CANopen

- ◆ Modeling
- ◆ Physical Layer
- ◆ Communication Objects
- ◆ Network Management
- ◆ Identifier Distribution
- ◆ Layer Setting Specification
- ◆ Framework for Programmable Devices
- ◆ Electronic Data Sheet (EDS)
- ◆ Device, Interface and Application Profiles
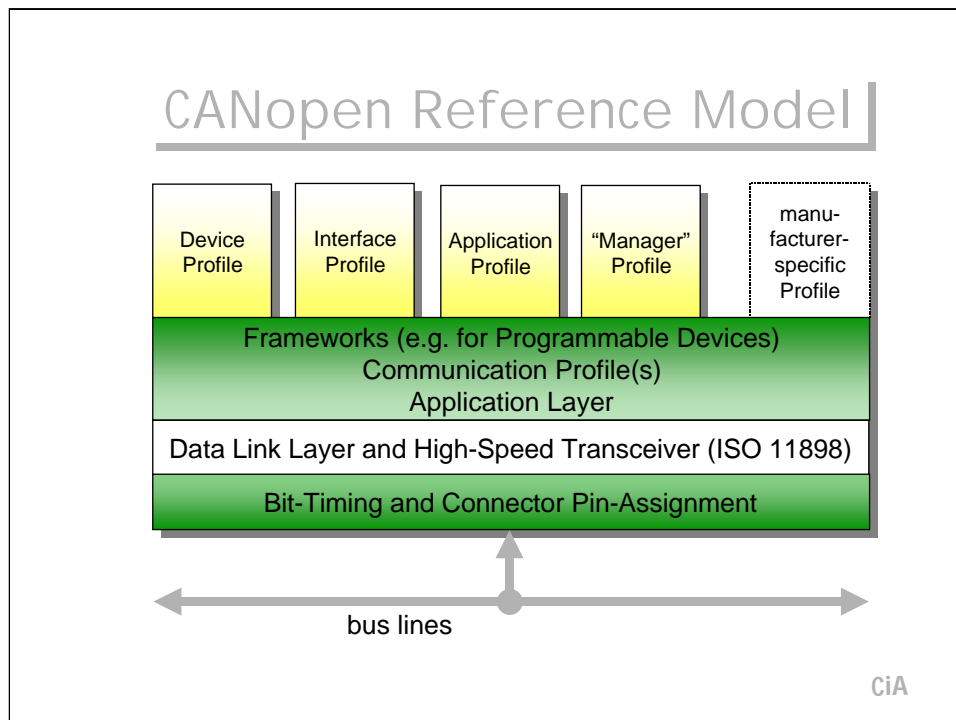- ◆ CANopen Conformance Test

CiA

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.
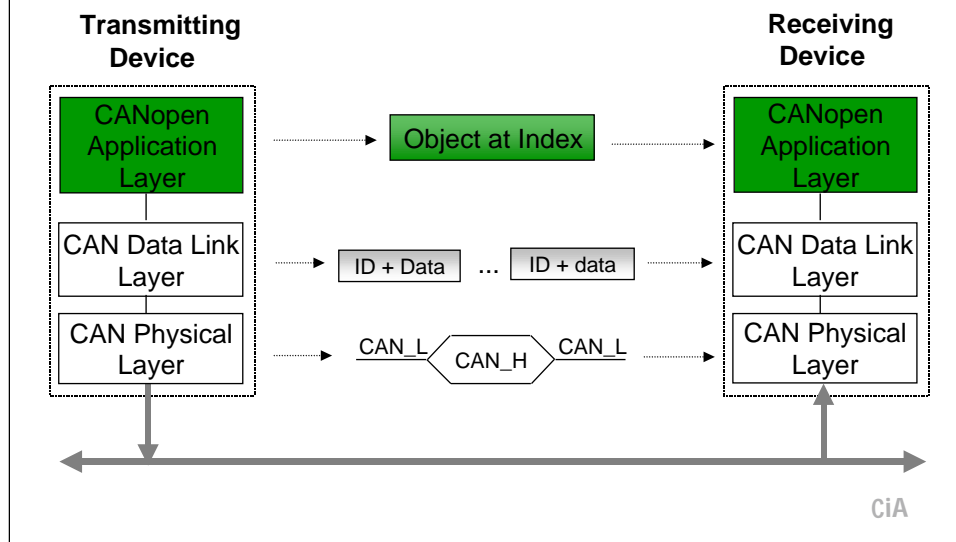
CANopen Reference Model

Device Profile	Interface Profile	Application Profile	"Manager" Profile	manufacturer-specific Profile

Frameworks (e.g. for Programmable Devices)
Communication Profile(s)
Application Layer

Data Link Layer and High-Speed Transceiver (ISO 11898)

Bit-Timing and Connector Pin-Assignment

bus lines

CiA

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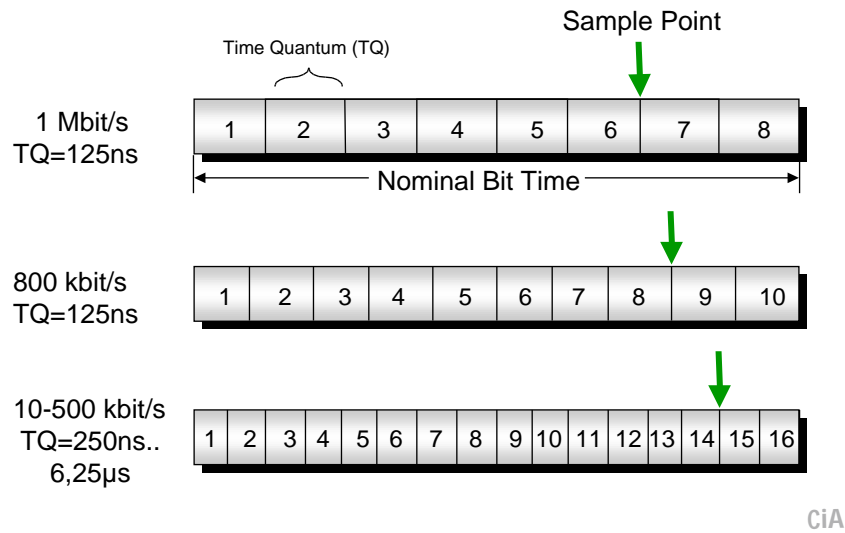
The standardized device profiles, interface profiles, and application profiles describes the default behavior and the optional functionality of devices, interfaces, and applications.

Protocol Layer Interactions

Transmitting Device		Receiving Device
CANopen Application Layer	Object at Index	CANopen Application Layer
CAN Data Link Layer	ID + Data ... ID + data	CAN Data Link Layer
CAN Physical Layer	CAN_L CAN_H CAN_L	CAN Physical Layer

CiA

The protocol layer interactions describe the communication on the different layers. On the CANopen application layer the devices exchanges 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

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)

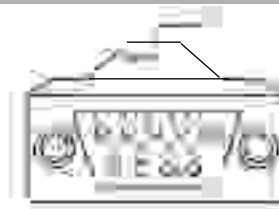| Bit rate Bus length [1]	Nominal bit time t_b	Number of time quanta per bit	Length of time quantum t_q	Location of sample point	Index
1 Mbit/s 25 m	1 µs	8	125 ns	6 t_q (750 ns)	0
800 kbit/s 50 m	1.25 µs	10	125 ns	8 t_q (1 µs)	1
500 kbit/s 100 m	2 µs	16	125 ns	14 t_q (1.75 µs)	2
250 kbit/s 250 m [2]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

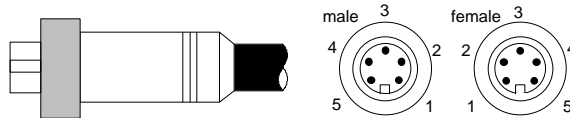| Pin	Signal	Description
1	-	Reserved
2	CAN_L	CAN_L bus line dominant low
3	CAN_GND	CAN Ground
4	-	Reserved
5	(CAN_SHLD)	Optional CAN Shield
6	GND	Optional Ground
7	CAN_H	CAN_H bus line dominant high
8	-	Reserved
9	(CAN_V+)	Optional CAN external positive supply

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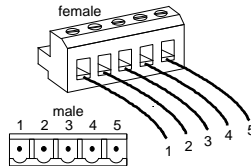
5-pin Mini Style Connector : ANSI/B93.55M-1981

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

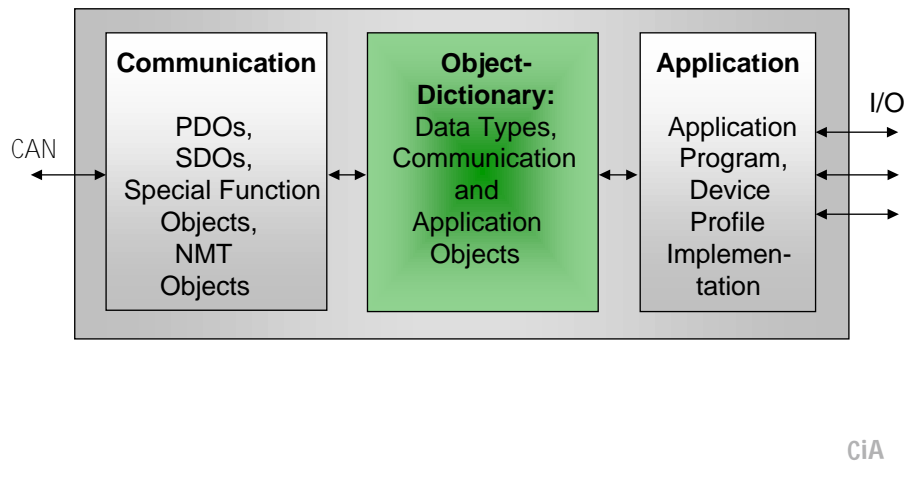| Pin	Signal	Description
1	(CAN_SHLD)	Optional CAN Shield
2	(CAN_V+)	Optional CAN external positive supply (dedicated for supply of transceiver and optocouplers, if galvanic isolation of the bus node applies)
3	CAN_GND	Ground / 0V / V-
4	CAN_H	CAN_H bus line (dominant high)
5	CAN_L	CAN_L bus line (dominant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

CANopen Device Model

Communication	Object-Dictionary:	Application
PDOs, SDOs, Special Function Objects, NMT Objects	Data Types, Communication and Application Objects	Application Program, Device Profile Implementation

CAN

I/O

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 INTEGERn has values in the integers. The value range is $-2^{n-1}, ..., 2^{n-1}-1$ (bit sequences of length n). Data of basic type UNSIGNEDn has values in the non-negative integers. The value range is $0, ..., 2^{n-1}-1$ (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: Unsigmed8 - 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 DIFFERENCS 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)

* not forRecords

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

* only for sub-objects in Arrays and Records
** only for sub-objects in Records

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

```c
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 Objects

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 compatiblity 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 compatiblity reasons			
100C	VAR	guard time	Unsigned32	rw	O
100D	VAR	life time factor	Unsigned32	rw	O
100E		Reserved for compatiblity reasons			
100F		Reserved for compatiblity 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

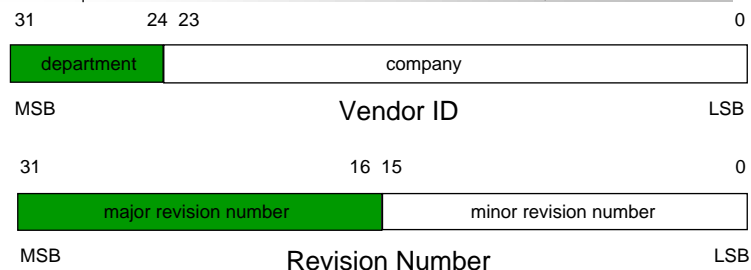| 0	1	2	3	4	5	6	7
Generic Error	Curre nt	Voltage	Tempe-rature	Commu-nication Err or	Device pr ofile-specific	Reserv-ed (0)	Manu -facturer -specific
M	O	O	O	O	O	O	O

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.

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

31   24   23   0

department	company

MSB   Vendor ID   LSB

31   16   15   0

major revision number	minor revision number

MSB   Revision Number   LSB

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.
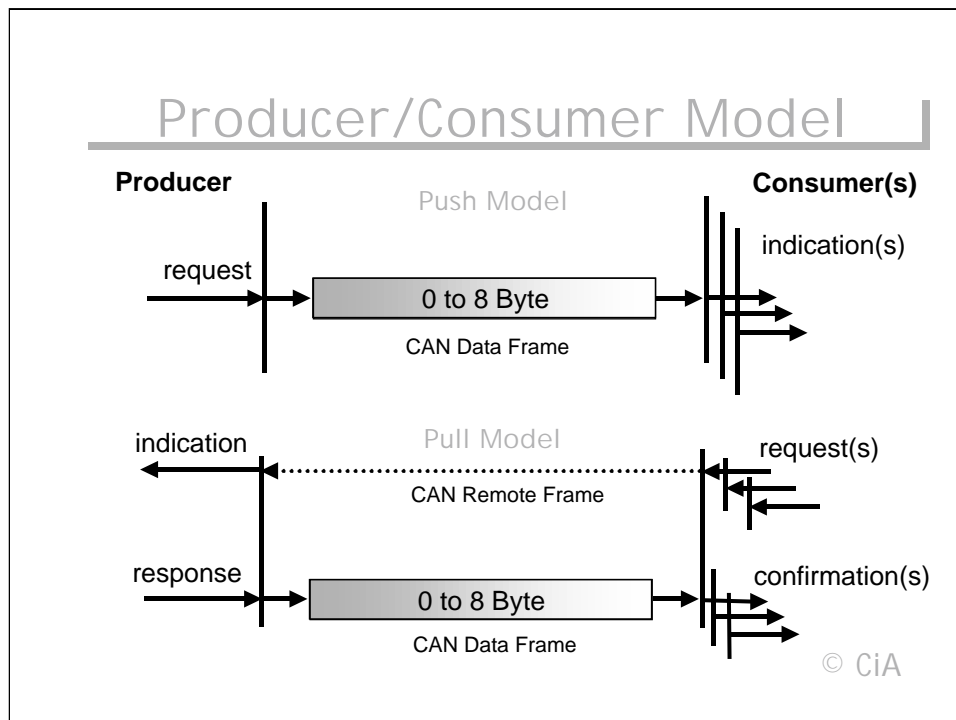
Communication Protocols

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

CiA

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

• The real-time data transfer is performed by the Process Data Objects (PDOs) protocol.

• With Service Data Objects (SDOs) protocols the read and write access to entries of a device object dictionary is provided.

• Special Function Object protocols provide application-specific network synchronization, time stamping and emergency message transmissions.

• The Network Management (NMT) protocols provide services for network initialization, error control and device status control.

Producer/Consumer Model

Producer

Push Model

request

0 to 8 Byte

CAN Data Frame

Consumer(s)

indication(s)

Pull Model

indication

CAN Remote Frame

request(s)

response

0 to 8 Byte
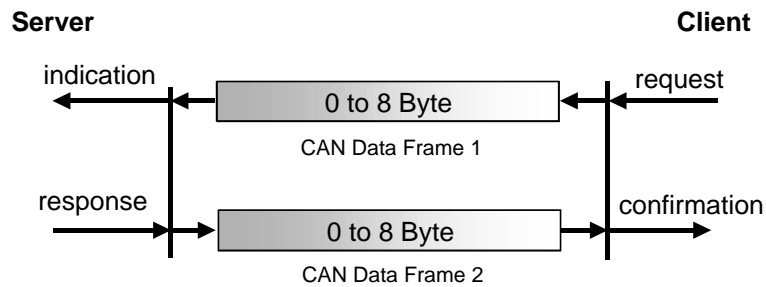
CAN Data Frame

confirmation(s)

© CiA

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).
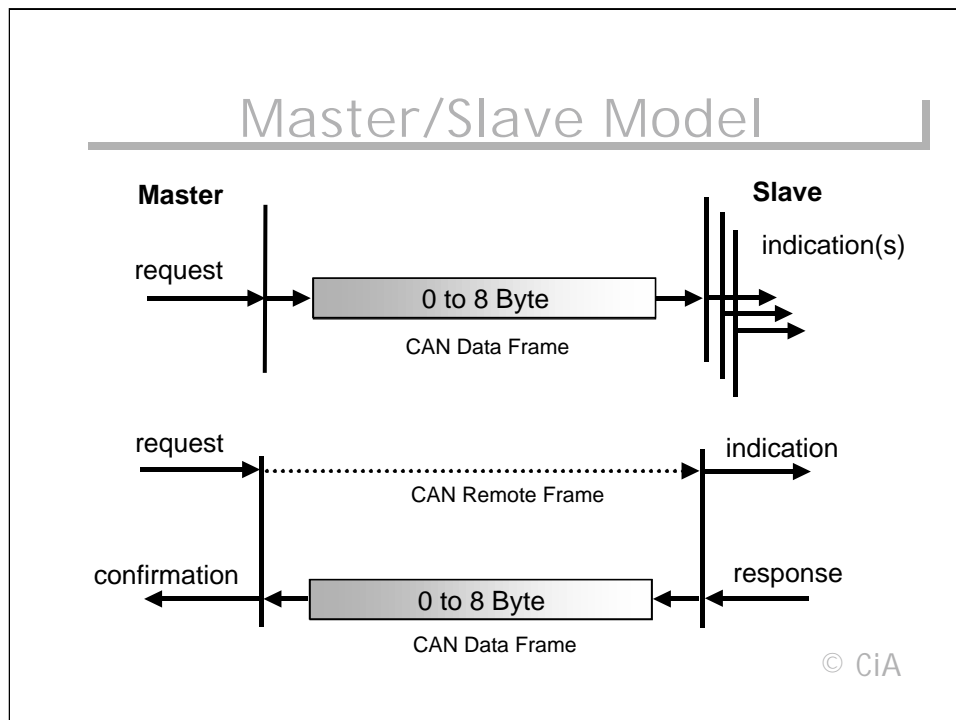
Client/Server Model

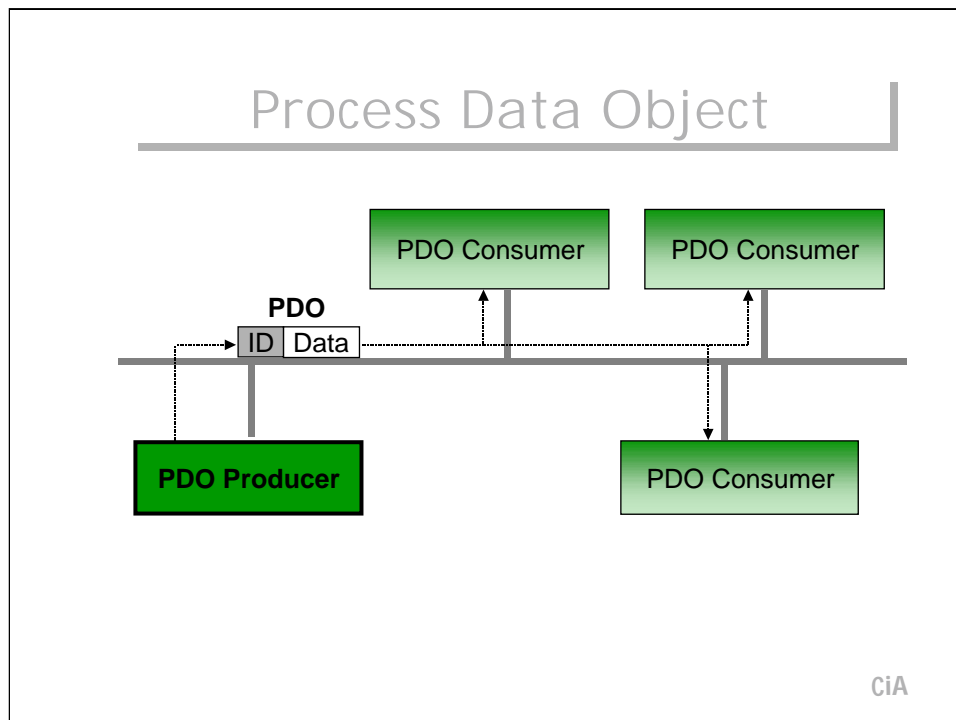| Server		Client
indication ←	0 to 8 Byte ←	← request
	CAN Data Frame 1	
response →	→ 0 to 8 Byte	confirmation →
	CAN Data Frame 2	

© CiA

In the classical Client/Server model the client transmits a messages, which will be responded by the server, so that the client get 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 byte. 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.

Master/Slave Model

Master   **Slave**

request   indication(s)

| 0 to 8 Byte |

CAN Data Frame

request   indication

. . . .
CAN Remote Frame

confirmation   response

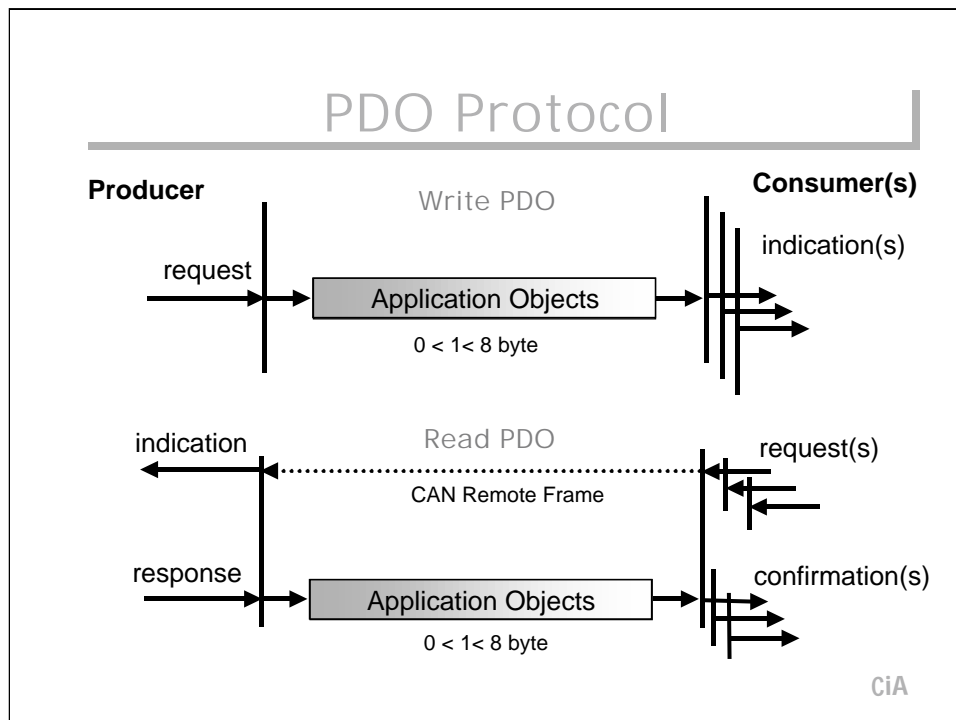| 0 to 8 Byte |

CAN Data Frame

© CiA

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.

Process Data Object

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_PDO) with a specific identifier, which corresponds to the identifier of the Receive-PDO (R_PDO) of one or more consumers.

PDO Protocol

Producer   **Write PDO**   Consumer(s)

request   indication(s)

Application Objects

0 < 1< 8 byte

indication   **Read PDO**   request(s)

CAN Remote Frame

response   confirmation(s)

Application Objects

0 < 1< 8 byte

CiA

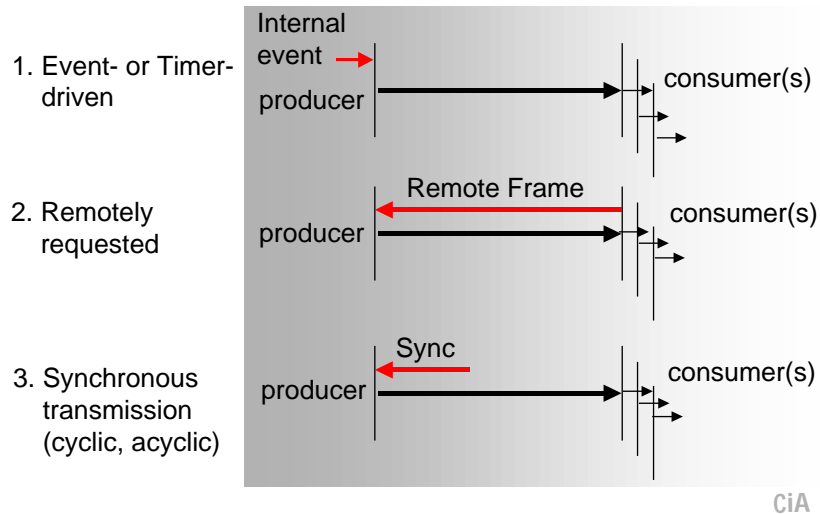There are two PDO services: to write a PDO and to read a PDO.

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.
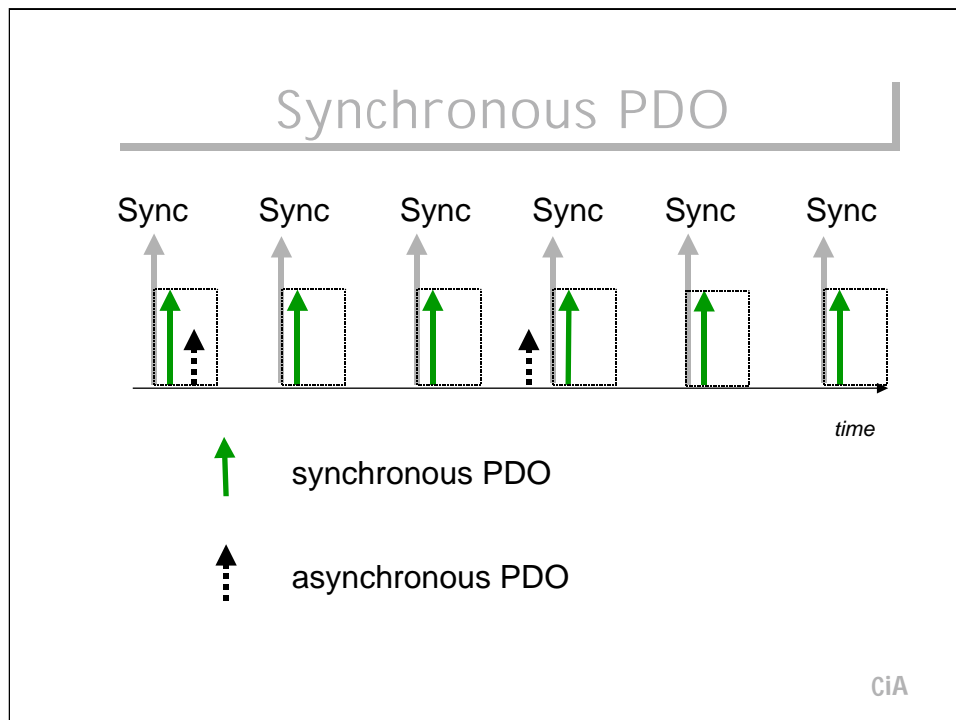
PDO Scheduling Modes

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.
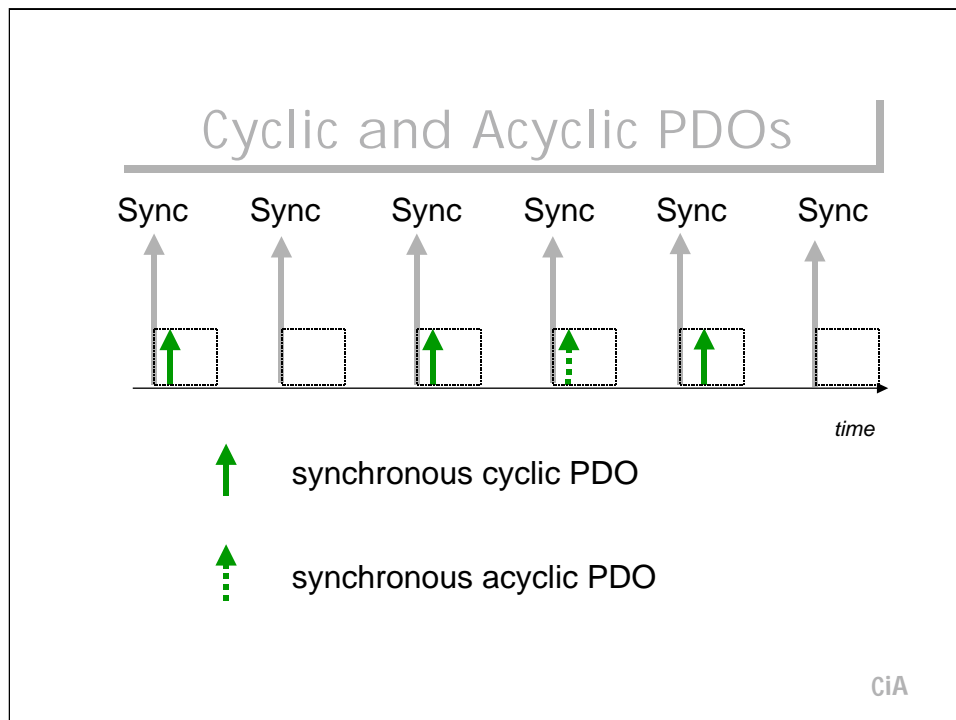
Synchronous PDO

Sync Sync Sync Sync Sync Sync

time

↑ synchronous PDO

↑ asynchronous PDO

CiA

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
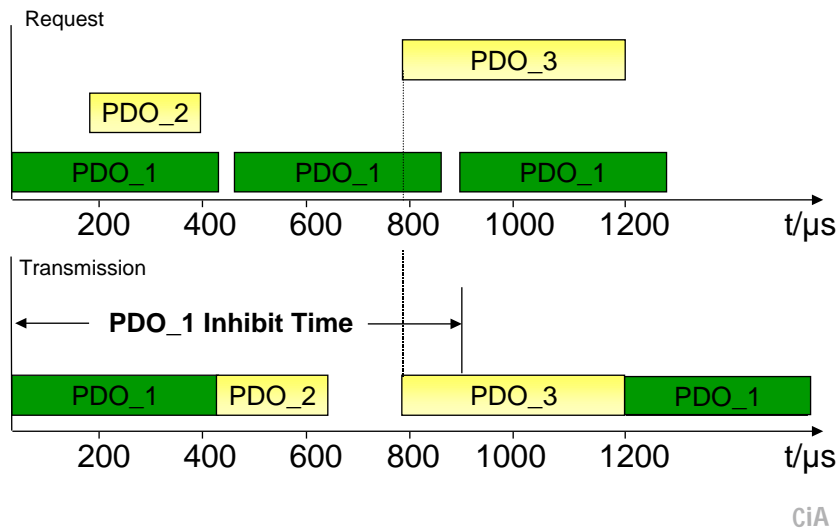[5] application event is defined in the device profile

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'

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.

Virtual Transmission Windows

PDO_n Event Time

PDO_n Inhibit Time

PDO_n

PDO_n

1 2 3 4 5 6 7 t/ms

CiA

Combining the Inhibit-Timer and the Event-Timer the system designer can assign to each Transmit-PDO an virtual window, in which the object is transmitted via the network. The earliest time of PDO transmission is the inhibit-time and the latest time of sending the PDO is the event-time. This virtual transmission window can be used to reduce busload.

PDO Parameter Sets

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

CiA

Communication Parameter |

Index	Sub-Index	Description	Data Type
1XXXh	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

Object Dictionary		
xxxxh	xxh	Appl.Object A
yyyyh	yyh	Appl.Object B
zzzzh	yyh	Appl.Object C

PDO_1 Mapping			
0	3		
1	yyyyh	yyh	8
2	zzzzh	zzh	16
3	xxxxh	xxh	8

PDO_1	Appl.Obj.B	Application Object C	Appl.Obj.A

Mapping Parameter

Index	Sub-Index	Description	Data Type
1XXXh	0h	number of mapped	Unsigned8
	1h	1st object	Unsigned32
	2h	2nd object	Unsigned32
	..	..	..
	40h	64th object	Unsigned32

31	16	15	8	7	0
16-bit Index		8-bit Sub-index		length	

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.

Service Data Object

Peer-to-Peer Communication

SDO Client

Node n-1   OD

SDO 1

ID 1 Data

ID 2 Data

Node n   OD

SDO Server OD

CiA

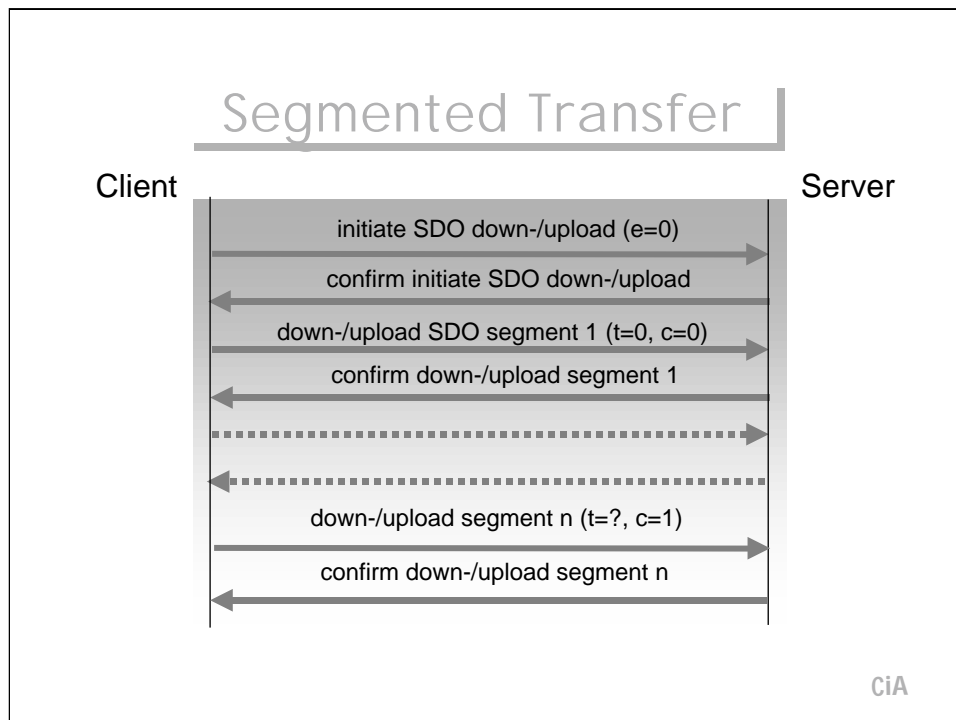With Service Data Objects (SDO) the access to entries of a device object dictionary is provided. One SDO uses two CAN Data Frames with different identifiers, because the communication is confirmed. By means of a SDO a peer-to-peer communication channel between two devices may be established. The owner 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.

Segmented Transfer

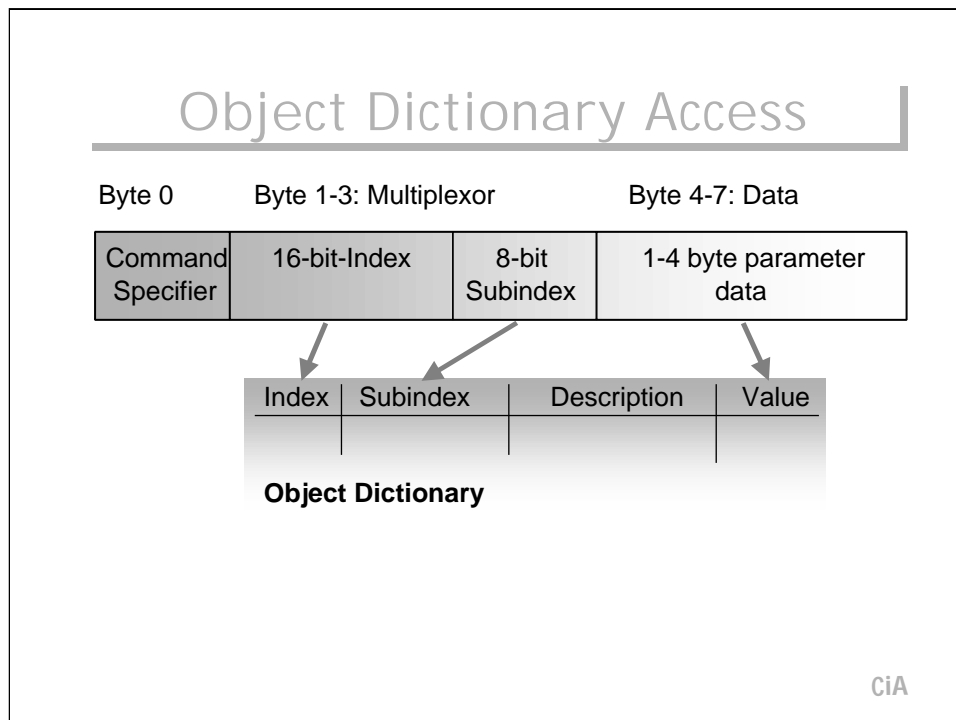| Client		Server
	initiate SDO down-/upload (e=0) →	
	← confirm initiate SDO down-/upload	
	down-/upload SDO segment 1 (t=0, c=0) →	
	← confirm down-/upload segment 1	
	·········→	
	←·········	
	down-/upload segment n (t=?, c=1) →	
	← confirm down-/upload segment n	

CiA

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.

Object Dictionary Access

Byte 0	Byte 1-3: Multiplexor		Byte 4-7: Data
Command Specifier	16-bit-Index	8-bit Subindex	1-4 byte parameter data

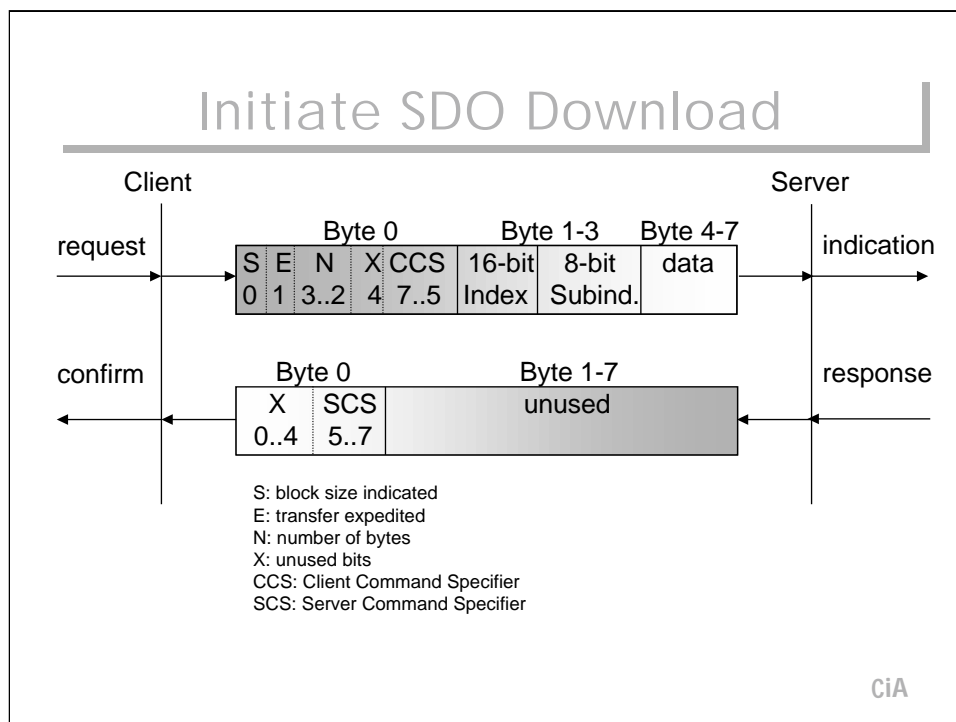| Index	Subindex	Description	Value

Object Dictionary

CiA

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.

Initiate SDO Download

Client   Server

request   indication

					Byte 0		Byte 1-3	Byte 4-7
S	E	N	X	CCS		16-bit	8-bit	data
0	1	3..2	4	7..5		Index	Subind.	

confirm   response

Byte 0		Byte 1-7
X	SCS	unused
0..4	5..7	

S: block size indicated
E: transfer expedited
N: number of bytes
X: unused bits
CCS: Client Command Specifier
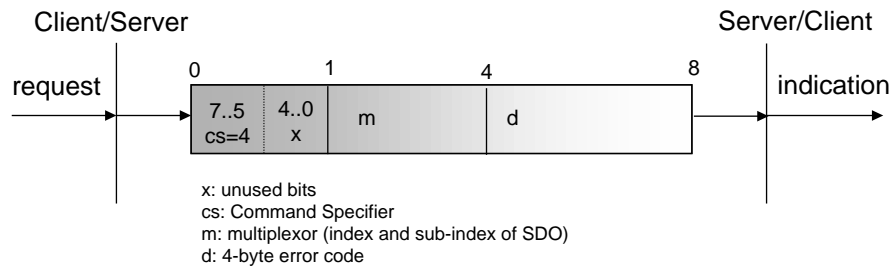SCS: Server Command Specifier

CiA

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.

Download SDO Segment

	Client					Server	

request →

	Byte 0			Byte 1-7	
C	N	t	CCS	segment	
0	1..3	4	5..7	data	

→ indication →

confirm ←

	Byte 0		Byte 1-7	
X	t	SCS	unused	
0..3	4	5..7		

← response

C: indicates the last segment
N: number of bytes
t : toggle bit
X: unused bits
CCS: Client Command Specifier
SCS: Server Command Specifier

CiA

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.
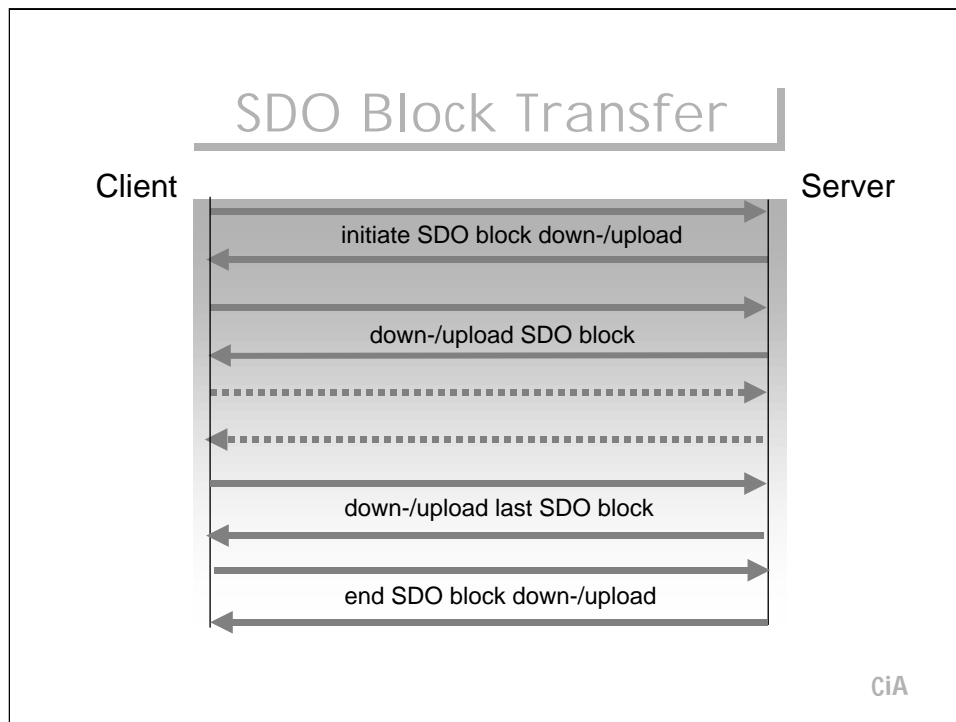
Abort SDO Transfer

Client/Server   Server/Client

request   indication

0	1	4	8
7..5 cs=4	4..0 x	m	d

x: unused bits
cs: Command Specifier
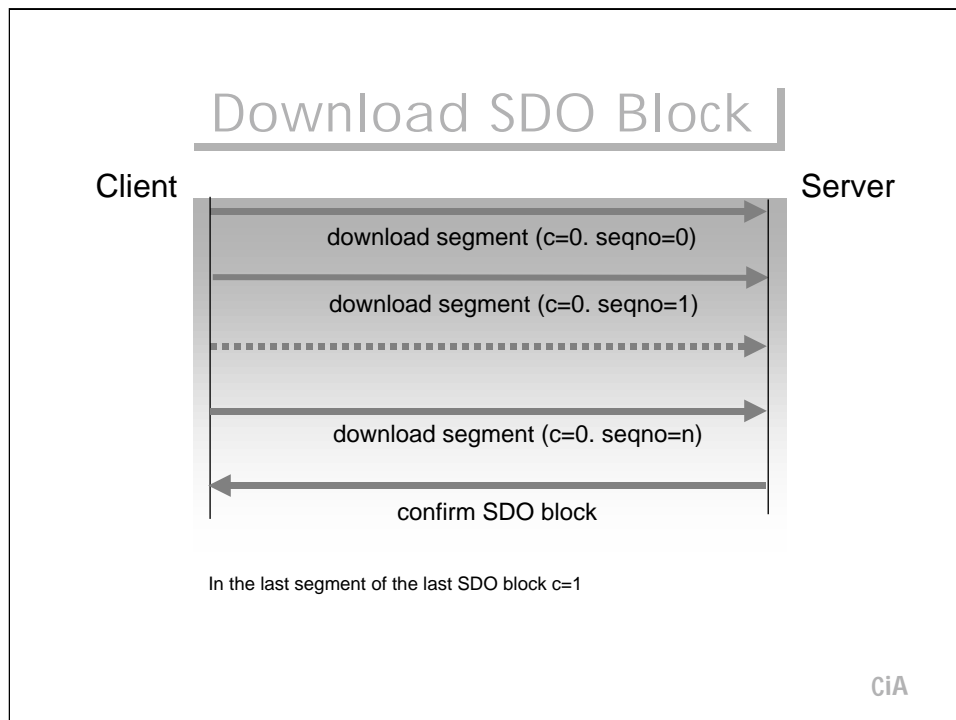m: multiplexor (index and sub-index of SDO)
d: 4-byte error code

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.

SDO Block Transfer

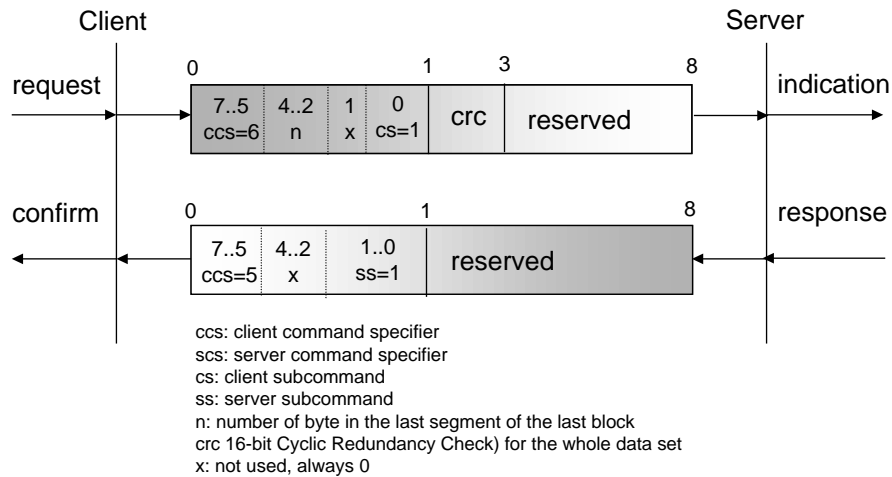| Client		Server
	initiate SDO block down-/upload	
	down-/upload SDO block	
	down-/upload last SDO block	
	end SDO block down-/upload	

CiA

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.

Download SDO Block

Client | Server

download segment (c=0. seqno=0)

download segment (c=0. seqno=1)

download segment (c=0. seqno=n)

confirm SDO block

In the last segment of the last SDO block c=1

CiA

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

Client   Server

request → indication

| 7..5 ccs=6 | 4..2 n | 1 x | 0 cs=1 | crc | reserved |

confirm ← response

| 7..5 ccs=5 | 4..2 x | 1..0 ss=1 | reserved |

ccs: client command specifier
scs: server command specifier
cs: client subcommand
ss: server subcommand
n: number of byte in the last segment of the last block
crc 16-bit Cyclic Redundancy Check) for the whole data set
x: not used, always 0

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 polynominal 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 Set

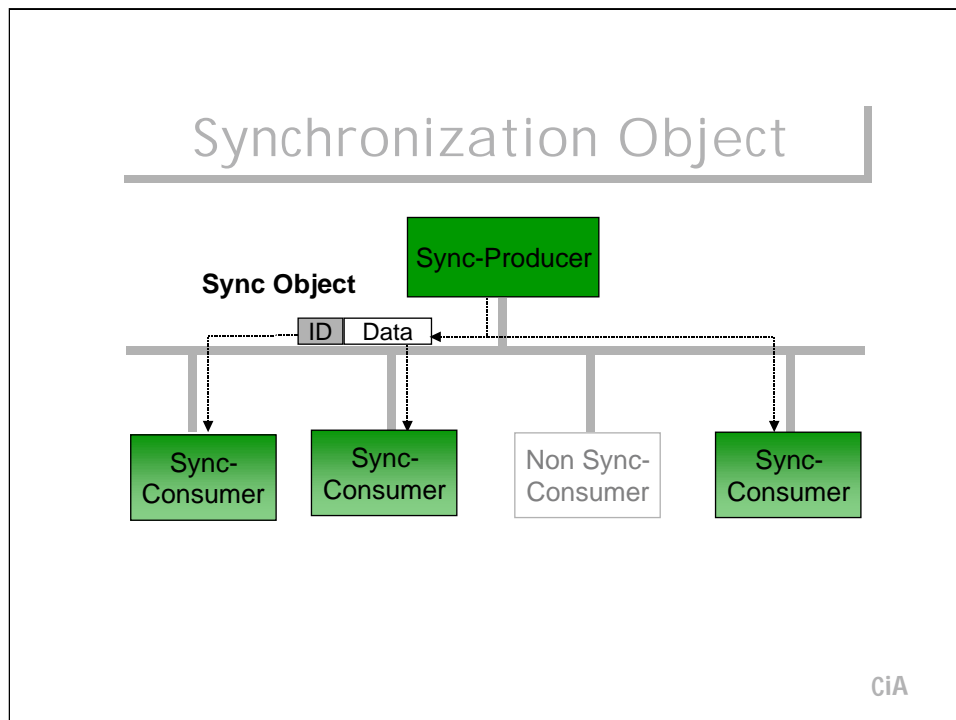|					
Server SDO Parameter (22H)					
1200	RECORD	1st Server SDO parameter	SDOParameter	ro	O
1201	RECORD	2nd Server SDO parameter	SDOParameter	rw	M/O**
.....	.....	.....	.....	.....	.....
127F	RECORD	128th Server SDO parameter	SDOParameter	rw	M/O**
Client SDO Parameter (22H)					
1280	RECORD	1st Client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

CiA

SDO Parameter Record

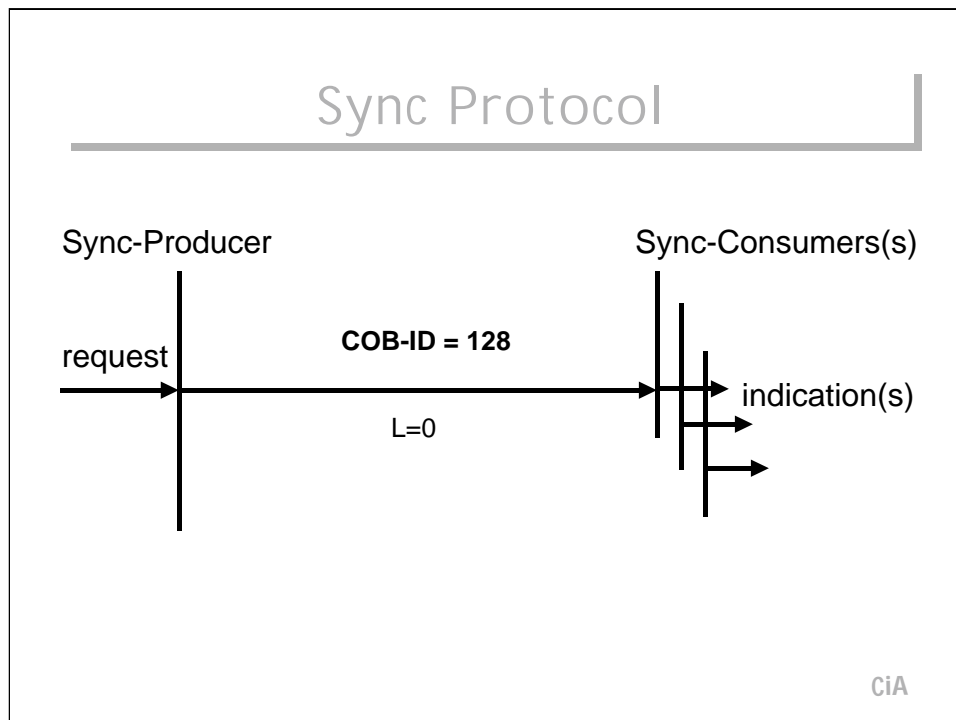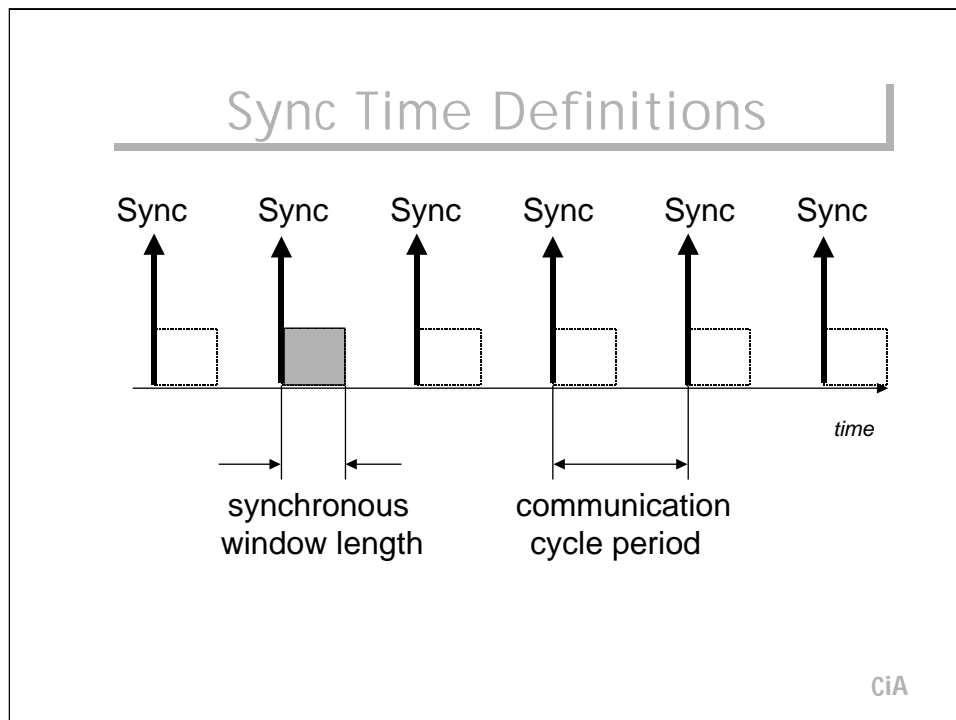| Index	Sub-Index	Description	Data Type
1XXXh	0h	number of entries	Unsigned8
	1h	COB-ID client -> server	Unsigned32
	2h	COB-ID client <- server	Unsigned32
	3h	node ID	Unsigned8

CiA

Synchronization Object

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.

Sync Protocol

Sync-Producer   Sync-Consumers(s)

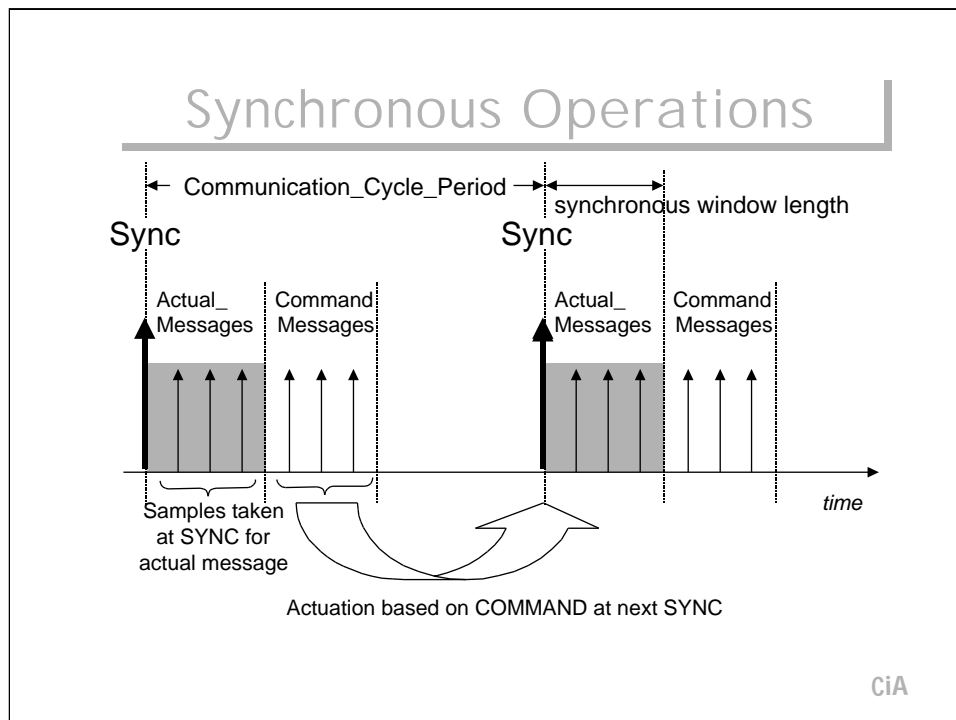request   **COB-ID = 128**   indication(s)

L=0

CiA

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.

Sync Time Definitions

Sync   Sync   Sync   Sync   Sync   Sync

time

synchronous
window length

communication
cycle period

CiA

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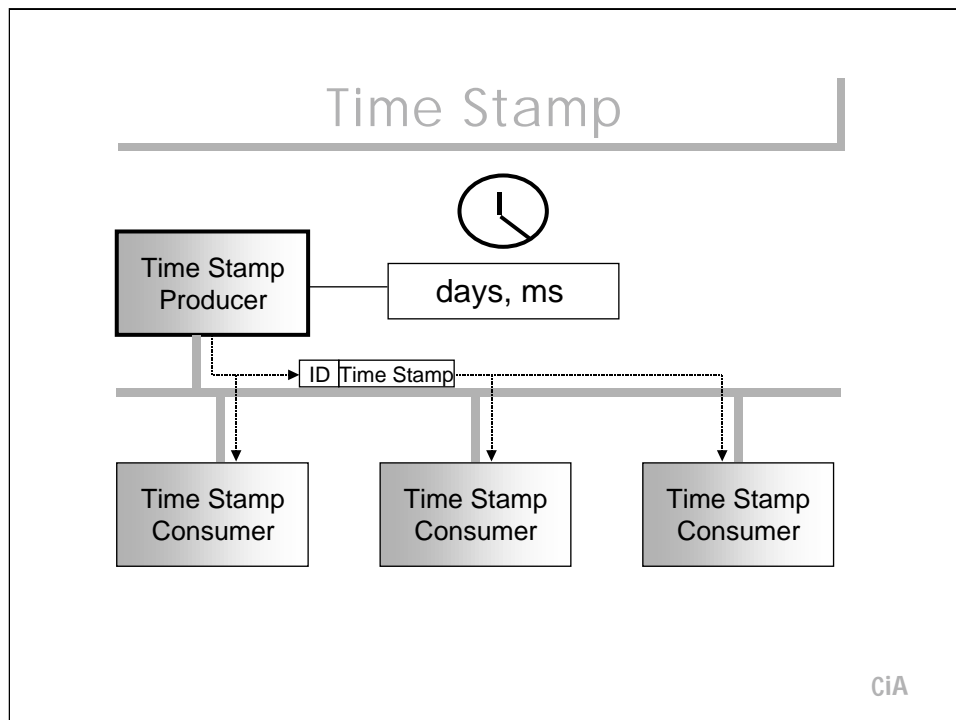
There can be a time jitter in transmission by the Sync-Producer corresponding approximately to the latency due to another message being transmitted just before the Sync Object. You also have to consider the case, that the Sync is transmitted twice.

Synchronous Operations

Communication_Cycle_Period

synchronous window length

Sync

Sync

Actual_ Messages

Command Messages

Actual_ Messages

Command Messages

time

Samples taken at SYNC for actual message

Actuation based on COMMAND at next SYNC

CiA

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 window. In the next time window, the CANopen devices transmit the output values to the actuators. However, these values are not valid until the next Sync message is received.

You can configure all the Receive-PDOs with an asynchronous transmission type, so that the PDOs will be transmitted cyclically and the commands will be valid immediately after receiving the PDOs.

Time Stamp

Time Stamp Producer — days, ms

ID | Time Stamp

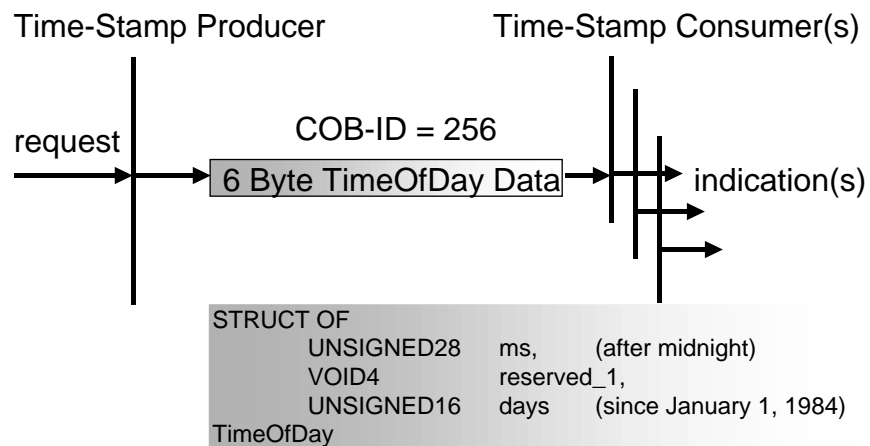Time Stamp Consumer | Time Stamp Consumer | Time Stamp Consumer

CiA

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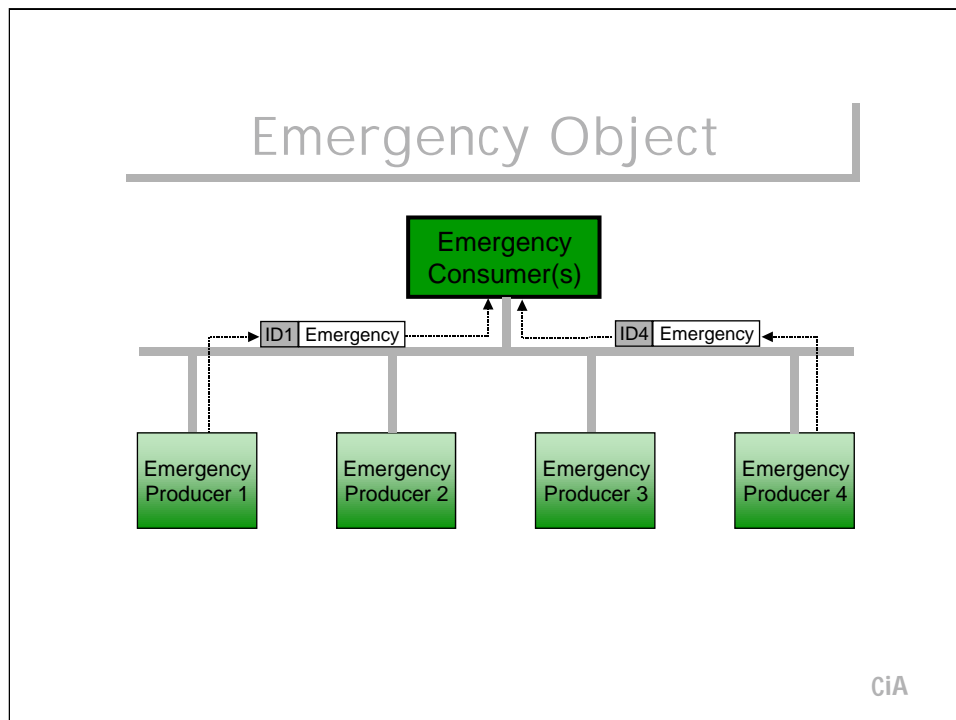
The high-resolution time-stamp is encoded as unsigned32 with a resolution of 1 microsecond which means that the time counter restarts every 72 minutes. It is configured by mapping the high resolution time-stamp (object 1013h) into a PDO.

Time-Stamp Protocol

Time-Stamp Producer   Time-Stamp Consumer(s)

request   COB-ID = 256

6 Byte TimeOfDay Data   indication(s)

```
STRUCT OF
      UNSIGNED28     ms,        (after midnight)
      VOID4          reserved_1,
      UNSIGNED16     days     (since January 1, 1984)
TimeOfDay
```
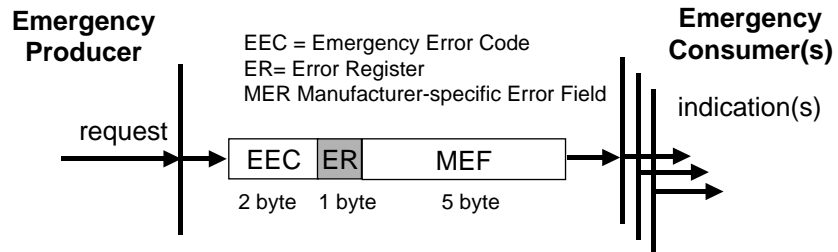
CiA

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 Object

Emergency Consumer(s)

ID1 Emergency   ID4 Emergency

Emergency Producer 1   Emergency Producer 2   Emergency Producer 3   Emergency Producer 4

CiA

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 Protocol

Emergency Producer

EEC = Emergency Error Code
ER = Error Register
MER Manufacturer-specific Error Field

Emergency Consumer(s)

indication(s)

request

EEC	ER	MEF
2 byte	1 byte	5 byte

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)

NMT-Master

ID 0 NMT

NMT-Slave

NMT-Slave

NMT-Slave

NMT-Slave
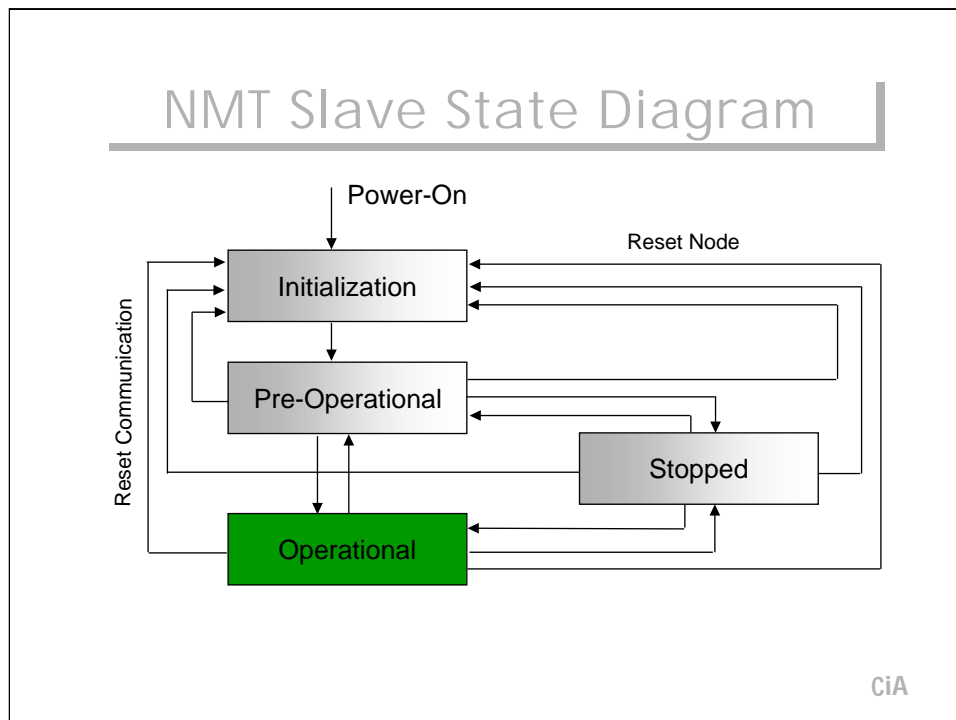
CiA

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.
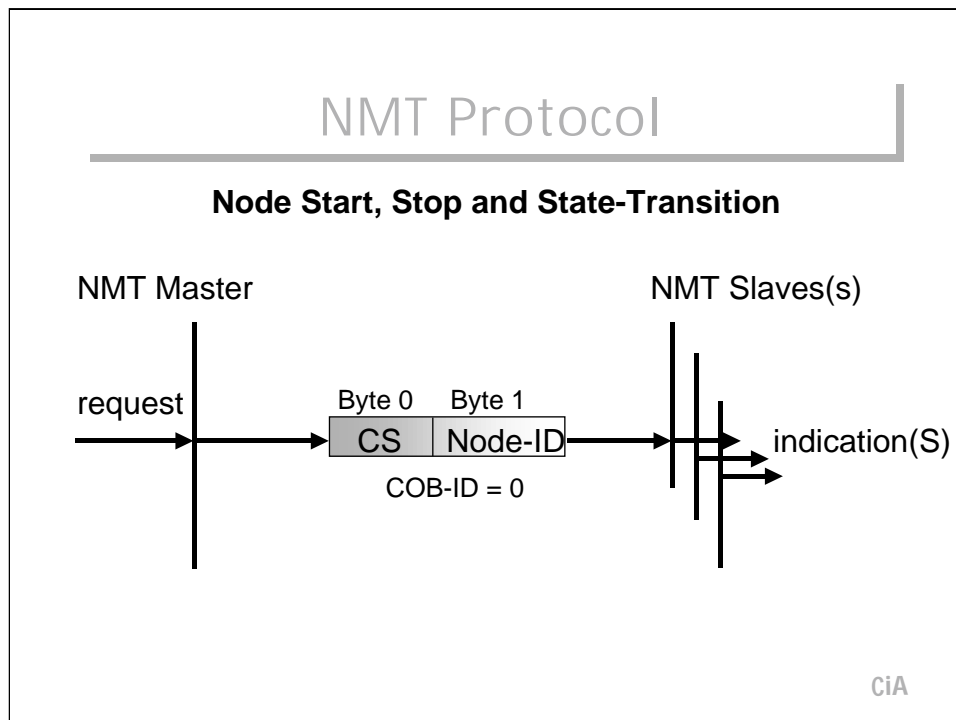
NMT Slave State Diagram

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).
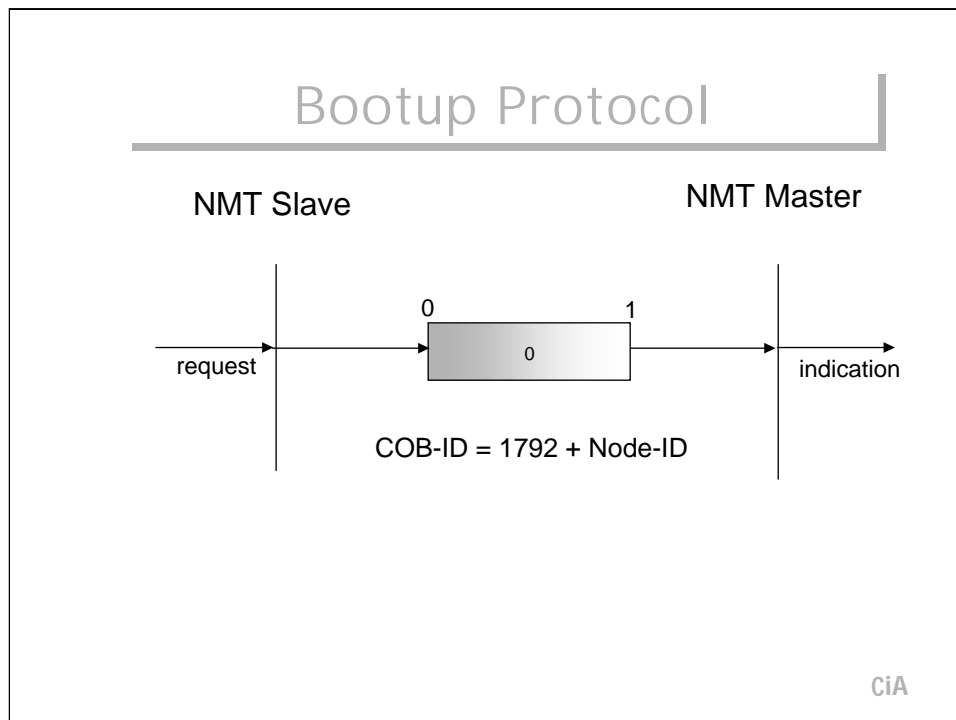
NMT Protocol

Node Start, Stop and State-Transition

NMT Master   NMT Slaves(s)

request

```
          Byte 0   Byte 1
          CS       Node-ID
          COB-ID = 0
```

indication(S)

CiA

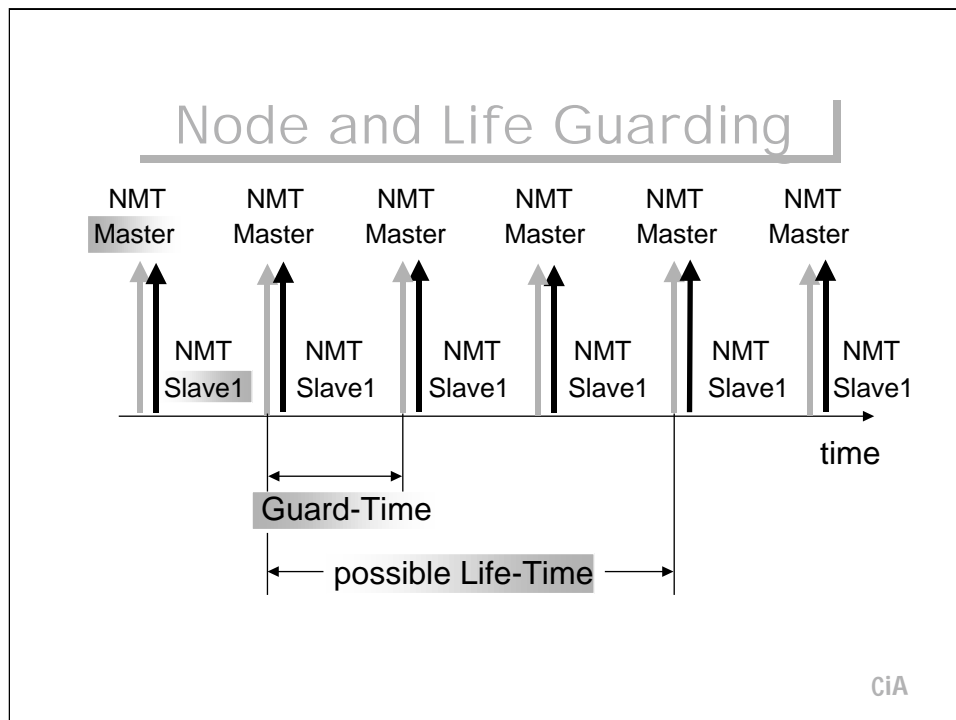CANopen Network Management provides the following five services, which can be distinguished by the Command Specifier (CS):

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.

Initialization State

Reset Application

Reset Communication

Initialization

Power-On

Pre-Operational

CiA

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 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 1011h), the power-on values are the default values according to the communication and device profile.

Bootup Protocol

NMT Slave   NMT Master

request   0   1   indication
                          [0   ]

COB-ID = 1792 + Node-ID

CiA

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.

Node and Life Guarding

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.

Node Guarding Protocol

NMT Master		NMT Slave

Remote Frame

request → → indication

Bit 7	Bit 6 - 0
Toggle-bit	Node State

confirmation ← ← response

Remote Frame

request → → indication

Bit 7	Bit 6 - 0
Toggle-bit	Node State

confirmation ← ← response

Network Event*   *Node Event**

indication   * if guarding error   indication

CiA

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 got to a save state.

Heartbeat Protocol

Heartbeat Producer   Heartbeat Consumer(s)

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

Bit-No.: 10   ...   0

COB-Identifier | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Function Code   Module-ID

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
PDO2 (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
LSS slave	1111	2020	-
LSS master	1111	2021	-

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 deleted. 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.
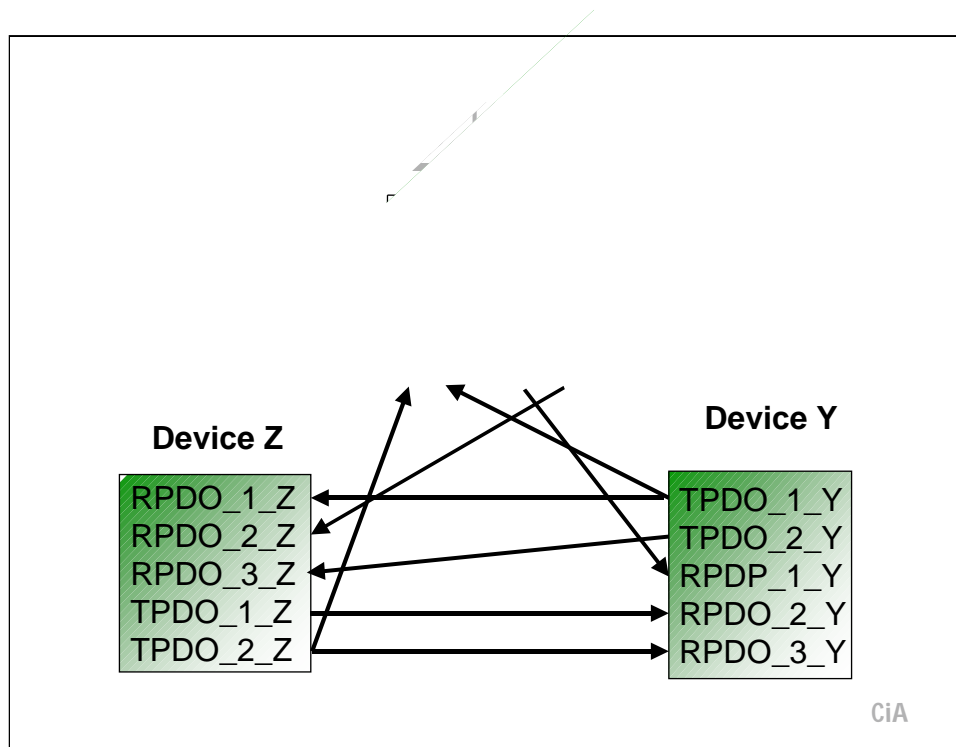
Predefined PDO Connections

Master
RPDO_1_M
RPDO_2_M
TPDO_1_M
TPDO_2_M
RPDO_3_M
RPDO_4_M
TPDO_3_M
TPDO_4_M

Slave X
TPDO_1_X
TPDO_2_X
RPDO_1_X
RPDO_2_X

Slave Y
TPDO_1_Y
TPDO_2_Y
RPDO_1_Y
RPDO_1_Y

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).

Device Z

RPDO_1_Z
RPDO_2_Z
RPDO_3_Z
TPDO_1_Z
TPDO_2_Z

Device Y

TPDO_1_Y
TPDO_2_Y
RPDP_1_Y
RPDO_2_Y
RPDO_3_Y

CiA

PDO Linking enables to establish flexible Multi-Master connections. These connections can be realized during Pre-Operational State by using comfortable configuration tools. The entries in the object dictionaries will be deleted and created 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 from 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 introduce 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.
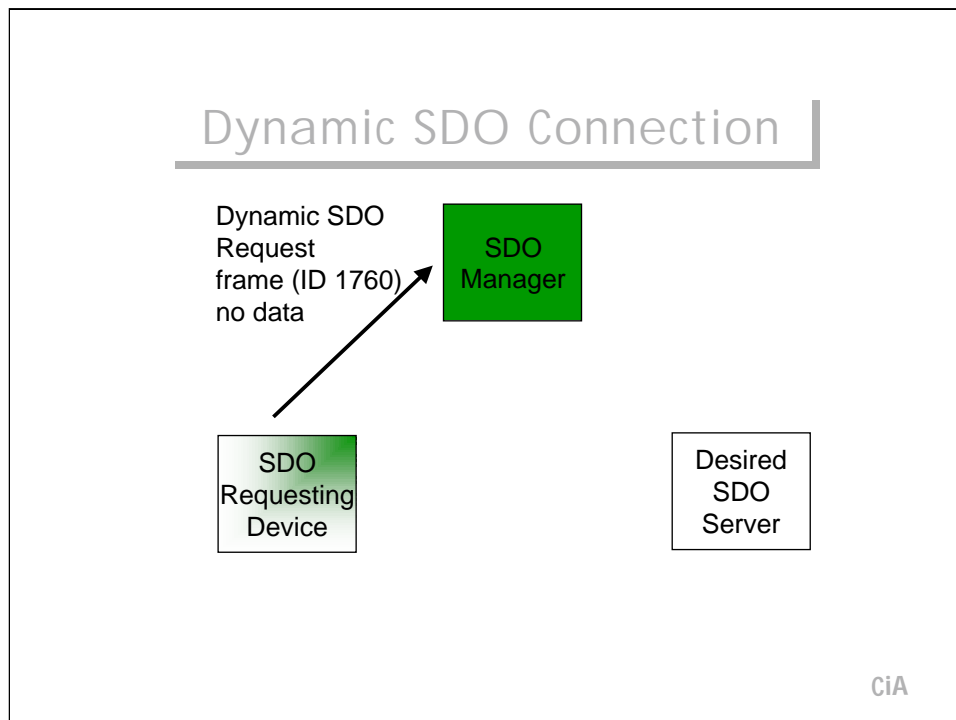
Programmable Devices

- 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).

Dynamic SDO Connection

Dynamic SDO
Request
frame (ID 1760)
no data

SDO
Manager

SDO
Requesting
Device

Desired
SDO
Server

CiA

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.

Dynamic SDO Connection

Scan and Establishment of SDO (SRD is Client)

SDO Manager

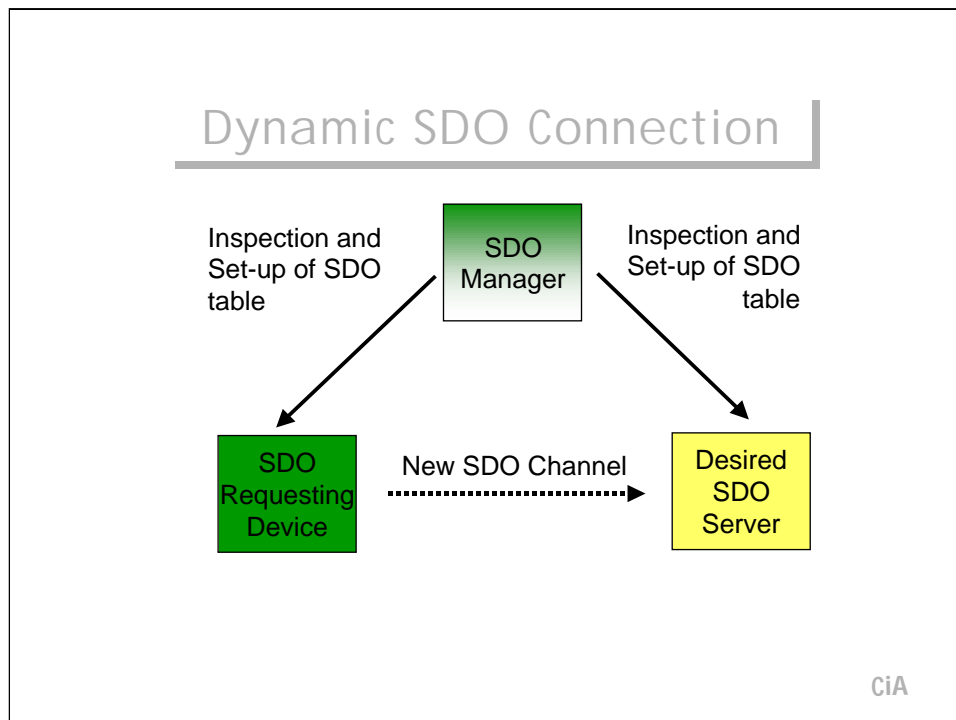New SDO request

SDO Requesting Device

Desired SDO Server

CiA

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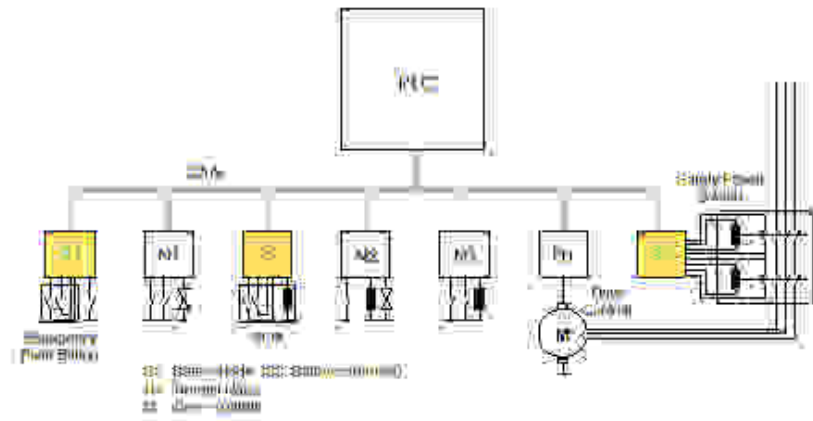
Hereafter the SDR can perform requests to the SDO Manager via the new SDO channel. It will use this to request a connection to the "Desired SDO Server".

Dynamic SDO Connection

Inspection and
Set-up of SDO
table

SDO
Manager

Inspection and
Set-up of SDO
table

SDO
Requesting
Device

New SDO Channel

Desired
SDO
Server

CiA

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.
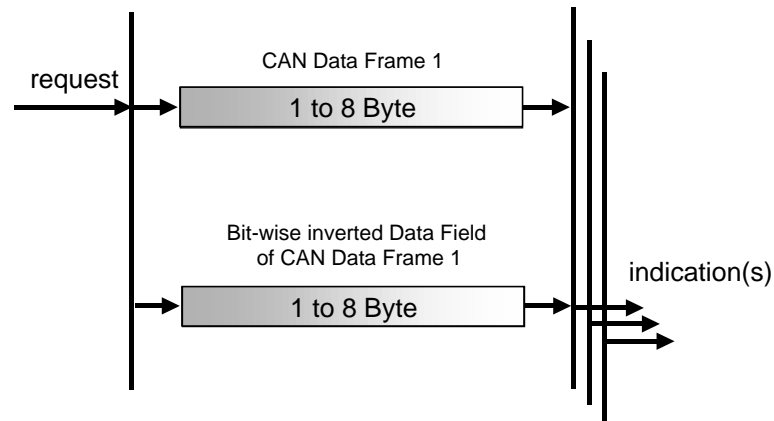
CANopen users require transmission of standard communication objects and transmission of safety-relevant data on the very same physical network. The CANopen communication profile for safety-relevant data transmission is compatible with the CiA DS-301 Version 4.0 CANopen application layer and communication 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.
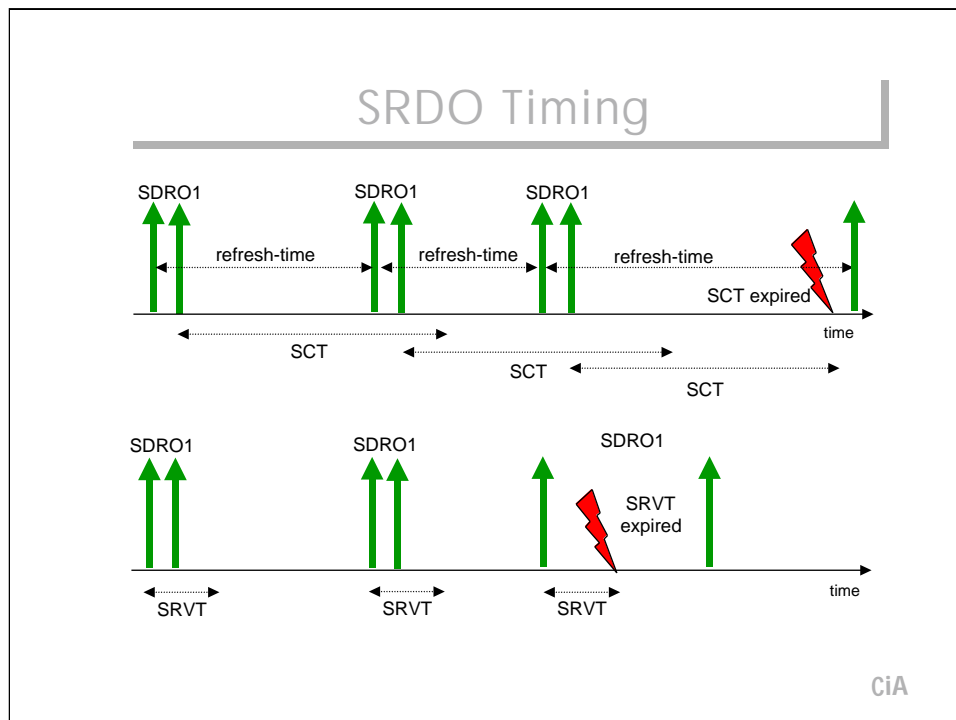
Safety-Relevant Data Object

request

CAN Data Frame 1

1 to 8 Byte

Bit-wise inverted Data Field
of CAN Data Frame 1

1 to 8 Byte

indication(s)

CiA

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 indentifier 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.
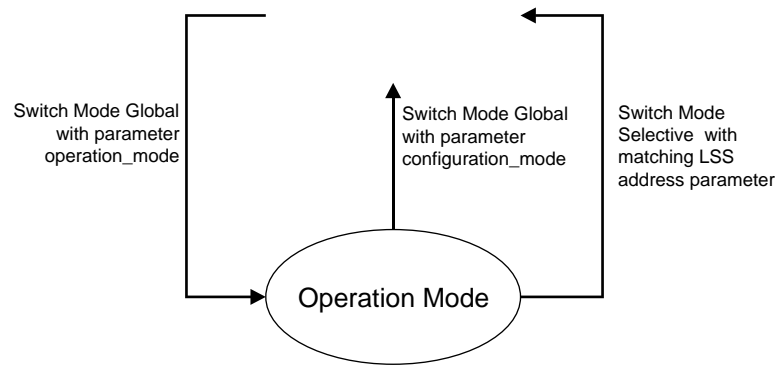
Pre-Defined Identifier

Node ID	SRDO	Object Entry	SRDO Identifiers Data	Complement	Direction
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.
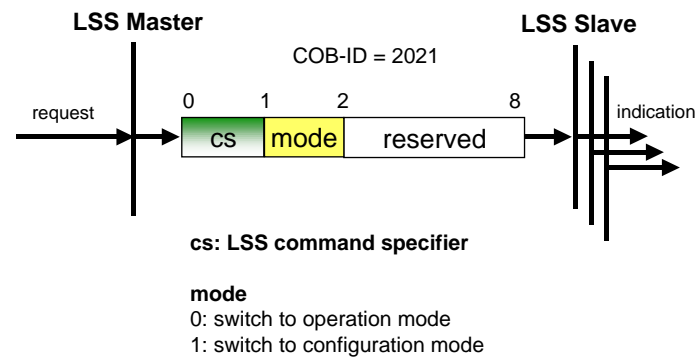
All other communication relationships (one producer - many user, many producer - one user or mixed) are possible, but must be configured accordingly to the application.
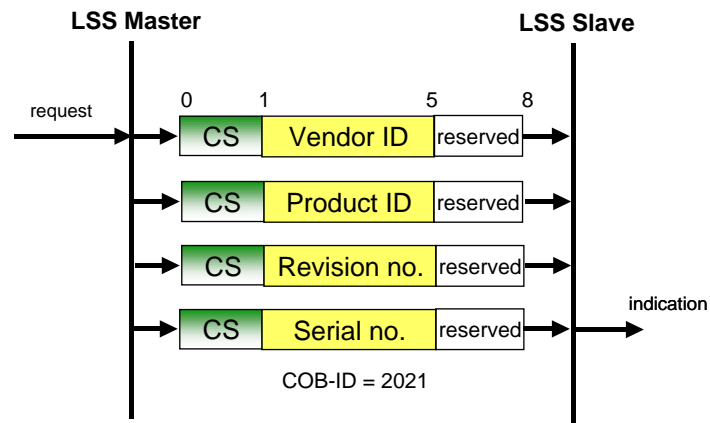
Layer Setting Services

Switch Mode Global
with parameter
operation_mode

Switch Mode Global
with parameter
configuration_mode

Switch Mode
Selective with
matching LSS
address parameter

Operation Mode

Switch Mode Global

LSS Master

COB-ID = 2021

LSS Slave

request

| 0 | 1 | 2 | | 8 |

cs | mode | reserved

indication

cs: LSS command specifier

mode
0: switch to operation mode
1: switch to configuration mode

CiA

Switch Mode Selective

LSS Master   **LSS Slave**

request

0   1   5   8

CS	Vendor ID	reserved
CS	Product ID	reserved
CS	Revision no.	reserved
CS	Serial no.	reserved

indication

COB-ID = 2021

CiA

Configure Node ID

LSS Master   **LSS Slave**

COB-ID = 2021

request ——→ | cs | new ID | reserved | ——→ indication
             0   1   2   8

confirmation ←—— | cs | ec | se | reserved | ←—— response
                  0   1   2   3   8

COB-ID = 2020
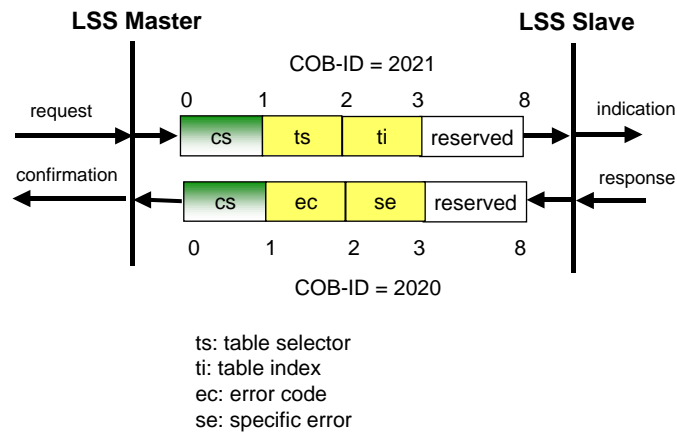
new ID: Node ID to be configured
cs: command specifier
ec: error code
se: specific error

CiA

Configure Bit Timing
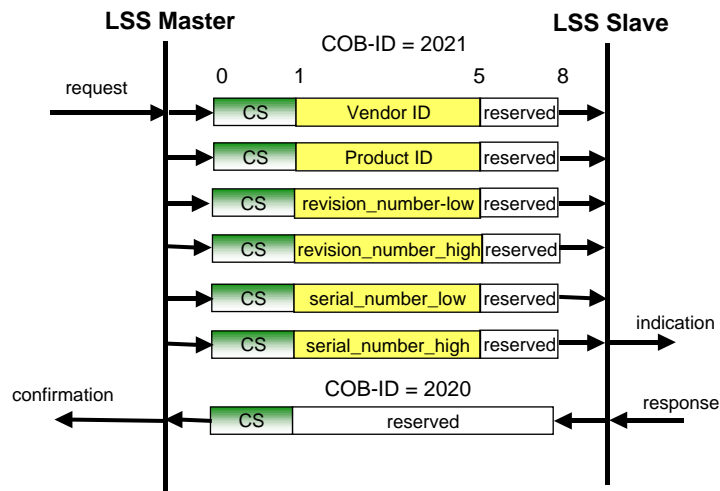
LSS Master   **LSS Slave**

COB-ID = 2021

request → | cs | ts | ti | reserved | → indication
         0   1   2   3   8

confirmation ← | cs | ec | se | reserved | ← response
              0   1   2   3   8

COB-ID = 2020

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

CiA

Activate Bit Timing

LSS Master   **LSS Slave**

COB-ID = 2021

request

0	1	3	8	indication
cs	switch_delay	reserved		

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 DSP-404:   **Measuring Devices and Closed-Loop Controllers**
- CiA DSP-406:   **Encoders**
- CiA DSP-408:   **Proportional Hydraulic Valves**
- CiA WD-409:   **Door Control (Railways)**
- CiA WDP-411   **Truck Gateway**
- CiA WDP-413   **X-Ray Generator**

Interface Profile Specification

- CiA DSP-405:   **IEC 1131 Programmable Devices**
- CiA WDP-410   **ASAM/GDI**

Application Profile Specification

- CiA WD-407:   **Public Transportation**
- CiA WDP-412   **Maritime Applications**

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
- Default PDO Transmission Parameters
- Default PDO Mapping Parameter
- Additional Emergency Codes
- Additional Data Types
- Application Object Descriptions

CiA

Device Type Object

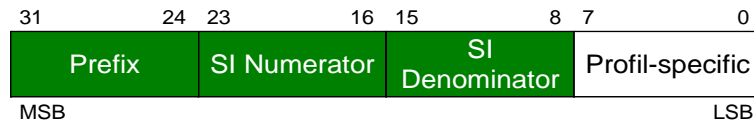MSB   LSB

Additional Information	Device Profile Number

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

31	24	23	16	15	8	7	0

Prefix	SI Numerator	SI Denominator	Profil-specific

MSB   LSB

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>

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 it's 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 TPDO | 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 RPDO | 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 to 4th TPDO | 16- bit input | 16- bit input | 16- bit input | 16- bit input |

4 x 16-bit analog inputs transmitted asynchronously

| 2nd to 4th RPDO | 16- bit output | 16- bit output | 16- bit output | 16- bit output |

4 x16-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 delete the default PDOs and to create new PDOs.

The default PDO communication parameters are specified in the objects 1400h to 1403h and 1800h to 1803h. The transmission for all PDOs is by default asynchronous, and the inhibit time is 0. The default PDO mappings are specified in the objects 1600h to 1603h and 1A00h to 1A03h.

Optional I/O Access

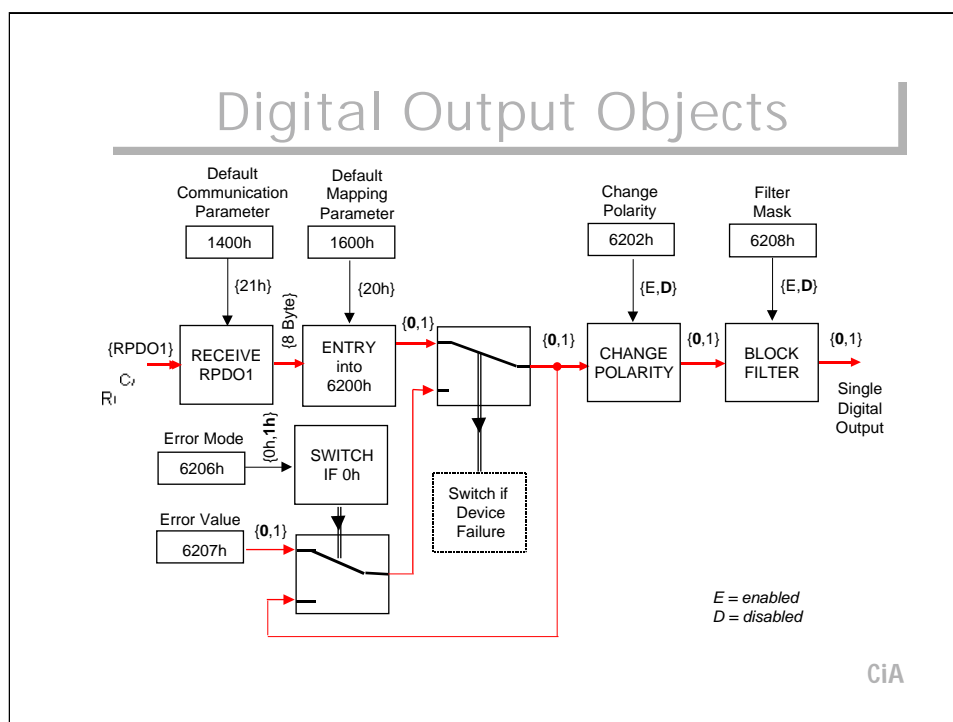| Digital Inputs	Analog Inputs
• single-bit access (6020h to 6027h)	• Integer8 access (6400h)
• 2-byte access (6100h)	• Integer32 access (6402h)
• 4-byte access (6120h)	• Float access (6403h)
	• manufacturer-specific (6404h)
Digital Outputs	**Analog Outputs**
• single-bit access (6220h to 6227h)	• Integer8 access (6410h)
• 2-byte access (6300h)	• Integer32 access (6412h)
• 4-byte access (6320h)	• Float access (6413h)
	• 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.
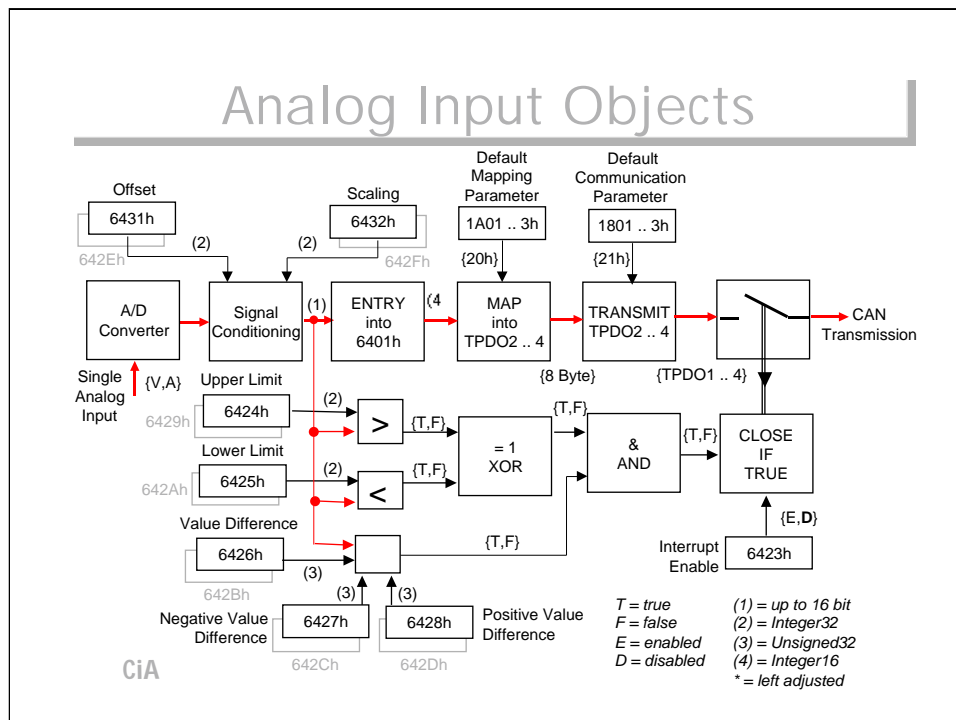
Digital Input Objects

The default TPDO of I/O modules with digital inputs is transmitted asynchronously if one input is changed. With the object 6005h (interrupt enable) you can disable the transmission of PDOs. The objects 6002h (polarity input) and 6003h (filter constant) are optional and disabled by default.

Devices with PDO event-timer transmit the corresponding data frame, when the timer elapses or one digital input is changed. A cyclic synchronous TPDO is transmitted only if a specific Sync telegram has been received - independently of the input states. An acyclic synchronous TPDO is send if Sync telegram has been received and an input has been changed.
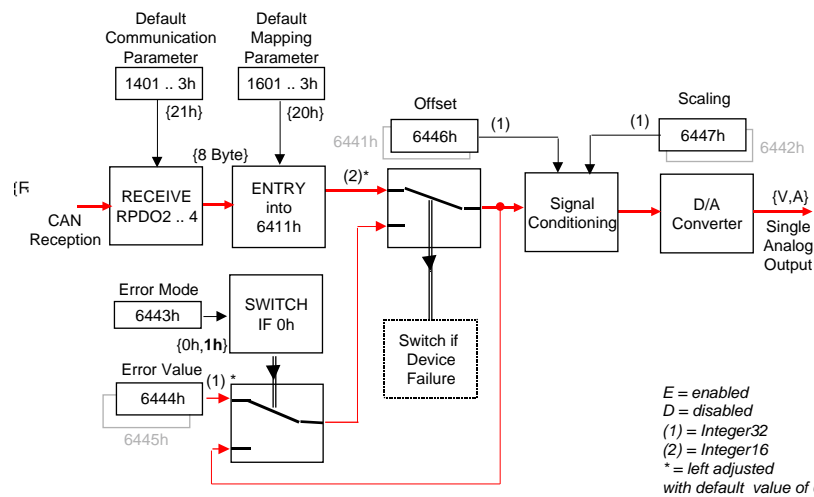
Digital Output Objects

The default RPDO of I/O modules with digital outputs is received asynchronously. If an internal device failure occurs the output is set to "0" by default. The error value of an output also can be set to "1" by the object 6207h (error value). Another possibility is to keep the last output value, than the object 6206h (error mode) has to be set to 0h.

Analog Input Objects

The transmit-trigger configuration for PDO with analog inputs provides a lot options. By default, the TPDO is not transmitted. The transmission has to be enabled by the object 6423h (interrupt enable). All PDO trigger parameter such as upper and lower limit as well as the value differences objects can be implemented in Integer and Unsigned formats as well as in Float format. If both format options are implemented, the change in one object is automatically changing the correspondent parameter in the other format.

Analog Output Objects

The default value of an analog output is 0. In case of internal device failure the default error value of 0 (Integer32) is set. This can be changed to any other value or the output value can be kept.

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).

Drive Profile Architecture

CANopen Device Profile CiA DSP-402

Modes of Operation

| Homing Mode | Profile Position Mode | Inter-polated Position Mode | Profile Velocity Mode | Profile Torque Mode | | Velocity Mode |

Device Control Statemachine

CANopen Communication Profile CiA DS-301

CAN

CiA

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.

DS-402 State Machine

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.

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

CiA

A drive supporting more then 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	vl_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 | NDRT | Index | S-Index | NDRT = New Data Ready Transmit |

| R_PDO_2 | NDRR | Index | S-Index | NDRR = New Data Ready Receive |

| T_PDO_3 | 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 | 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.)
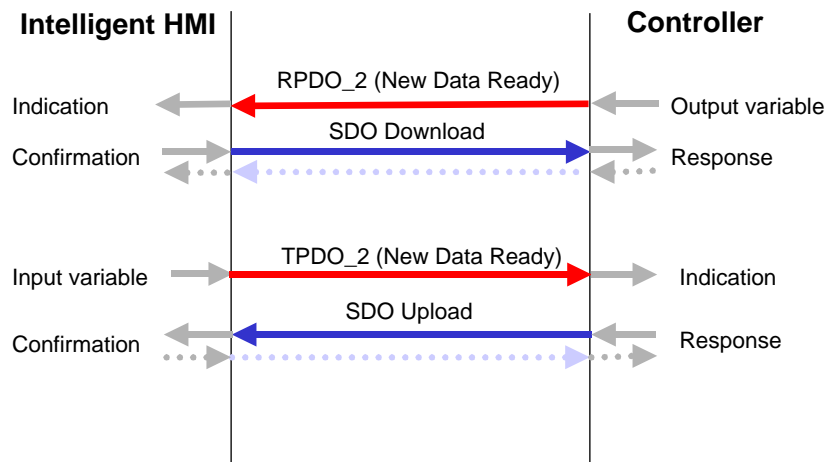
The second Transmit-PDO signalize that a variable value has changed. The index and sub-index relate to the changed variable.

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

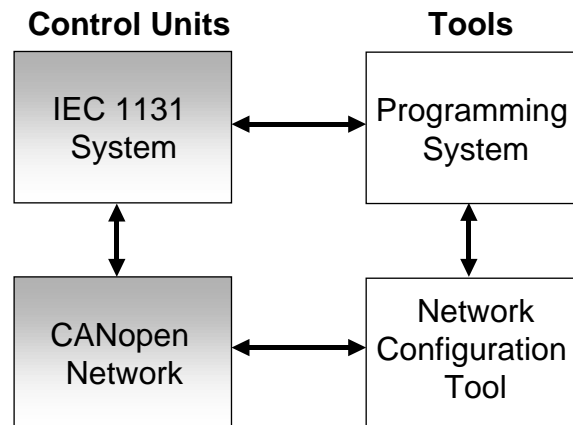| Intelligent HMI		Controller
Indication	RPDO_2 (New Data Ready)	Output variable
Confirmation	SDO Download	Response
Input variable	TPDO_2 (New Data Ready)	Indication
Confirmation	SDO Upload	Response

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

Control Units

Tools

IEC 1131 System ⟷ Programming System

CANopen Network ⟷ Network Configuration Tool

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

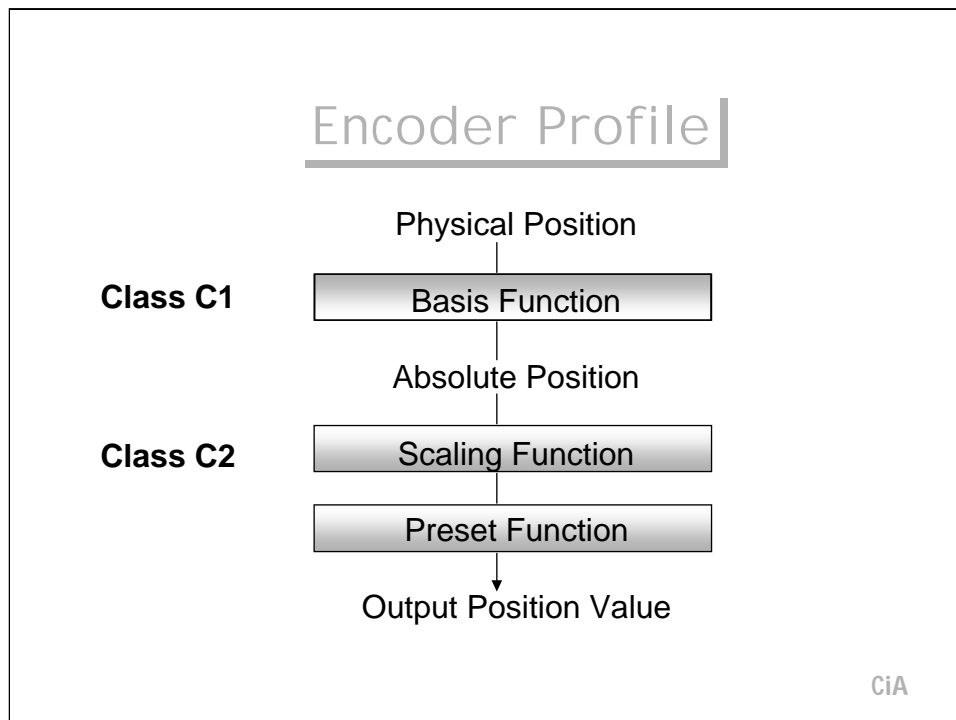| No.	Function Block
1	CIA405_RECV_EMY_DEV
2	CIA405_RECV_EMY
3	CIA405_SDO_WRITE4
4	CIA405_SDO_WRITE7
5	CIA405_SDO_WRITE14
6	CIA405_SDO_WRITE21
7	CIA405_SDO_READ4
8	CIA405_SDO_READ7
9	CIA405_SDO_READ14
10	CIA405_SDO_READ21

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.

Encoder Profile

Physical Position

Class C1 | Basis Function |

Absolute Position

Class C2 | Scaling Function |

| Preset Function |

Output Position Value

CiA

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.
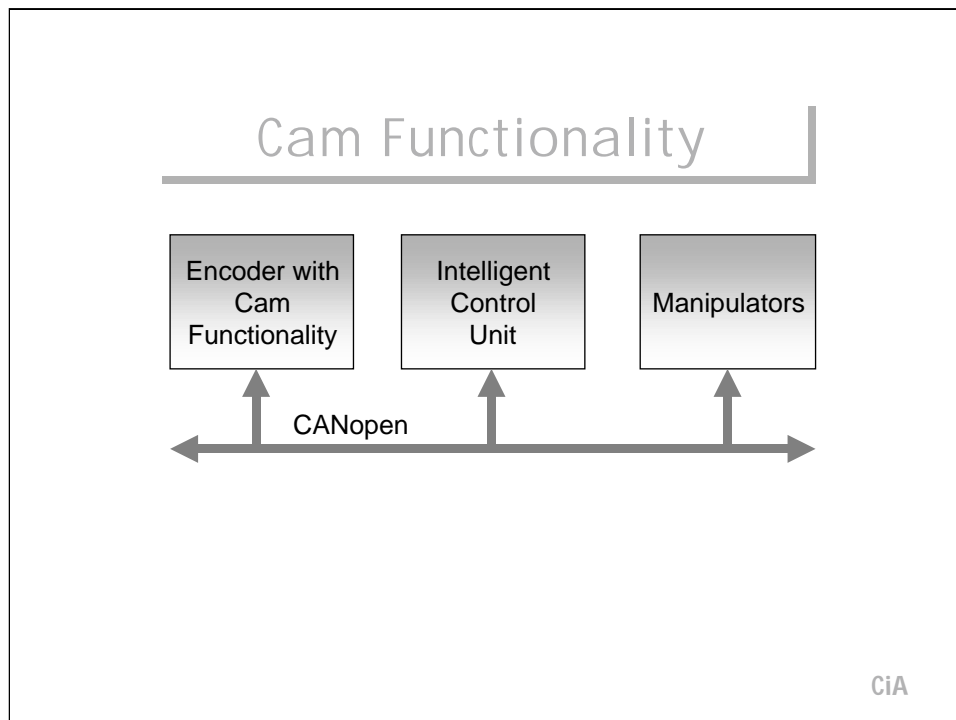
Encoder Profile (2)

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.

Cam Functionality

| Encoder with Cam Functionality | Intelligent Control Unit | Manipulators |

CANopen

CiA

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)
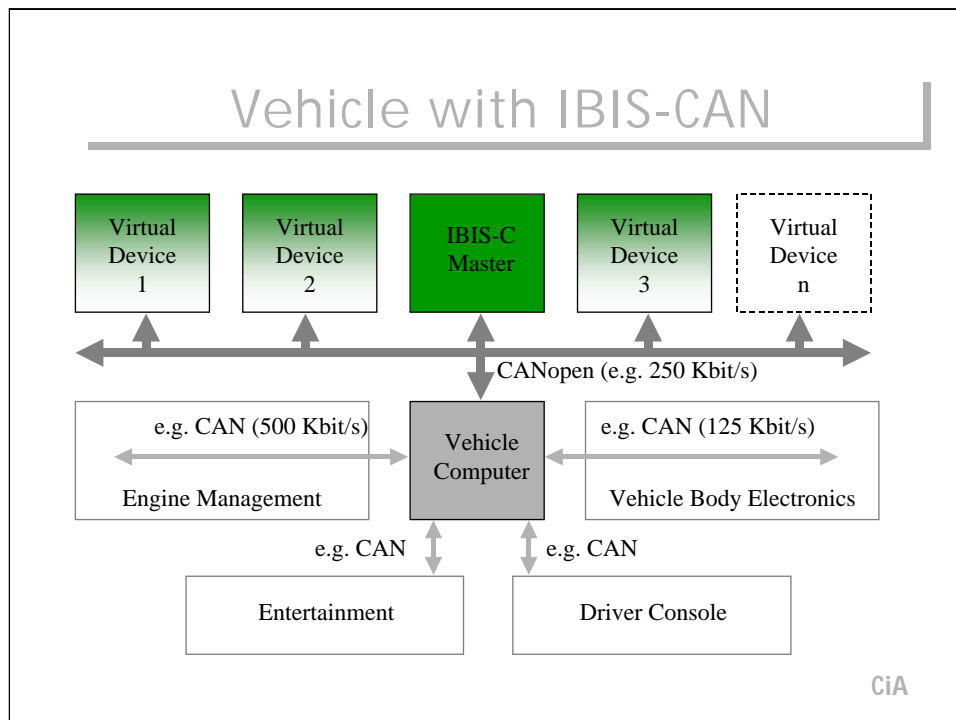
Byte 0	Byte 1	Byte 2	Byte 3

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:
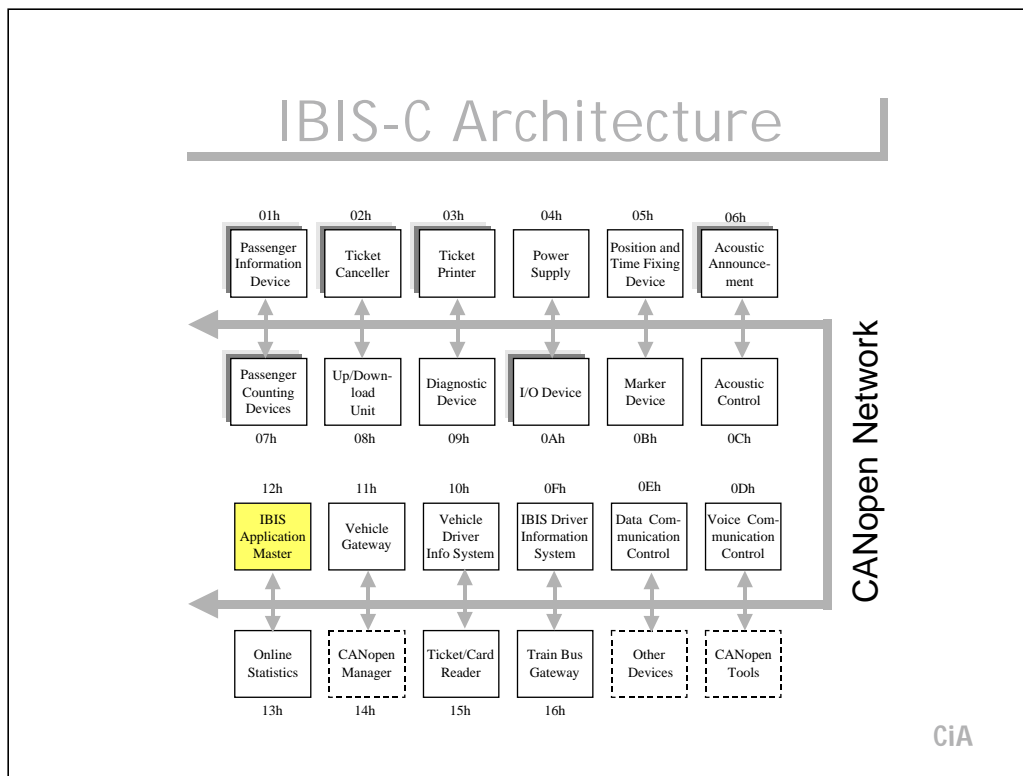
• Polled mode: The host polls the actual position value via a Remote Transmit Request.

• 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.

Vehicle with IBIS-CAN

| Virtual Device 1 | Virtual Device 2 | IBIS-C Master | Virtual Device 3 | Virtual Device n |

CANopen (e.g. 250 Kbit/s)

e.g. CAN (500 Kbit/s) — Vehicle Computer — e.g. CAN (125 Kbit/s)

Engine Management — Vehicle Body Electronics

e.g. CAN — e.g. CAN

Entertainment — Driver Console

CiA

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.

IBIS-C Architecture

01h	02h	03h	04h	05h	06h
Passenger Information Device	Ticket Canceller	Ticket Printer	Power Supply	Position and Time Fixing Device	Acoustic Announcement
Passenger Counting Devices	Up/Download Unit	Diagnostic Device	I/O Device	Marker Device	Acoustic Control
07h	08h	09h	0Ah	0Bh	0Ch
12h	11h	10h	0Fh	0Eh	0Dh
IBIS Application Master	Vehicle Gateway	Vehicle Driver Info System	IBIS Driver Information System	Data Communication Control	Voice Communication Control
Online Statistics	CANopen Manager	Ticket/Card Reader	Train Bus Gateway	Other Devices	CANopen Tools
13h	14h	15h	16h		

CANopen Network

CiA

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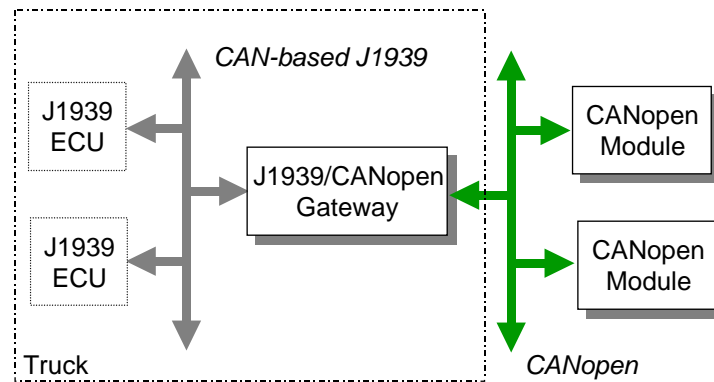
Many of the application objects are derived from the IBIS specification. IBIS-CAN application profile is the successor and is based on CANopen. The profile specifies additional virtual devices and related application objects.

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)
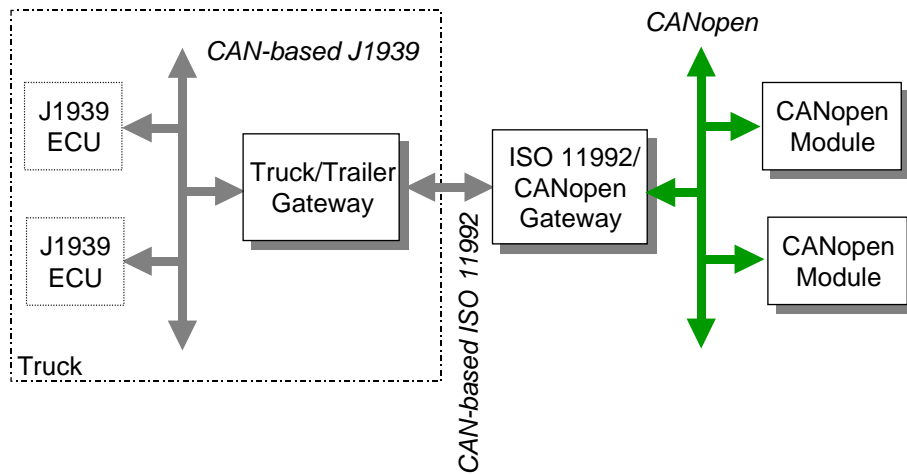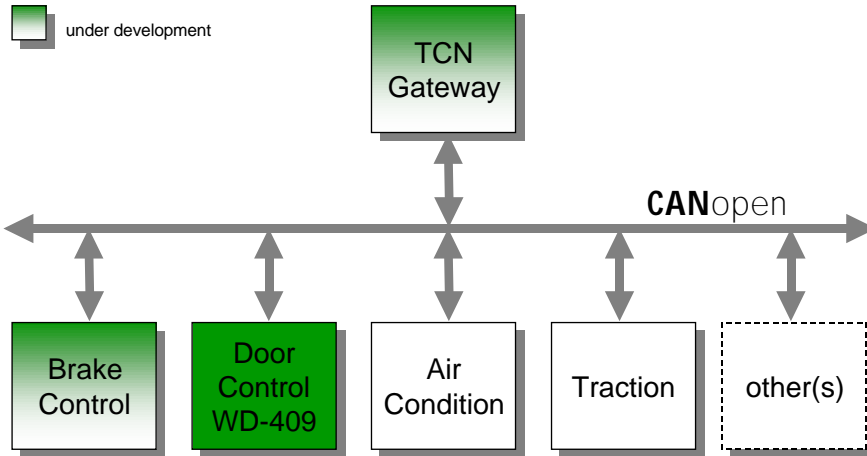- Special connectors including pin assignment

CiA

Truck Gateway

CAN-based J1939

J1939
ECU

J1939
ECU

J1939/CANopen
Gateway

Truck

CANopen
Module

CANopen
Module

CANopen

CiA

Truck Gateway (2)

CANopen Railway Profiles

- under development
- TCN Gateway
- **CAN**open
- Brake Control
- Door Control WD-409
- Air Condition
- Traction
- other(s)

CiA

© **CiA** ◆ Am Weichselgarten 26 ◆ D-91058 Erlangen ◆ headquarters@can-cia.de

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			6010h	
	1h	door opened		Dh	step enabled
	2h	door closed		Eh	step locked
	3h	door enabled		Fh	ramp opened
	4h	door locked		10h	ramp closed
	5h	automatic close enabled		11h	ramp enabled
	6h	emergency disabled		12h	ramp locked
	7h	stop request activated		13h	not used
	8h	door re-opened		14h	not used
	9h	door forced closed		15h	not used
	Ah	door opposite side enabled		16h	not used
	Bh	step opened		17h	not used
	Ch	step closed		18h	not used

CiA

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
              2h close door
              3h enable door
              4h lock door
              5h enable automatic close
              6h disable emergency
              7h activate stop request
              8h reopen door
              9h close door forced
              Ah enable door opposite side
              Bh open step
              Ch close step

6010h   Dh enable step
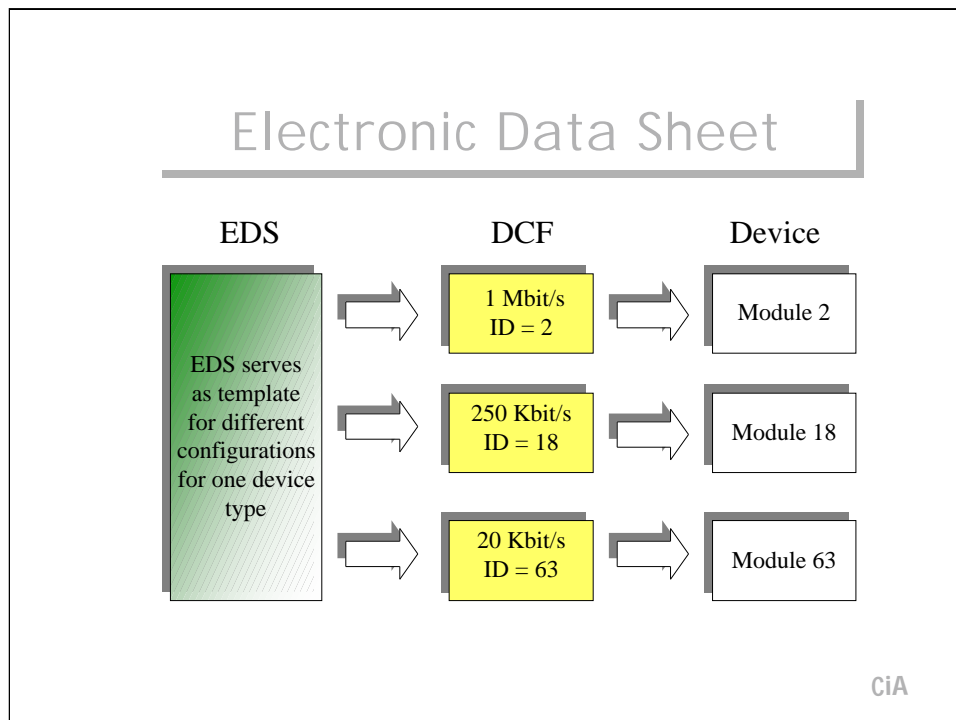   Eh lock step
              Fh open ramp
            10h close ramp
            11h enable ramp
            12h lock ramp
            13h not used
            14h not used
            15h not used
            16h not used
            17h not used
            18h not used

CiA

CANopen Vehicle Profiles

CANopen

Diesel ECU	Electronic Gear	Hydraulics	Joy Stick	other(s)

■ CiA draft standard proposal   □ CiA work items

CiA

Electronic Data Sheet

EDS — DCF — Device

EDS serves as template for different configurations for one device type

1 Mbit/s ID = 2 → Module 2

250 Kbit/s ID = 18 → Module 18

20 Kbit/s ID = 63 → Module 63

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:
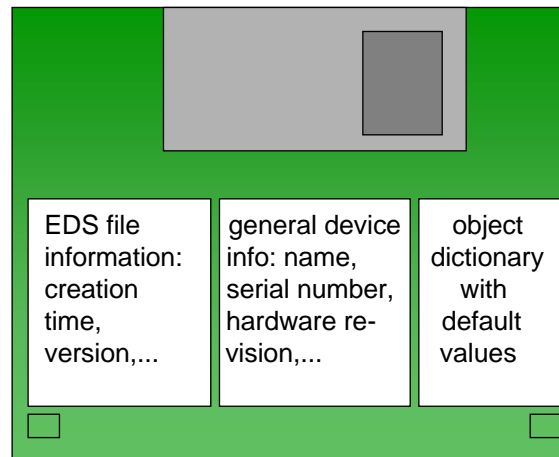
• configuration of CANopen devices,

• 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.

Electronic Data Sheet

| EDS file information: creation time, version,... | general device info: name, serial number, hardware re-vision,... | object dictionary with default values |

CiA

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,
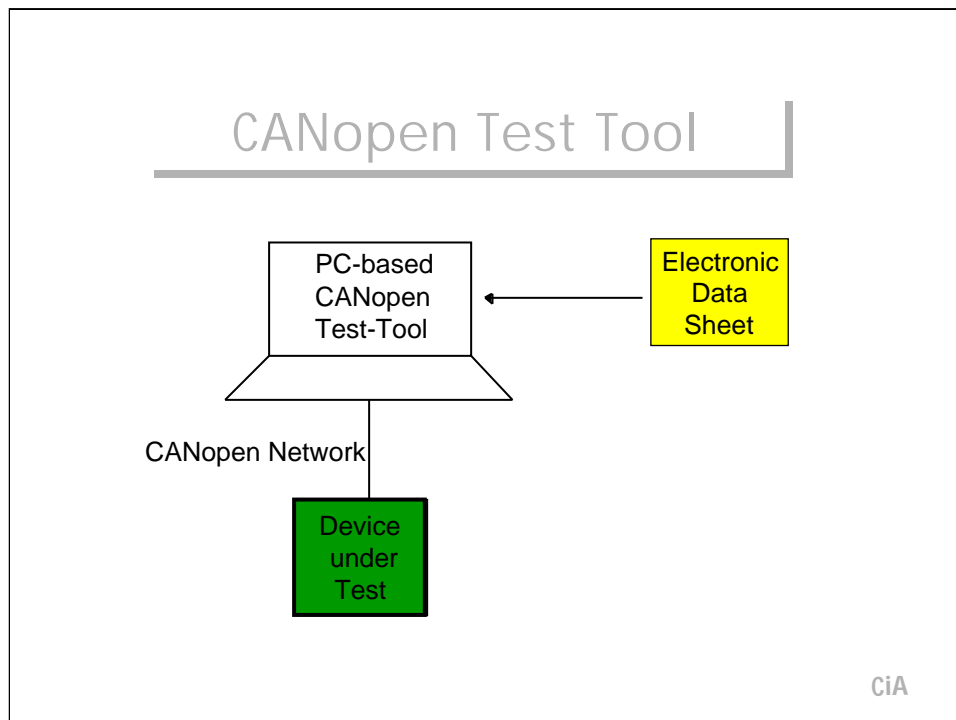
• object dictionary with default values.

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 e 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.

CANopen Test Tool

PC-based CANopen Test-Tool

Electronic Data Sheet

CANopen Network

Device under Test

CiA

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.