronously							
1st R_PDO	8-bit out	8-bit out	8-bit out	8-bit out	8-bit out	8-bit out	8-bit out	8-bit out
	8 x 8-bit digital outputs received asynchronously							
2nd T_PDO	16-bit input		16-bit input		16-bit input		16-bit input	
	4 x 16-bit analog inputs transmitted asynchronously							
2nd R_PDO	16-bit output		16-bit output		16-bit output		16-bit output	
	4 x 16-bit analog outputs received asynchronously							

© CiA

If a module supports a specific type of I/O it must support the related default PDOs. It is open to a manufacturer to specify additional PDO mappings and it is also open to a user to change these default settings by changing the mapping structure, if the module supports variable mapping on these PDOs.

The default PDO parameters are specified in the objects 1400h, 1800h, 1401h resp. 1801h. The transmission for all PDOs is by default asynchronous, and the inhibit time is 0. The default mappings are specified in the objects 1600h, 1A00h, 1601 resp. 1A01h.

Optional I/O Access

Digital Inputs

- single-bit access (6020h to 6027h)
- 2-byte access (6100h)
- 4-byte access (6120h)

Analog Inputs

- 1-byte access (6400h)
- 4-byte access (6402h)
- manufacturer-specific (6404h)

Digital Outputs

- single-bit access (6220h to 6227h)
- 2-byte access (6300h)
- 4-byte access (6320h)

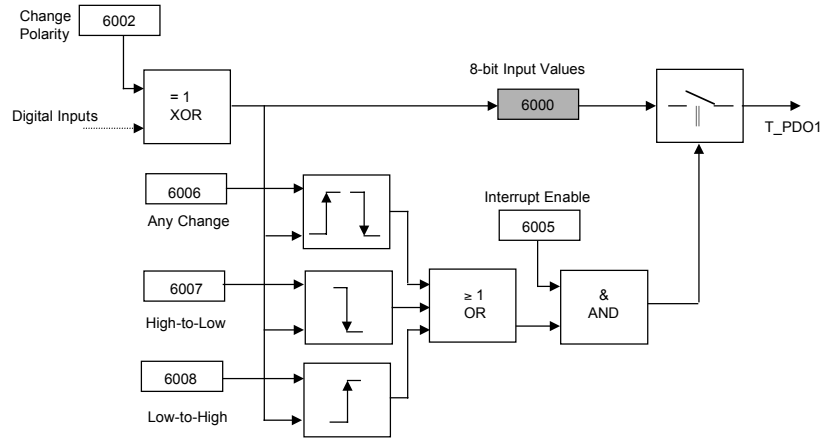
Analog Outputs

- 1-byte access (6410h)
- 4-byte access (6412h)
- manufacturer-specific (6414h)

© CiA

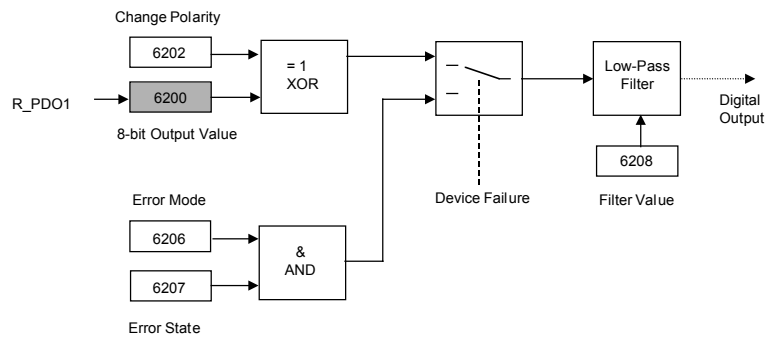
Besides the I/O access specified for the default mapping, the CANopen I/O module profile supports optionally different other access methods. These optional access methods require variable PDO mapping or these application objects can only be transmitted by SDO communication.

Digital Input Objects



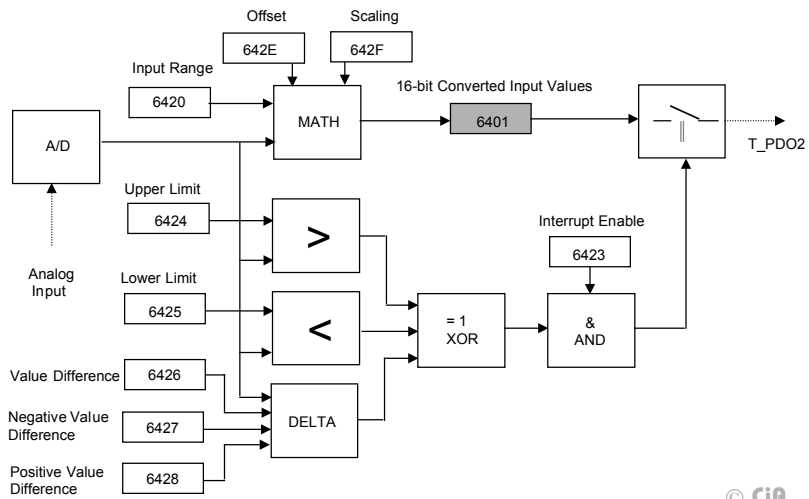
© CiA

Digital Output Objects



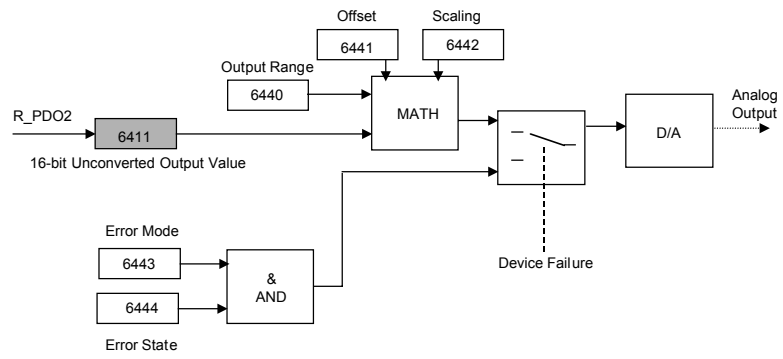
© CiA

Analog Input Objects



© CiA

Analog Output Objects



© CiA

Drive and Motion Control Profile

Object 1000h: Device Type

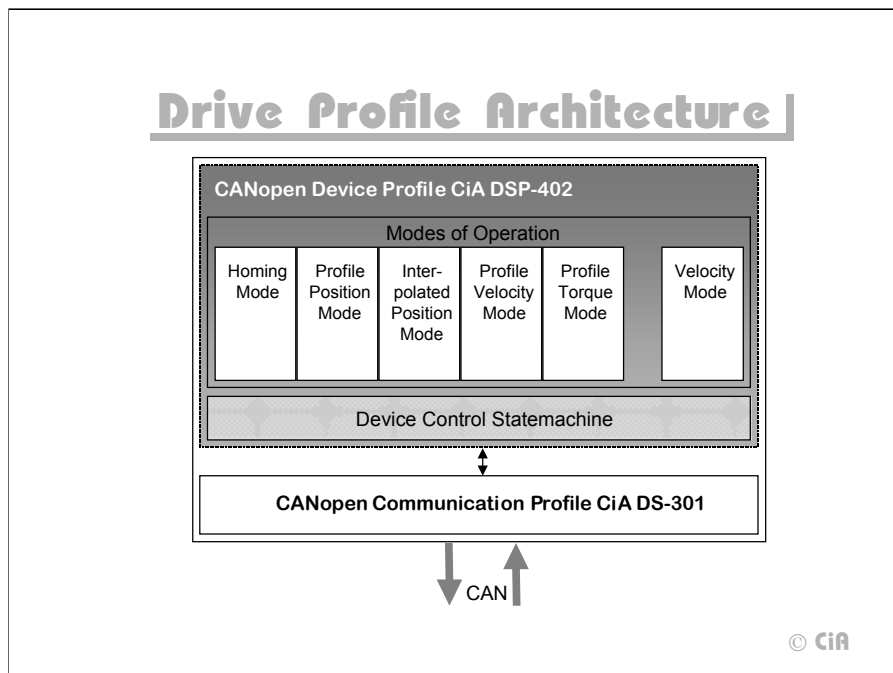
MSB		LSB	
Additional Information		Device Profile Number: 402d	
mode bits	type		
31	24 23	16 15	0

mode bits (24 to 31):
00h
(manufacturer-specific)

drive type (16 to 23):
01h (frequency converter)
02h (servo drive)
03h (stepper motor)
80h (I/O module)
FFh (multi device module)

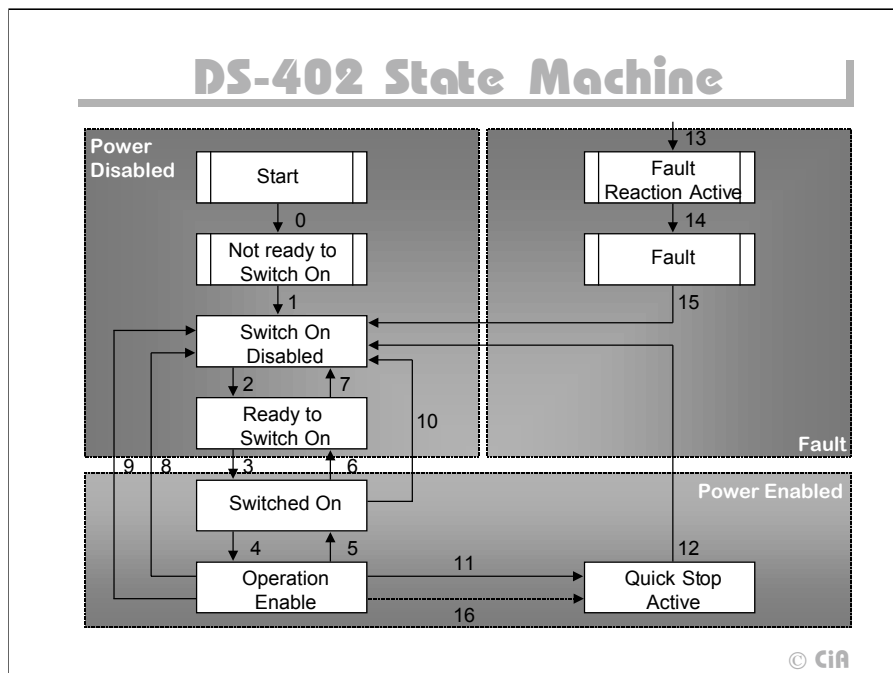
© CiA

The purpose of this profile is to give drives an understandable and unique behavior on the CANopen network. The purpose of drive units is to connect axle controllers or other motion controllers to the CAN bus. They can receive configuration information what is done via SDO services normally for I/O configuration, limit parameter for scaling or application-specific parameter. At runtime, data can be obtained from the drive unit via CAN bus by either polling or event driven (interrupt).



The starting and stopping of the drive and several mode specific command are executed by the state machine, The operation mode defines the behavior of the drive:

- Homing Mode describes the various methods to find a home position, reference point, datum, or zero point.
- Profile Position Mode defines the positioning of the drive. Speed, position and acceleration can be limited and profiled moves using a Trajectory Generator are also possible.
- Interpolated Position Mode describes the time interpolation of single axles and the spatial interpolation of coordinated axles. Synchronization mechanisms and interpolation data buffers are covered as well.
- Profile Velocity Mode is used to control velocity of the drive with no special regard of the position. It supplies limit functions and trajectory generation.
- Profile Torque Mode defines the torque control with all related parameter.
- Velocity Mode is a simple mode used by many frequency converters. It provides limit and ramp functions.



The state machine describes the device status and possible control sequence of the drive, A single state represents a special internal or external behavior. The state of the drive also determines, which commands are accepted. E.g. it is only possible to start a point-to-point move when the drive is in state Operation Enabled.

States may be changed using the Controlword and/or according to internal events. The current state can be read using the Statusword.

Default Receive PDOs

PDO	Mapping Object Index	Mapping Object Name	M/O	Comment
1	6040h	controlword	M	controls the state machine
2	6040h	controlword	O	controls the state machine and mode of operation
	6060h	modes_of_operation		
3	6040h	controlword	O	controls the state machine and the target position
	607Ah	target_position		
4	6040h	controlword	O	controls the state machine and the target velocity
	6081h	profile_velocity		
5	6040h	controlword	O	controls the state machine and the target torque
	6071h	target_torque		
6	6040h	controlword	O	controls the state machine and the nominal speed
	6042h	vl-target_velocity		
7	6040h	controlword	O	controls the state machine and the digital outputs
	60FEh	digital_outputs		
8	6040h	controlword	O	controls the state machine and modes of operations (broadcast PDO)
	6060h	modes_of_operation		

© CiA

A drive supporting more than one mode will mostly use more than one Default PDO. Therefore a lot of PDOs are predefined in respect to different possible modes of operation for drives.

The described PDO distribution should be used for every axle of a multi-device module with an offset of 64, e.g. the first PDO of the second axle gets the number 65. In this way a system with a maximum of 8 axles is supported.

It is open to the manufacturer to specify additional entries in the mapping table or define absolutely new PDO mappings and it is also open to change these default settings by changing the mapping structure, if the device supports variable PDO Mapping.

The Controlword is a 2-byte object (Unsigned16). The other mapped objects are of different length.

Default Transmit PDOs

PDO	Mapping Object Index	Mapping Object Name	M/O	Comment
1	6041h	statusword	M	shows status
2	6041h	statusword	O	shows status and actual
	6061h	modes_of-operation_display		mode of operation
3	6041h	statusword	O	shows status and
	6064h	position_actual_value		the actual position
4	6041h	statusword	O	shows status and
	606Ch	velocity_actual_value		the actual velocity
5	6041h	statusword	O	shows status and
	6077h	torque_actual_value		the actual torque
6	6041h	statusword	O	shows status and
	6044h	v1_control_effort		the actual speed
7	6041h	statusword	O	shows status and
	60FDh	digital_inputs		the digital inputs

© CiA

The Statusword is a 2-byte object (Unsigned16). The other mapped objects are of different length.

Multi-Axles Devices |

6000h to 67FF	axle 0
6800h to 6FFF	axle 1
7000h to 77FF	axle 2
7800h to 7FFF	axle 3
8000h to 87FF	axle 4
8800h to 8FFF	axle 5
9000h to 97FF	axle 6
9800h to 9FFF	axle 7

Remark: Optional I/O functionality must conform to CiA DSP-401 and can be implemented instead of an axle.

© CiA

The Drives and Motion Control profile object area is divided in eight ranges to realize 8 axles. For standard drives only the range 6000h to 67FFh is mandatory.

Additional it is possible to describe optional I/O functions combined with the drive. These I/O objects must conform to the I/O Module profile and can be implemented instead of an axle.

Device Profile for HMI

Object 1000h: Device Type

MSB		LSB
Additional Information		Device Profile Number
1st Bit	simple HMI	403d
2nd Bit	intelligent HMI	
3rd Bit	very intelligent HMI	
Rest	reserved	

© CiA

The CANopen Device Profile for HMI covers devices from simple text displays with small keyboards, displays with graphical capability, and displays with touch-screen functionality.

Simple HMI do not run application programs and may only support read key code, write lamp code, text monitor, and text cursor objects.

Intelligent HMI provides an additional pre-configured SDO Client. So this device can initiate up-/downloads from other nodes.

Very intelligent HMI provides DSP-302 functionality, in particular, dynamic SDO connections. It can be a "SDO Requesting Device (SRD).

Default HMI Mapping

T_PDO_1	Read Key Code								
R_PDO_1	Write Lamp Code								
T_PDO_2	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%; text-align: center;">NDRT</td> <td style="width: 25%; text-align: center;">Index</td> <td style="width: 25%; text-align: center;">S-Index</td> <td style="width: 25%;"></td> </tr> </table> NDRT = New Data Ready Transmit	NDRT	Index	S-Index					
NDRT	Index	S-Index							
R_PDO_2	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%; text-align: center;">NDRR</td> <td style="width: 25%; text-align: center;">Index</td> <td style="width: 25%; text-align: center;">S-Index</td> <td style="width: 25%;"></td> </tr> </table> NDRR = New Data Ready Receive	NDRR	Index	S-Index					
NDRR	Index	S-Index							
T_PDO_3	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%; text-align: center;">Key 1-8</td> <td style="width: 12.5%; text-align: center;">Key 9-16</td> <td style="width: 12.5%; text-align: center;">Key 17-24</td> <td style="width: 12.5%; text-align: center;">Key 25-32</td> <td style="width: 12.5%; text-align: center;">Key 33-40</td> <td style="width: 12.5%; text-align: center;">Key 41-48</td> <td style="width: 12.5%; text-align: center;">Key 49-56</td> <td style="width: 12.5%; text-align: center;">Key 57-64</td> </tr> </table>	Key 1-8	Key 9-16	Key 17-24	Key 25-32	Key 33-40	Key 41-48	Key 49-56	Key 57-64
Key 1-8	Key 9-16	Key 17-24	Key 25-32	Key 33-40	Key 41-48	Key 49-56	Key 57-64		
R_PDO_3	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 12.5%; text-align: center;">Lamp 1-8</td> <td style="width: 12.5%; text-align: center;">Lamp 9-16</td> <td style="width: 12.5%; text-align: center;">Lamp 17-24</td> <td style="width: 12.5%; text-align: center;">Lamp 25-32</td> <td style="width: 12.5%; text-align: center;">Lamp 33-40</td> <td style="width: 12.5%; text-align: center;">Lamp 41-48</td> <td style="width: 12.5%; text-align: center;">Lamp 49-56</td> <td style="width: 12.5%; text-align: center;">Lamp 57-64</td> </tr> </table>	Lamp 1-8	Lamp 9-16	Lamp 17-24	Lamp 25-32	Lamp 33-40	Lamp 41-48	Lamp 49-56	Lamp 57-64
Lamp 1-8	Lamp 9-16	Lamp 17-24	Lamp 25-32	Lamp 33-40	Lamp 41-48	Lamp 49-56	Lamp 57-64		

© CiA

The first Transmit-PDO contains the 2-byte Read Key Code object. The key status is 1 for pressed key and 0 for released key. The key code relates to the pressed key.

The first Receive-PDO contains the Write Lamp Code object. This 2-byte object defines the lamp attributes (blinking, color, etc.)

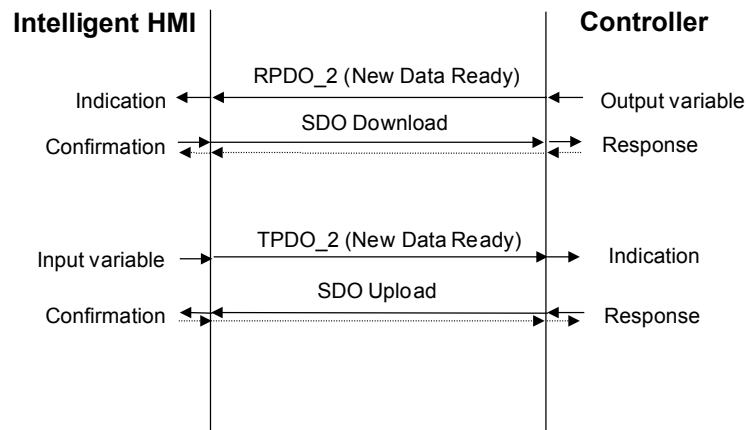
The second Transmit-PDO signalize that a variable value has changed. The index and sub-index relate to the changed variable.

The second Receive-PDO indicates that a variable value in a client has changed. The index and sub-index relate to the changed variable.

The third Transmit-PDO transmits by default the first 8 x 8 key values. A maximum of 255 x 8 key values is addressable (2040 keys).

The third Receive-PDO writes by default 8 x 8 lamp values. A maximum of 255 x 8 lamps is addressable (2040 lamps).

Principle of Data Exchange

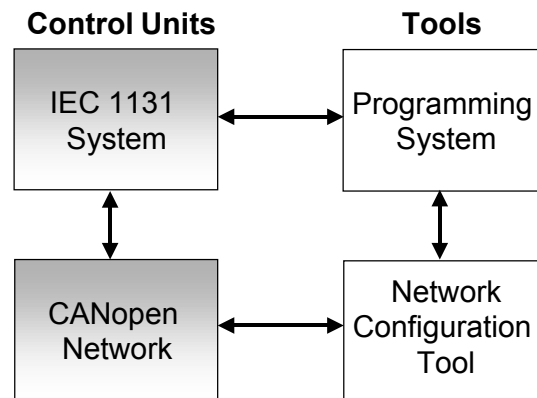


© CiA

A simple HMI device is based on a simple I/O module model. These devices only must support Read Code object (6000h), Write Lamp object (6200h), Text Monitor object (6210)h, and Text Cursor object (6211). More skilled simple HMI devices may support additionally Display Stored Data object (6212h) instead of the 6210h object. Variables are displayed by using SDO transfers to Write Output Variable objects. If an user is able to enter a variable, the HMI device signalizes with the 2nd Transmit-PDO that there are new values on Read Input Variable objects.

Intelligent HMI devices implement pre-configured Client-SDOs. So they can down- and up-load data from other devices. Acting as Client-SDO, the HMI device can read and write to the Object Dictionary of other nodes including the application master.

Profile for IEC 1131 Interfaces



© CiA

The CiA DSP-405 CANopen Interface Profile for IEC 1131 Programmable Devices is based on communication services specified in the CANopen Communication Profile and the Framework for Programmable CANopen Devices. This profile covers the access to a CANopen communication system from within an IEC 1131 program based on variables or on calls to function blocks as well as utility functions for debugging, monitoring and network management.

With respect to integrating tools for CANopen configuration and IEC-1131 programming and debugging, this profile defines two different kinds of integration:

- The network-centric approach, in which CANopen configuration is assumed to be done after CANopen configuration, being logically below configuration.
- The PLC-centric approach, in which CANopen configuration is assumed to be done after IEC-1131 programming, being logically only one part of the configuration of the complex PLC system.

Generating an application implements the handling of four interfaces.

Variable-Based Access

Input Variables		Output Variables	
A000h	Integer8	A480h	Integer8
A040h	Unsigned8	A4C0h	Unsigned8
A080h	Boolean	A500h	Boolean
A0C0h	Integer16	A540h	Integer16
A100h	Unsigned16	A580h	Unsigned16
A140h	Integer24	A5C0h	Integer24
A180h	Unsigned24	A600h	Unsigned24
A1C0h	Integer32	A640h	Integer32
A200h	Unsigned32	A680h	Unsigned32
A240h	Float (32)	A6C0h	Float (32)
A280h	Unsigned40	A700h	Unsigned40
A2C0h	Integer40	A740h	Integer40
A300h	Unsigned48	A780h	Unsigned48
A340h	Integer48	A7C0h	Integer48
A380h	Unsigned56	A800h	Unsigned56
A3C0h	Integer56	A840h	Integer56
A400h	Integer64	A880h	Integer64
A440h	Unsigned64	A8C0h	Unsigned64

© CiA

The Framework for Programmable CANopen Devices defines the usage of so-called segments. It leaves the concrete placement of the segments open. To ease implementations and for a much easier usage of software from different manufacturers, the usage of the segments is specified in the IEC-1131 Interface Profile: The segments are placed in the index range A000h to AFFFh. This allows any device to use the standard object dictionary entries of a specific device profile and additionally use of the Network Variables.

Object dictionary entries, which have names that are no legal variable names for IEC-1131-3 shall not be usable to the IEC-1131 system. No automatic renaming is defined in the profile.

Function Block Based Access

No.	Function Block
1	CIA405_RECV_EMY_DEV
2	CIA405_RECV_EMY
3	CIA405_SDO_WRITE4
4	CIA405_SDO_WRITE7
5	CIA405_SDO_WRITE14
6	CIA405_SDO_WRITE21
7	CIA405_SDO_READ4
8	CIA405_SDO_READ7
9	CIA405_SDO_READ14
10	CIA405_SDO_READ21

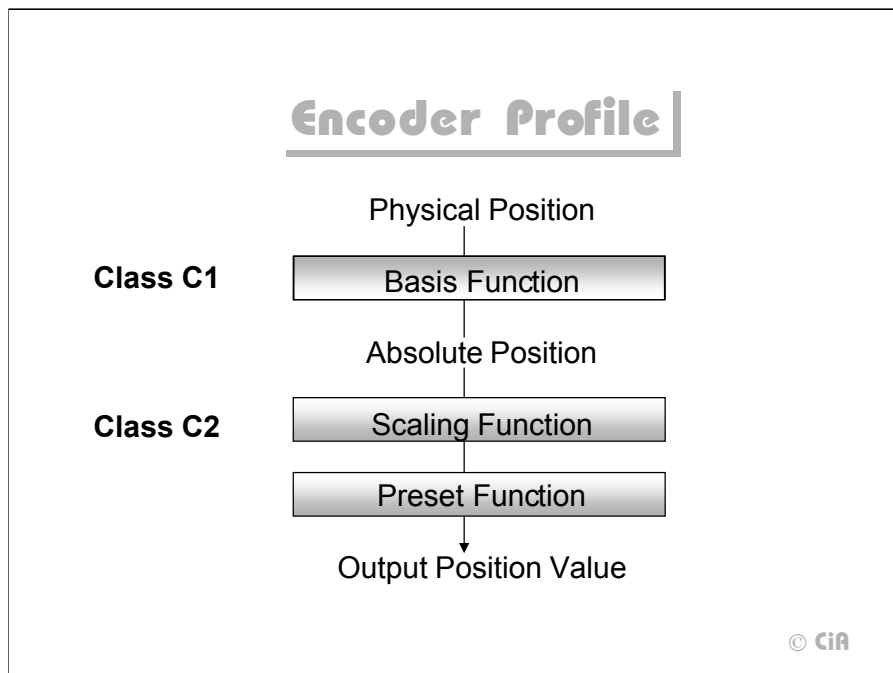
© CiA

The function block based access to CANopen communication services requires some naming conventions. A specific prefix shall be given to all function block and data type names defined in the IEC-1131 Interface Profile. Currently CIA405 is used as that prefix. The table can be used by IEC-1131 compliant systems to state the features covered.

The interface to function blocks is based on the assumption of passing information only on the length of the data being transmitted, not on the specific type of data. IEC-1131-3 does provide any data types, which could be used to define generic function blocks, but implementation of such function blocks would be much more effort than implementation of function blocks based only on the length of the data being transmitted.

It is possible to represent transmission of data of arbitrary length within IEC-1131-3. However, the interfaces to function blocks allowing for that would be quite more difficult to use and understand. Therefore different function blocks are defined in the interface profile, for the transmission of a fixed (maximum) amount of data each.

The interface profile is designed such that it is possible to wrap the calls with a timer block to implement time-outs. Additionally, it is allowed that lower level CANopen software implements their own time-outs and report such as errors to the caller.



The CANopen encoder profile CiA DSP-406 describes the functionality of CAN-connected encoders of two device classes:

- Class C1 is the mandatory class with a basic range of functions that all encoders must support (transformation of the physical position into an absolute position).
- Class C2 encoders support all class C1 functions and extended functions defined in class 2 (scaling function and preset function).

In addition to the two classes, there are pre-defined areas and reserved parameters for manufacturer-specific functions in the encoder device profile area.

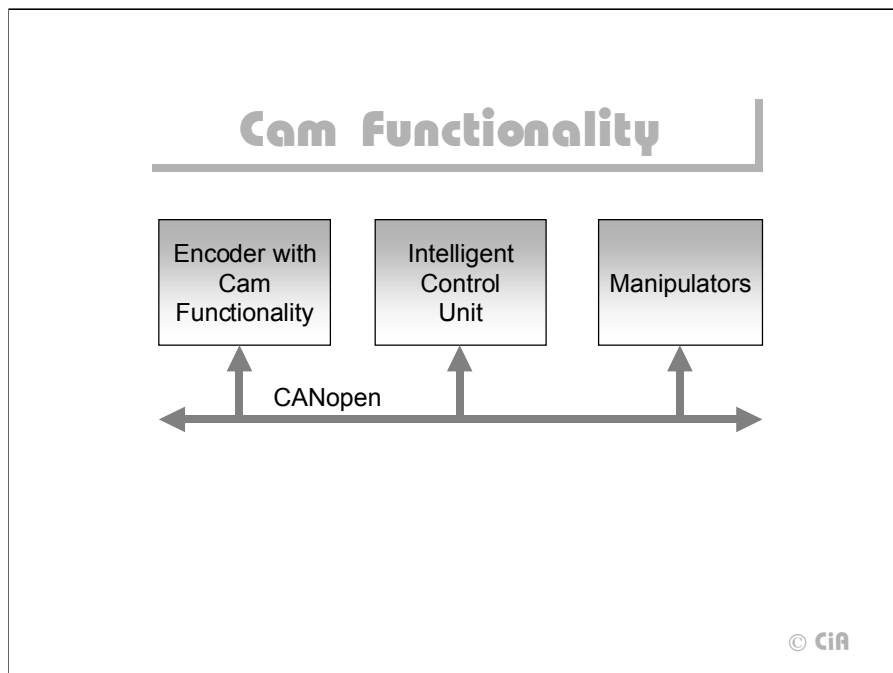
Encoder Profile (2)

Object 1000h: Device Type

MSB		LSB
	Additional Information	Device Profile Number
01	single-turn absolute rotary encoder	406d
02	multi-turn absolute rotary encoder	
03	single-turn absolute rotary encoder with electronic turn-count	
04	incremental rotary encoder	
05	incremental rotary encoder with electronic counting	
06	incremental linear encoder	
07	incremental linear encoder with electronic counting	
08	absolute linear encoder	
09	absolute linear encoder with cyclic coding	
10	multi-sensor encoder interface	
11 to 65535	reserved	

© CiA

The CiA DSP-406 CANopen device profile includes incremental and absolute linear and rotary encoders. Besides position and velocity output possibility complete cam functionality is covered. In addition it is possible to handle multi sensors through one CANopen Encoder.



Different operation steps in a machine tool are triggered by cams which synchronize the processes involved. Conventional systems use stand-alone electrical cam switch mechanisms. The signals from this mechanisms have to be routed to control units and then supplied to the manipulators such as cylinder drives, electrical conveyors etc. New concepts use encoders with CANopen bus interface and integrated cam functionality. So no additional electrical cam switching mechanism is necessary any more.

Optional up to 254 cam positions with a maximum of 8 cam's each channel can be supported by encoder devices compliant with CiA DSP-406 (Version 2.0). Each cam has parameters for minimum switch point, maximum switch point and setting a hysteresis to the switch points.

Encoder Default Mapping

1st Transmit_PDO (asynchronous transmission)

Byte 0	Byte 1	Byte 2	Byte 3
--------	--------	--------	--------

Output position value

2nd Transmit_PDO (synchronous transmission)

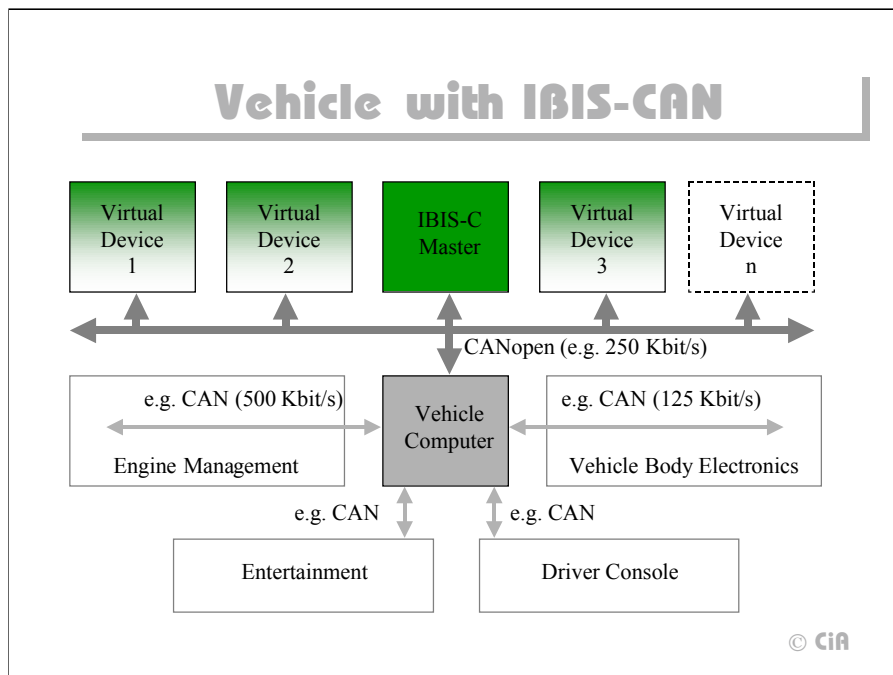
Byte 0	Byte 1	Byte 2	Byte 3
--------	--------	--------	--------

Output position value

© CiA

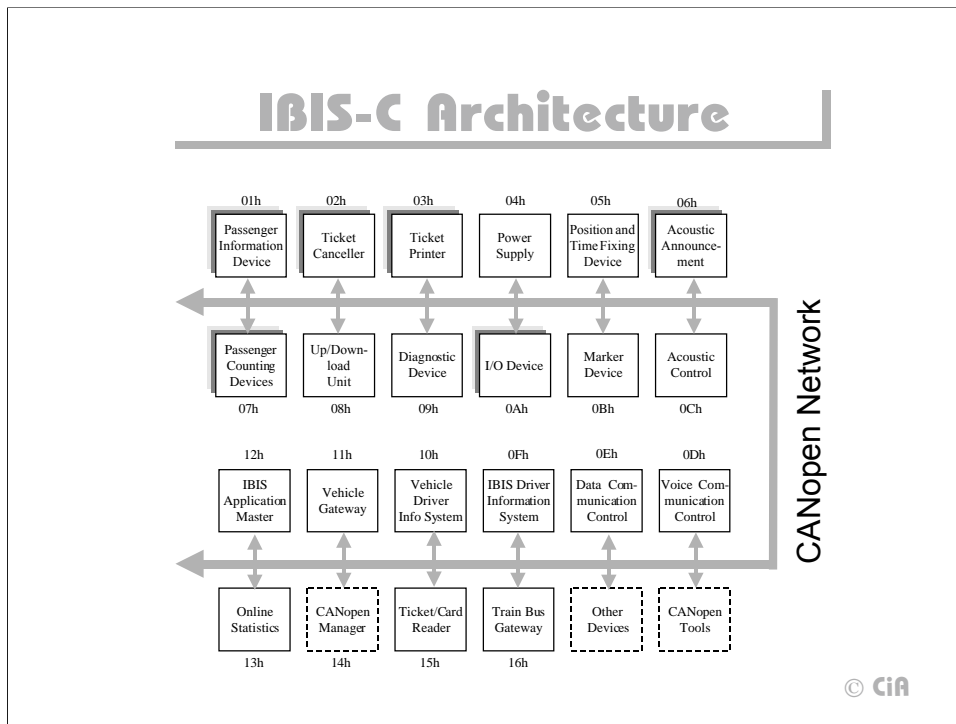
CiA DSP 406 defines two standard Process Data Objects which have to be implemented in the encoders: one for asynchronous transmission and the other one for the cyclic transmission functions. The position value of an encoder can be scanned in three different operating modes:

- Polled mode: The host polls the actual position value via a Remote Transmit Request.
- Cyclic mode: The encoder cyclically outputs the position value without being prompted by the host. The system integrator is responsible for selecting a suitable cycle time in accordance with the access priorities assigned to other devices connected to the bus. Values between 1 ms and 65535 ms can be selected.
- Sync mode: On receiving a Sync message the encoder outputs its position value. Synchronization within the network is guaranteed with the help of the CANopen Sync message. The Sync counter is responsible for additional weighting, i.e. in accordance with the number of Sync messages required to transmit a position value.



In omnibuses CAN networks are heavily in use. To connect ticket canceling machine, passenger information display, passenger counting units, etc. the German association of public transportation is developing in cooperation with CiA the IBIS-CAN specification. This is a CANopen Application profile.

The electronic heart of such an omnibus is the vehicle computer, which could support up to five CAN networks. In Germany several of these vehicle computers are IEC-1131 compliant control systems.



The CANopen Application Profile for Public Transportation specifies virtual devices, which resides in CANopen physical devices. One physical device may have several virtual devices. One virtual device can't resides in several physical devices.

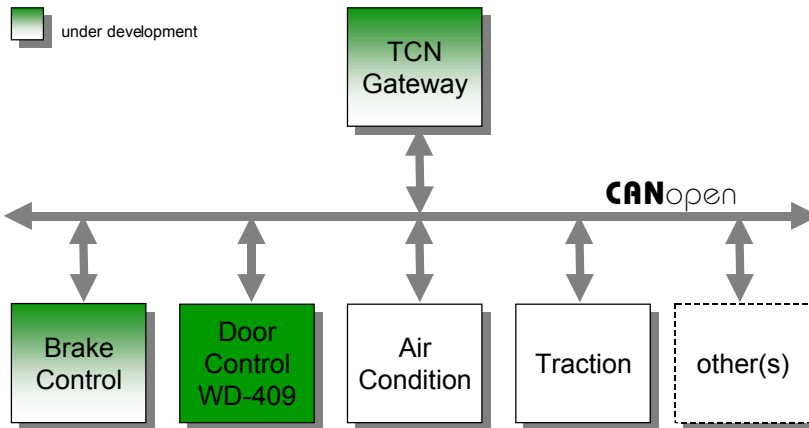
Many of the application objects are derived from the IBIS specification. IBIS-CAN application profile is the successor and is based on CANopen. The profile specifies additional virtual devices and related application objects.

Public Transport Physical Layer

- CAN Transceiver Chip according to ISO 11898
- Bit-timing according to CiA DS-301 CANopen Specification
- Default Baudrate is 250 Kbit/s (optional other baudrates)
- Line Topology with Drunk and Drop Lines
- max. Drop Line Length of 6 m at 250 Kbit/s
- max. accumulated Drop Line Length of 30 m at 250 Kbit/s
- Termination Resistors on both Ends (nominal 120 Ohm)
- Special connectors including pin assignment

© CiA

CANopen Railway Profiles



© CiA

Door Control Profile

T_PDO1

11	12	13	14	15	16	17	18	9	A	B	C	D	E	F	10	1	2	3	4	5	6	7	8
----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	----	---	---	---	---	---	---	---	---

6010h 1h door opened
2h door closed
3h door enabled
4h door locked
5h automatic close enabled
6h emergency disabled
7h stop request activated
8h door re-opened
9h door forced closed
Ah door opposite side enabled
Bh step opened
Ch step closed

6010h Dh step enabled
Eh step locked
Fh ramp opened
10h ramp closed
11h ramp enabled
12h ramp locked
13h not used
14h not used
15h not used
16h not used
17h not used
18h not used

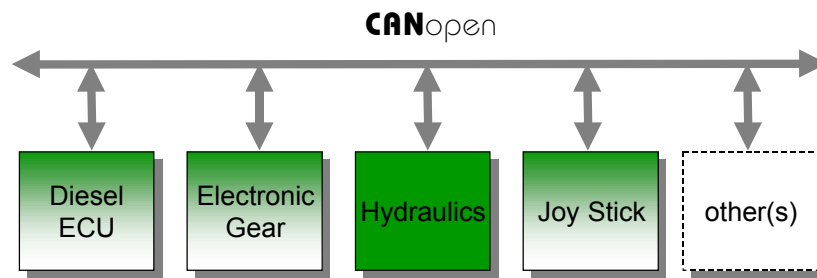
Door Control Profile


R_PDO1


11	12	13	14	15	16	17	18	9	A	B	C	D	E	F	10	1	2	3	4	5	6	7	8
----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	----	---	---	---	---	---	---	---	---

6010h	1h	open door	6010h	Dh	enable step
	2h	close door		Eh	lock step
	3h	enable door		Fh	open ramp
	4h	lock door		10h	close ramp
	5h	enable automatic close		11h	enable ramp
	6h	disable emergency		12h	lock ramp
	7h	activate stop request		13h	not used
	8h	reopen door		14h	not used
	9h	close door forced		15h	not used
	Ah	enable door opposite side		16h	not used
	Bh	open step		17h	not used
	Ch	close step		18h	not used

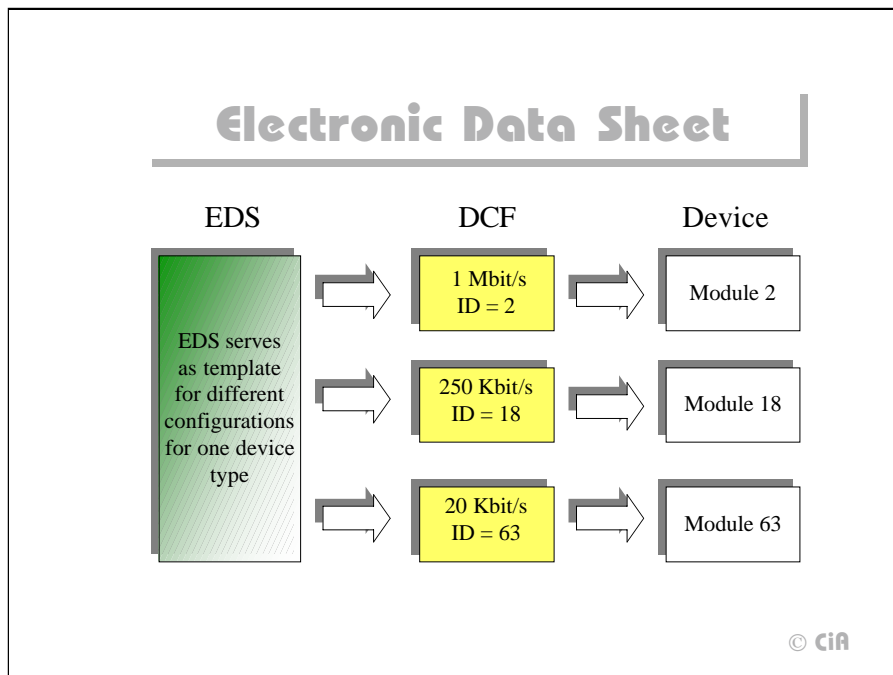
CANopen Vehicle Profiles



 CiA work draft

 CiA work draft proposal

© CiA



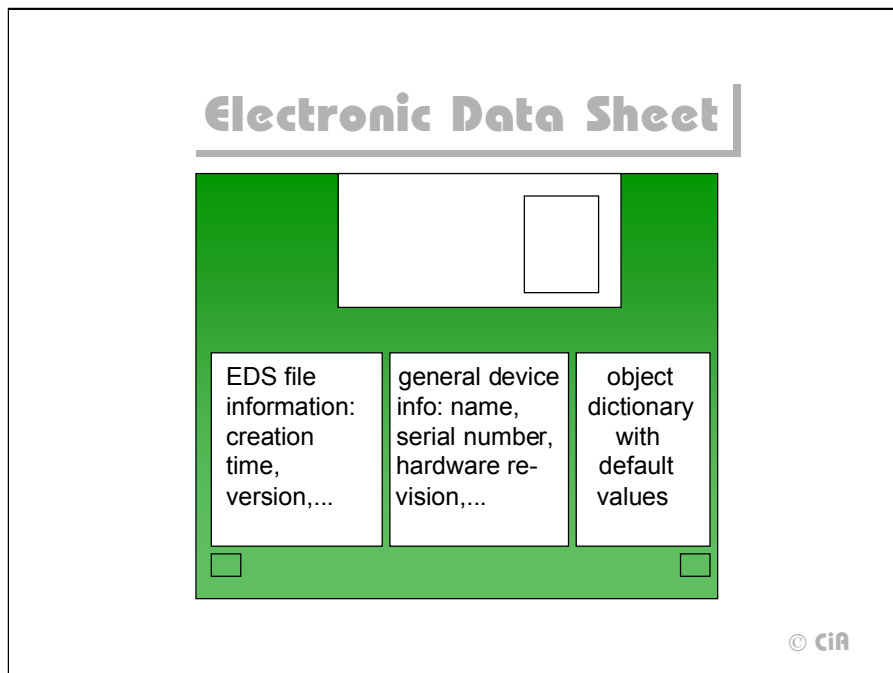
In order to give the user of a CANopen device more support the device's description should be available in a standardized way. The Electronic Data Sheet gives the opportunity to create standardized tools for:

- configuration of CANopen devices,
- designing networks with CANopen devices,
- managing project information on different platforms.

Therefore two types of files are introduced to define CANopen device with electronically means. The files are ASCII-coded and it is recommended to use the ANSI character set.

An EDS (Electronic Data Sheet) can be used to describe the communication functionality and objects as defined in the CANopen specifications. The EDS is the template for a device of a vendor. The DCF (Device Description File) describes the incarnation of a device not only with the objects but also with the configured values of objects. Furthermore a value for the baud-rate of a device and for the Module-ID are added.

Also a part of the CANopen conformance test bases on a comparison between the device under test and the EDS.



An EDS should be supplied by the vendor of a particular device. If a vendor has no EDS available for a CANopen device a default EDS can be used. The default EDS comprises all entries of a device profile for a particular device class. There are EDS generator tools available on the market.

An EDS can be divided into three parts:

- information regarding the EDS file,
- general device information,
- object dictionary with default values.

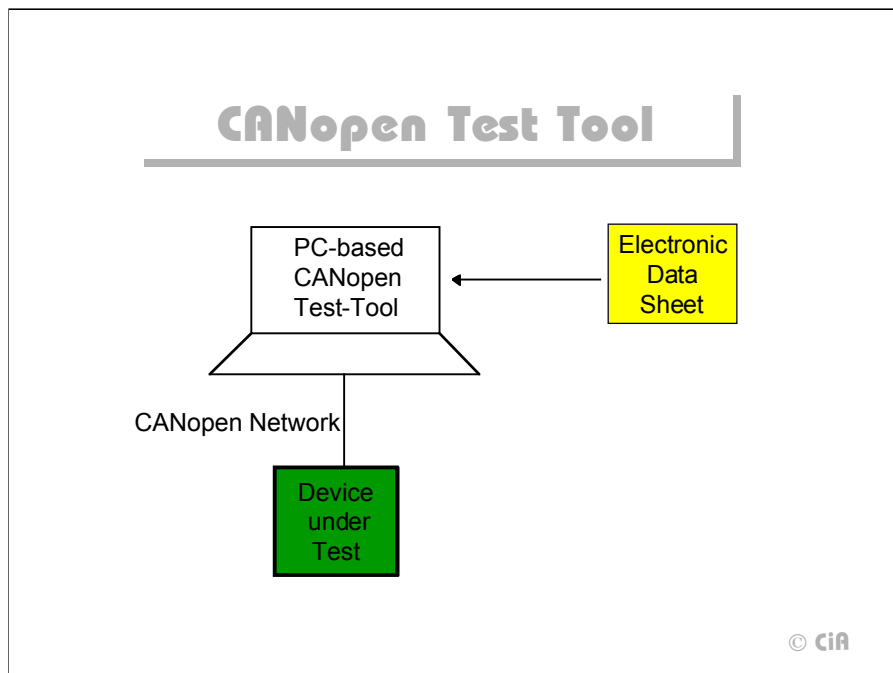
Conformance Testing

Test description for CANopen Devices:

- framework for conformance testing
- static test without timing requirements
- test with respect to the communication profile
- the device under test is tested alone without any application, interoperability is not tested
- a test system will be available at the CiA, so that every vendor and buyer of CANopen devices can check conformance to the standards
- availability of the test tool for self test

© CiA

The CANopen conformance testing specification developed by CAN in Automation has now been implemented and CiA offers the service of an official test laboratory where CANopen devices can be certified.



The devices are tested with respect to the CANopen Communication Profile DS-301 Version 3.0, not to a special device profile. The test specification includes a static test where timing requirements are not taken into consideration. For every test a test report will be generated listing all steps of the test and all errors that have occurred during the test.

In a first step the EDS (Electronic Data Sheet) of the CANopen device is tested. By means of an EDS a CANopen device can be described with respect to the contents of its object dictionary. The EDS is the base for all the existing CANopen configuration tools on the market. The following requirements shall be full filled by the contents of the EDS: correct value ranges, support of mandatory entries, references pointing to existing entries and consistency of the EDS.

In a second step the physical CANopen device is tested. This part includes the test of the Communication Protocol, the test of the EDS against the object dictionary and the verification of network states and transitions.

CANopen Certification at CiA

Basic Rate

per Test Session: 260.- Euro
130.- Euro for CiA Members

Rate per hour: 80.- Euro

Certificate: 100.- Euro
50.- Euro for CiA Members

The certificate describes the hardware of the device (CAN controller, microcontroller) and versions of the CANopen implementation and the EDS file.

© CiA

The certification will be done by the standard CANopen conformance testing tool on basis of a PC with a CAN interface module. This software tool has a standardized COTI interface and runs on different hardware platforms. The test software is now available by CiA and National Instruments and may be offered by other providers of CAN PC hardware as well.