

CANopen

training

Datalink Engineering
CAN and CANopen experts

© Ulrik Hagström 2010

ulrik.hagstrom@datalink.se



```
inOpenStatus ret = CI  
; flags = this->canF;  
; ccs = (flags >> 5)  
; (ccs == 1)  
  
*s = (flags >> 2) &  
*e = (flags >> 1) &  
*a = (flags >> 0) &  
ret = CANOPEN_OK;
```

lee

```
inOpenStatus ret = CI  
; flags = this->canF;  
; ccs = (flags >> 5)  
; (ccs == 1)  
  
*s = (flags >> 2) &  
*e = (flags >> 1) &  
*a = (flags >> 0) &  
ret = CANOPEN_OK;
```

lee



CANopen-kurs för 1-8 personer *hos er endast*

13.500:-

exkl. moms och resekostnader!

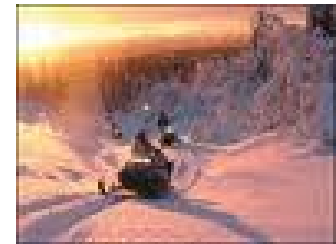
Erbjudandet gäller Sverige, Norge, Finland och Danmark.

Vi reserverar oss för prisändringar - kontakta sales@datalink.se för offert idag!

OBS! Priset inkluderar ingen aktivitet men vi tipsar gärna om events i Göteborgsområdet.

Konferensresa? & CANopen-kurs (2 x 3h) för 1-8 personer (*1-2 helgdagar*)

endast:



9.995:-

exkl. moms, lokal & resekostnader!

**Erbjudandet gäller Sverige, Norge, Finland och Danmark.
Vi reserverar oss för prisändringar - kontakta sales@datalink.se för offert idag!**



CANopen

<http://www.canopen.nu>

Mål och förutsättningar

- **Helhetsförståelse**
 - CANopen anses komplicerat till en början!
- **”Förstå ett produktblad” (buzz-words)**
 - Förstå vad CANopen noden kan och hur man kan använda den.
- **Dialog ger bästa resultat**
 - Finns inga dumma frågor!
- **Laboration (dag 2)**
 - Vi repeterar mycket av det vi tittat på. Fördjupning i CAN.
- **Ta nästa steg ?**
 - Kunna söka djupare detaljkunskaper och veta var de finns.

Agenda

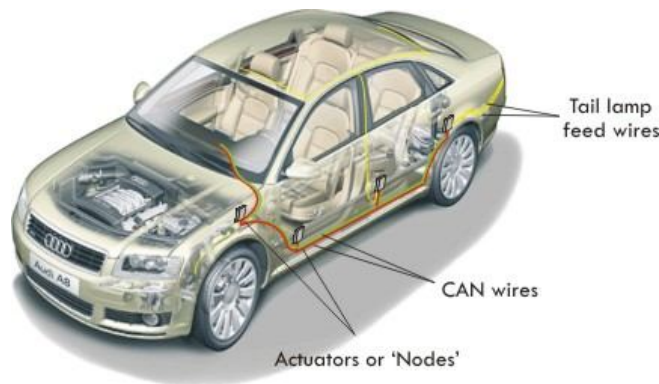
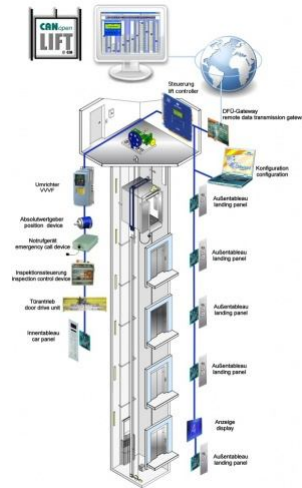
- **'Basic' CAN (1)**
 - History, application examples, CAN frame, priority, characteristics.
- **CANopen (1)**
 - History, keywords, communication layer...
- **CANopen design (1)**
 - Tools, files.
- **CANopen lab (2)**
 - Work with BK5120, basic configuration, NMT, repetition.
- **Advanced CAN (2)**
 - Signal levels, calculations, error handling in detail, registers.
- **Discussion (2)**

Part 1: Basic CAN



Industrial automation
(noise-critical env.)

Building automation
(designed for control)



Automotive (low cost, reliable, volumes)



Also suitable
for small networks

CAN milestones

BOSCH



1986 - Robert Bosch GmbH
requested by Mercedes.



1987 - The first CAN silicon
fabricated in by Intel.

1988 - CAN available for
everybody.

1991 - Mercedes S

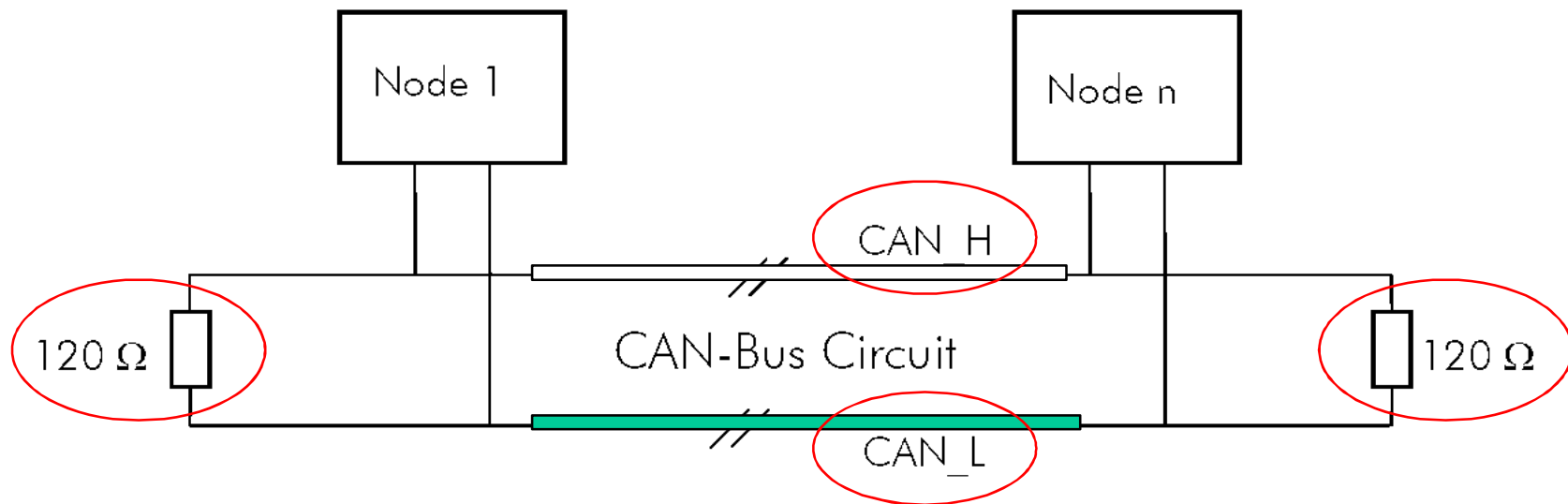


1993 - ISO 11898
specification.

1998 - Volvo S80



All CAN networks



Additional wires: 24 V + ground + sheild.

(most cases) Diffetential = Twinned CAN HI / CAN LO for best results!

Important numbers

- ~ Max 110 nodes on one *physical* network.
- 1 Mbps \Leftrightarrow 40 m.
5 Kbps \Leftrightarrow 10 000 meters.
- 1 bit error each 0.7 s, 500 kbit/s, 8h / day, 365 days / year statistical average: 1 undetected error in 1000 years (24h: 333 years)

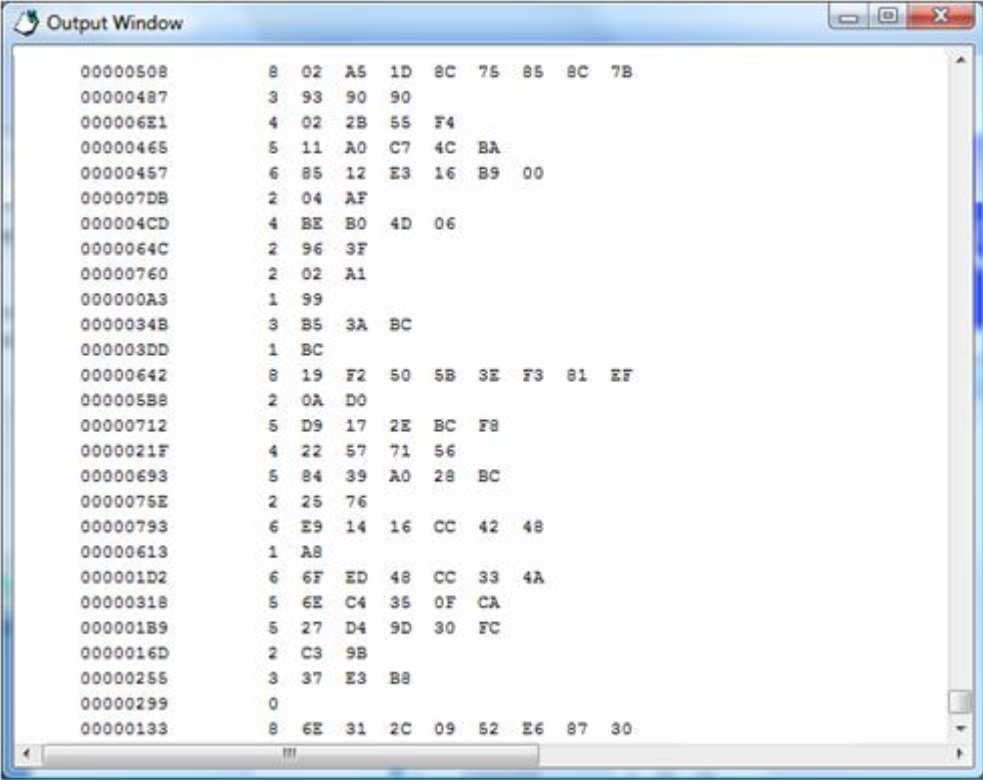
CAN offer

- Transmit/Receive a message
- Error handling
- Collision resolution
(CSMA/CR with priority)

Transmit / Receive

- CAN message contains:
 - identifier (also implements priority)
 - Data (0-8 bytes)
 - CRC checksum and other error protection data fields.
 - Needs to be interpreted by higher layer protocol (HLP).
- Multi-master capability
 - Any CAN node may send a message if bus is idle using non destructive bus arbitration.
- Network wide data consistency
 - All receiving nodes decide if they like to accept the message *but no target receive guarantees to transmitter.*

Transmit / Receive



```
Output Window
00000508      8 02 A5 1D 8C 75 85 8C 7B
00000487      3 93 90 90
000006E1      4 02 2B 55 F4
00000465      5 11 A0 C7 4C BA
00000457      6 85 12 E3 16 B9 00
000007DB      2 04 AF
000004CD      4 BE B0 4D 06
0000064C      2 96 3F
00000760      2 02 A1
000000A3      1 99
0000034B      3 B5 3A BC
000003DD      1 BC
00000642      8 19 F2 50 5B 3E F3 81 EF
000005B8      2 0A D0
00000712      5 D9 17 2E BC F8
0000021F      4 22 57 71 56
00000693      5 84 39 A0 28 BC
0000075E      2 25 76
00000793      6 E9 14 16 CC 42 48
00000613      1 A8
000001D2      6 6F ED 48 CC 33 4A
00000318      5 6E C4 35 0F CA
000001B9      5 27 D4 9D 30 FC
0000016D      2 C3 9B
00000255      3 37 E3 B8
00000299      0
00000133      8 6E 31 2C 09 52 E6 87 30
```

Error handling

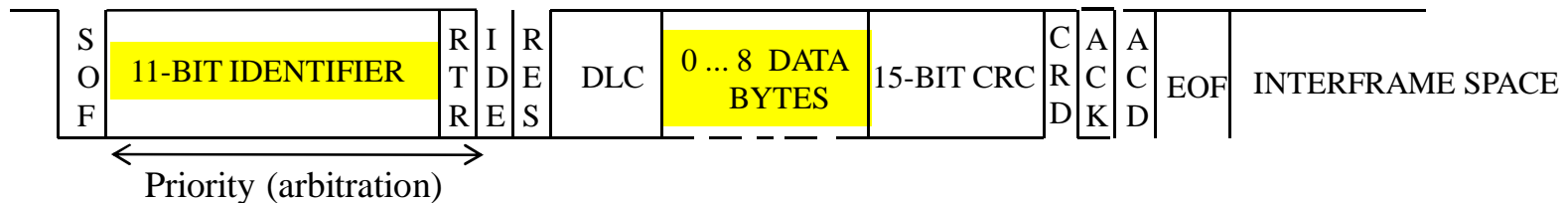
- Error checks are done **by all nodes** (transmitter do bit monitoring, receiver verifies CRC, form bit and more...)
- A CAN message is accepted by **all nodes or no node** (network wide consistency).
- Automatic retransmission on error.

Collision resolution

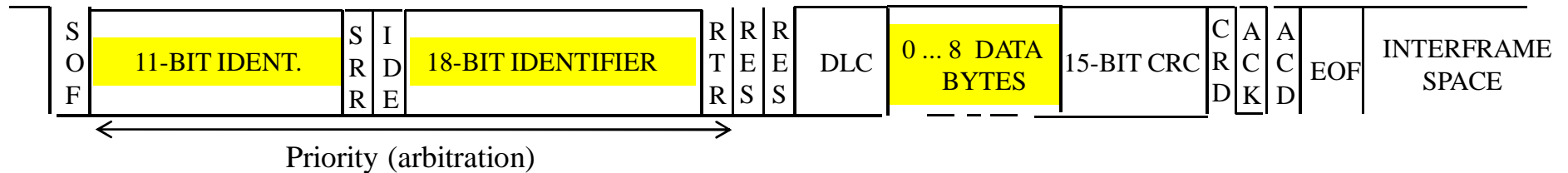
- Collisions never happens because:
CSMA/CR = Carrier Sense Multiple Access
Collision Resolution
- Collision Resolution is handled using priorities (“non destructive arbitration”).

CAN frame

Standard CAN frame (CAN – V.2.0A) (11 bit identifier)



Extended CAN frame (CAN – V.2.0B) (29 bit identifier)



Number of data bytes	0	1	2	3	4	5	6	7	8
Minimum message length	44	52	60	68	76	84	92	100	108
Maximum message length	51	60	70	80	89	99	108	118	128
Bitstuffing	(7)	(8)	(10)	(12)	(13)	(15)	(16)	(18)	(20)

Non destructive arbitration



- t0 Both “Node A” and “Node B” consider bus idle.
- t1 Both nodes start transmit “SOF” (Start of frame)
- t2 “Node A” transmits dominant bit and “Node B” recessive, and “Node A” wins the arbitration.

2010-11-20 (arbitration field = identifier + rtr-bit)

Copyright Ulrik Hagström 2010

Part 2: CANopen agenda

- Short history.
- Communication model, COBID addressing.
- Object Dictionary (OD).
- Service Data Object (SDO).
- Process Data Object (PDO).
- Error Control Protocol, Emergency Protocol (EMCY).

- Device Profiles (CANopen “plug and play”).
- Design flow, EDS, DCF, Configuration Management.
- Multiplex PDO, Time Stamp.

What is CANopen ?

- Higher level protocol (HLP) based on CAN. Ethernet is on it's way!
- Developed by CiA (CAN in Automation, can-cia.org, non-profit, 500 members).
- Considered leading standard in Europe (CAN based field bus (DeviceNet (ODVA) in USA (“a closed-club”))

1994 CAL/Philips Medical)
adopted by CiA.
1995 CANopen DS-301 v.2.0
2006 CANopen DS-301 v.4.1

cia
CANopen



Advantages using CANopen

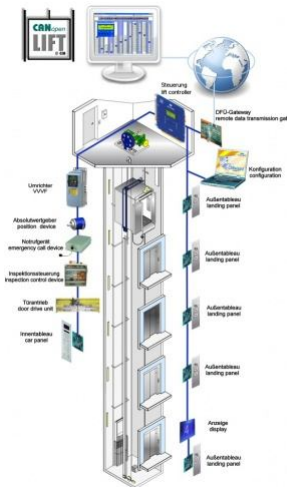
→ CANopen unburdens dealing with *CAN-specific details*.

→ Standardized highly *flexible* configuration.

→ Off-the-shelf devices, tools, and protocol stacks at reasonable prices.

→ CANopen device profiles enable "*plug and play*".

CiA 417: CANopen lift



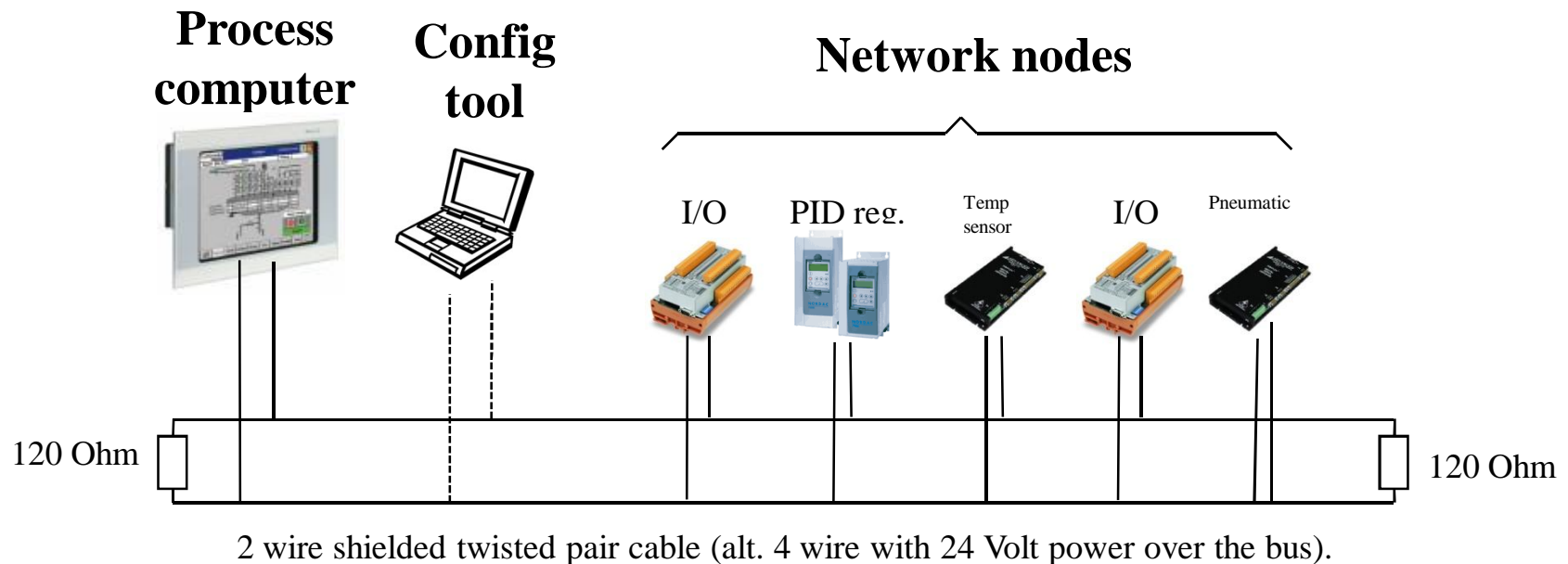
CiA 416: Building door (lock, open, fire, key cylinder, etc.)

Device Profiles

CiA 401	Generic I/O Modules
CiA 402	Drives and Motion Control
CiA 404	Measuring devices and Closed Loop Controllers
CiA 405	IEC 61131-3 Programmable Devices
CiA 406	Rotating and Linear Encoders
CiA 408	Hydraulic Drives and Proportional Valves
CiA 410	Inclinometers
CiA 412	Medical Devices
CiA 413	Truck Gateways
CiA 414	Yarn Feeding Units (Weaving Machines)
CiA 415	Road Construction Machinery
CiA 416	Building Door Control
CiA 417	Lift Control Systems
CiA 418	Battery Modules
CiA 419	Battery Chargers
CiA 420	Extruder Downstream Devices
CiA 422	Municipal Vehicles – CleANopen
CiA 423	Railway Diesel Control Systems
CiA 424	Rail Vehicle Door Control Systems
CiA 425	Medical Diagnostic Add-on Modules
CiA 445	RFID Devices

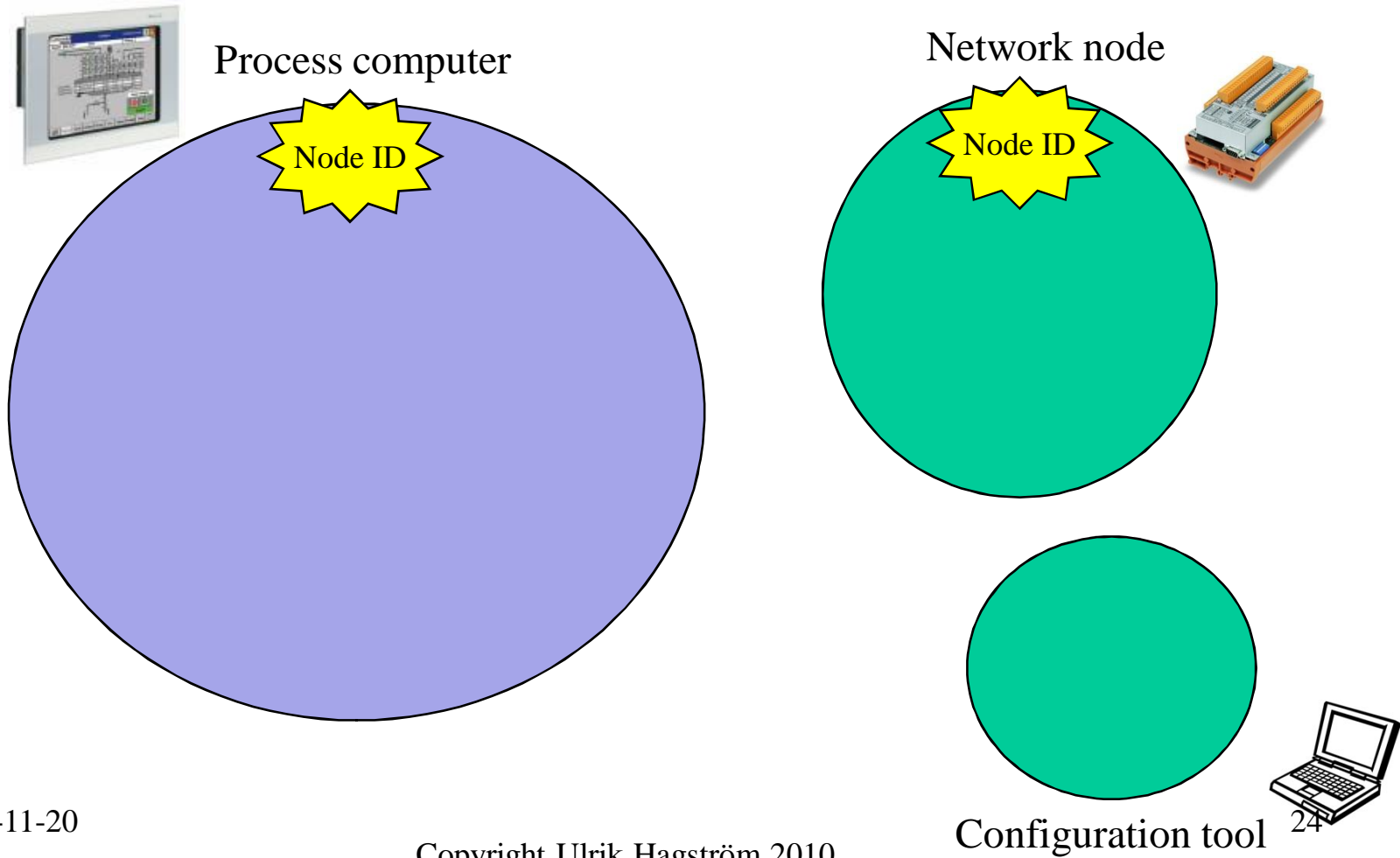
- **DS-301** **Communication profile, “basic parts of CANopen”.**
- DS-302 Framework for programmable CANopen devices (boot up, configuration manager).
- DS-306 EDS (Electronic data sheet, template), DCF (Device configuration file, values)
- DS-4xx Device profiles (“plug and play I/O, servo etc, HMI”)

Example CANopen network



- All nodes have a **node id** value (1-127, 0 is broadcast address)
- All nodes can be accessed after boot by default connection set (node-id relation).

Node functionality

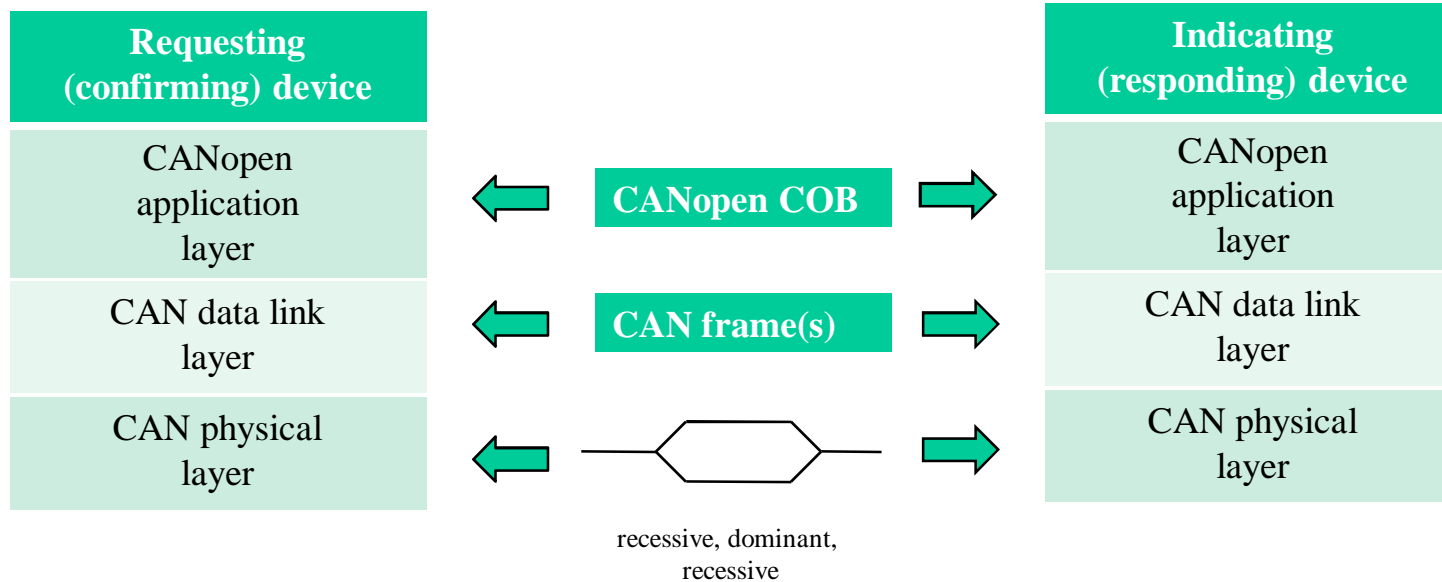


2010-11-20

Copyright Ulrik Hagström 2010

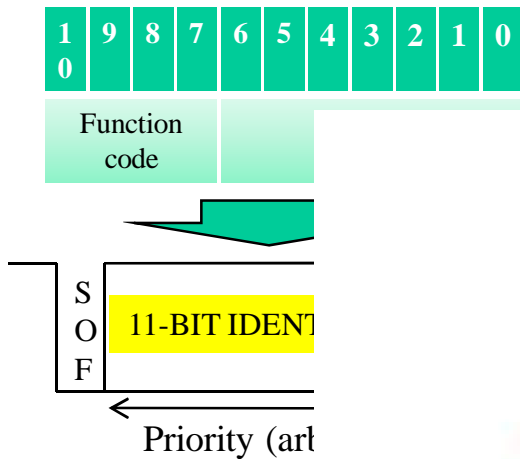
Configuration tool 24

CANopen OSI model



COB = Communication object

COB-ID



COB-ID in *configuration area (OD)* is 32 bits
(upper bits 31-29 are status flags, lower bits equal CAN bus (11 bits)).

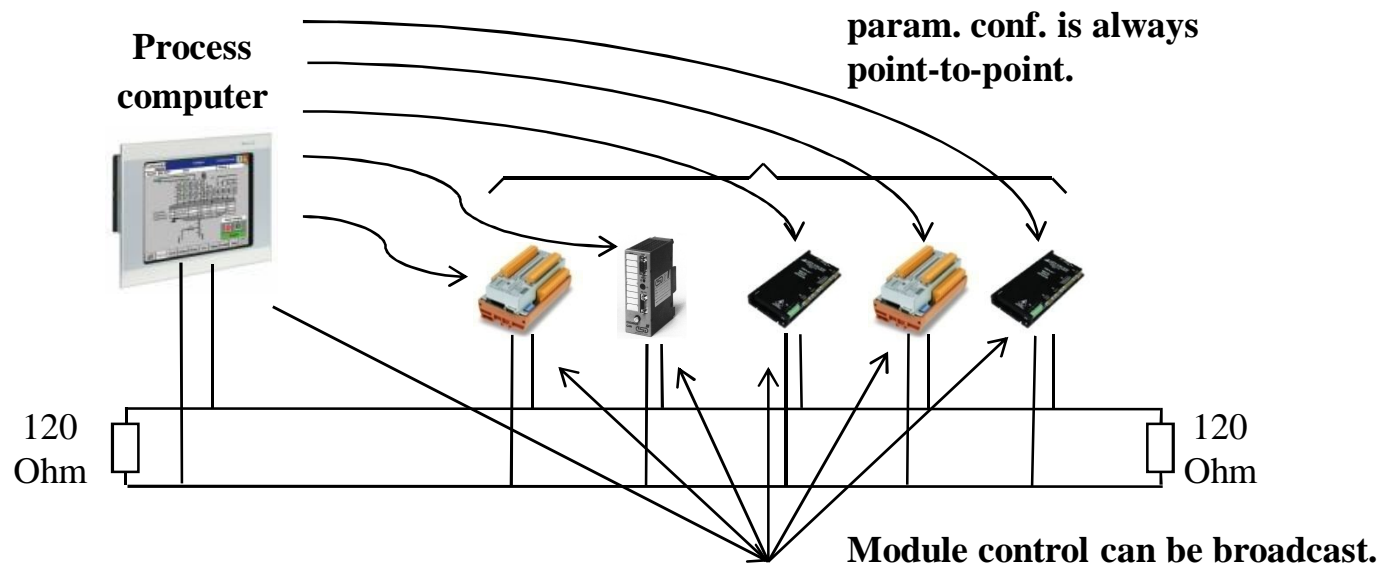


COB-ID (hex)	Priority	Reserved
0x0	0	
0x1	0	
0x101-0x180	0	
0x581-0x5ff	1	
0x601-0x67f	1	
0x6e0		Reserved
0x701-0x77f	1110xxxxxxx	NMT (node error control)
0x780-0x7ff		reserved



Network initialization process

1. All nodes initialize and enter pre-operational state after power on.
2. Process computer configures the nodes (parameter configuration).
3. Process computer sets nodes in operational state (module control).



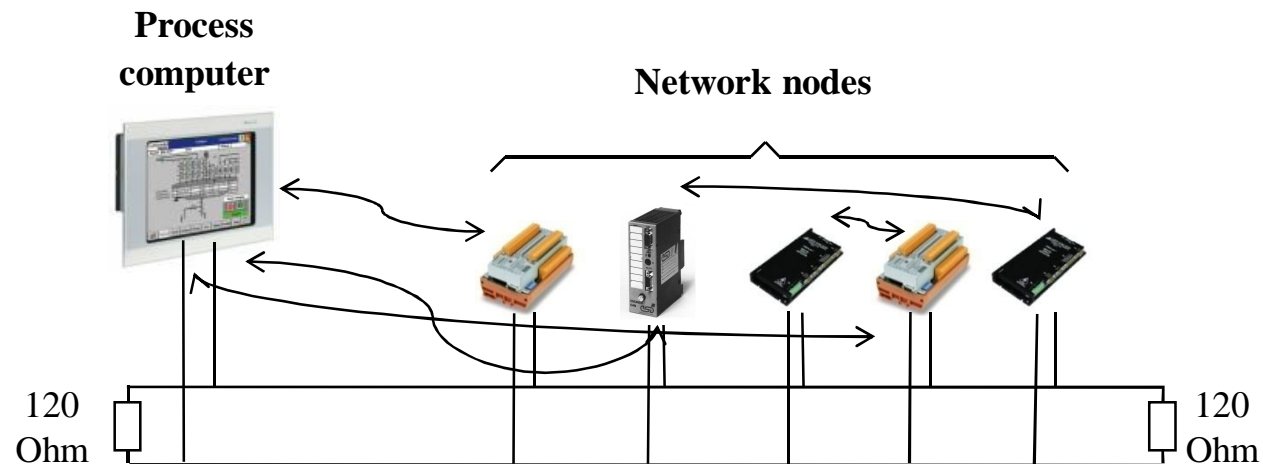
Power on or Hardware Reset



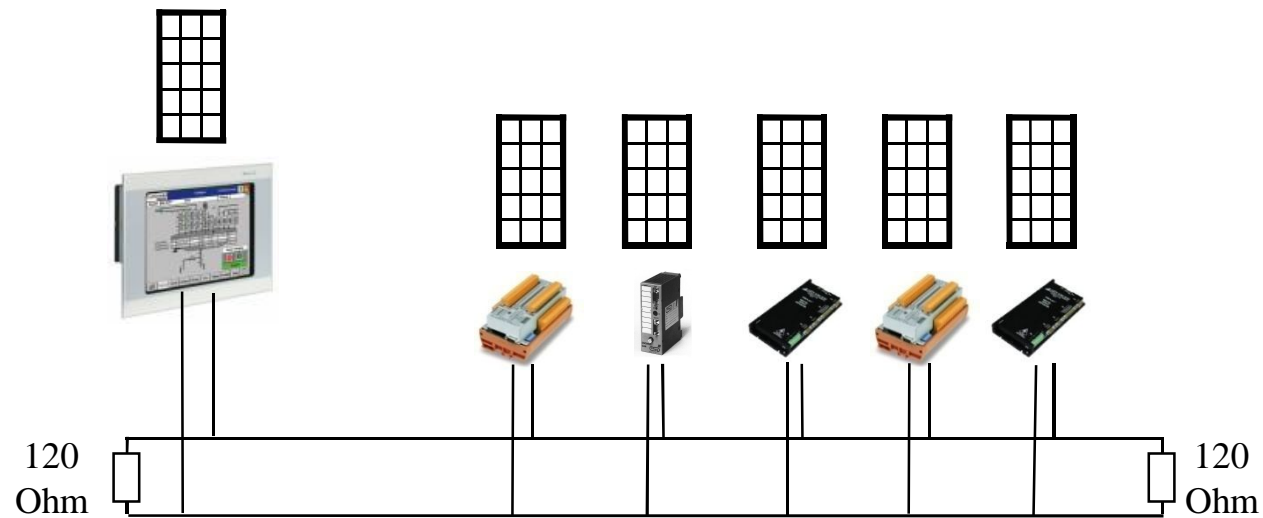
State	Description
Initialization	Initialization at power on the CANopen slave (with minimum boot-up capability) performs an initialization sequence and enters automatically into the Pre-Operational state .
Pre-operational	Parameter configuration. Node guarding and respond to node-guarding protocol.
Stopped	Node is disabled, no communication except node guard response.
Operational	Have it's process data channels active. Have parameter conf. channels active. Send emergency messages on an error event .

Running network

1. Process computer monitor operational state of nodes.
2. Process data can be sent from process computer to network nodes or between network nodes directly.



Object Dictionary



OD = "parameter configuration area"

Object Dictionary

- OD desc

mandatory

- EDS

behaviour.

entations)

(306)



Object Dictionary

Object Index	Sub Index	Data Type	Bit contents	Description
0x1000	0	UINT32		Device Type.
		Byte 1		Device Profile. Additional info.
		Byte 2		
		Byte 3		
		Byte 4		
0x1001		UINT8		Error Register (Read error type on node).
0x1002		UINT32		Manufacturer status register.
0x1003		UINT32		Predefined error field.
0x1004				Reserved (for number of PDOs?)
0x1005		UINT32		COBID SYNC
0x1006		UINT32		Communication cycle period.
...				
0x1008		STRING		Device name.
0x1009		STRING		Hardware version.
0x100A		STRING		Software version.
0x100B				Reserved (for setting new node ID?)
0x100C		UINT16		Guard time
2010-11-20 0x100D		UINT8		Copyright Ulrik Hagström 2010. Lifetime factor

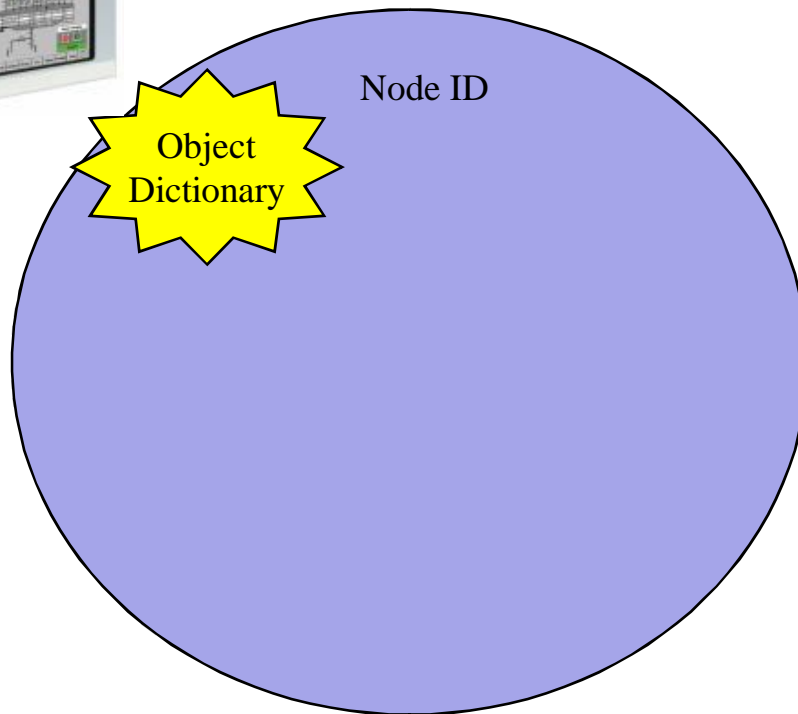
Supported data types in OD

Basic Data Type	Supported types / values
BOOLEAN	{true, false}
VOID	Bit sequence of n bits.
UNSIGNED INTERGER	{ 8, 16, 24, 32, 40, 48, 56, 64 } bits
INTEGER	{ 8, 16, 24, 32, 40, 48, 56, 64 } bits
SIGNED FLOATING POINT	{8} , {23} bits
OCTET STRING	Array of UINT8
VISIBLE STRING	Array of UINT8
UNICODE STRING	Array of UINT16
TIME OF DAY	48 bits { UINT28 ms, VOID4 reserved, UINT16 days }
TIME DIFFERANCE	48 bits { UINT28 ms, VOID4 reserved, UINT16 days }

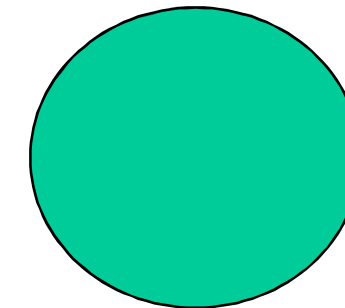
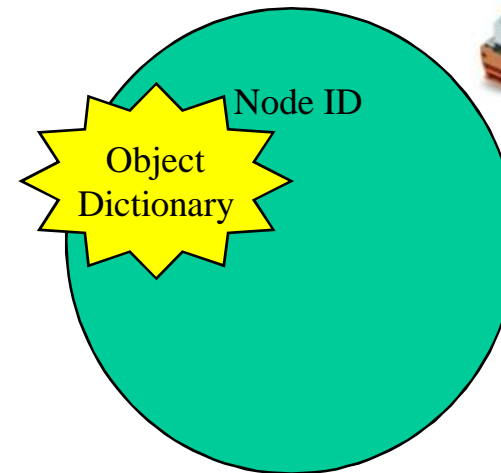
Node functionality



Process computer

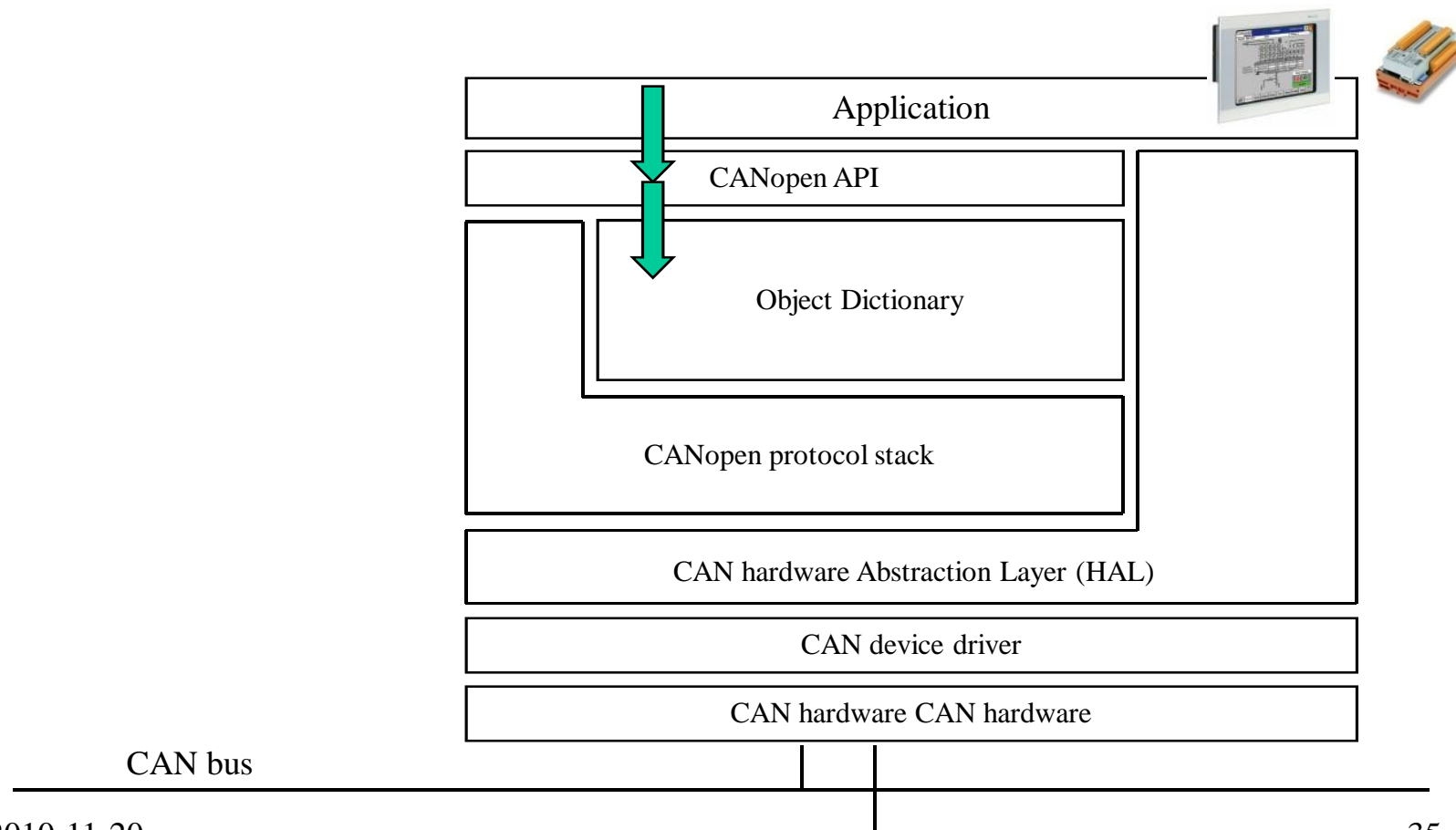


Network node

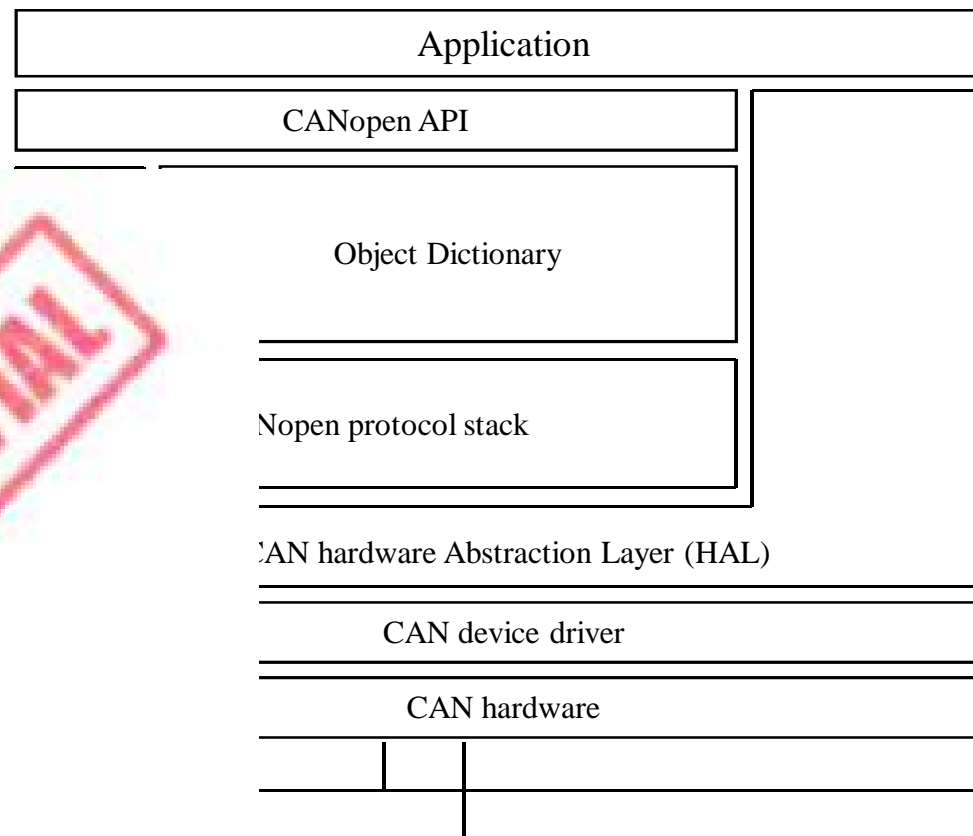


Configuration tool 34

Access to local Object Dictionary



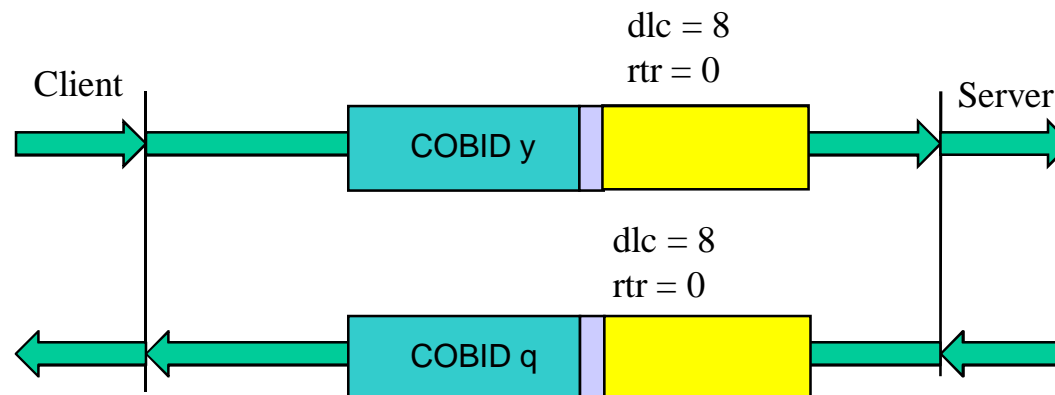
Parameter configuration via CAN-bus?



CONFIDENTIAL

Service Data Object (SDO)

- Mainly used for parameter configuration of remote node.
- Transfer protocol for parameters (also FW upgrade).



Service Data Object (SDO)



- Re...

usually

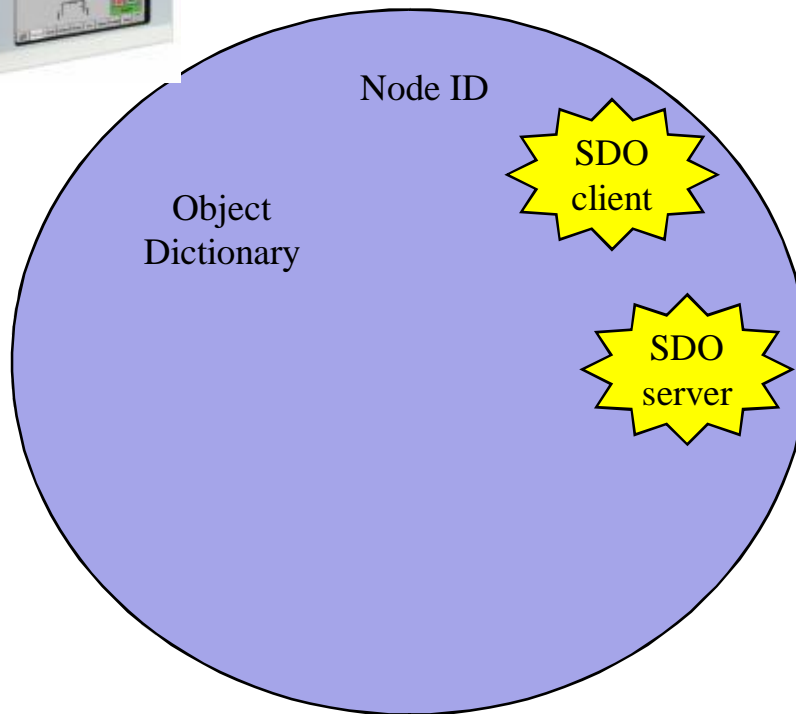
de.
es OD,
manager).

CONFIDENTIAL

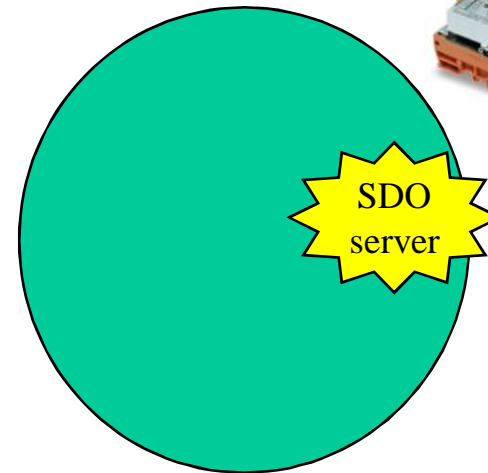
Node functionality



Process computer



Network node



Configuration tool 39

SDO transfer protocols

Expedited transfer

1 to 4 bytes

Segmented transfer (“normal transfer”)

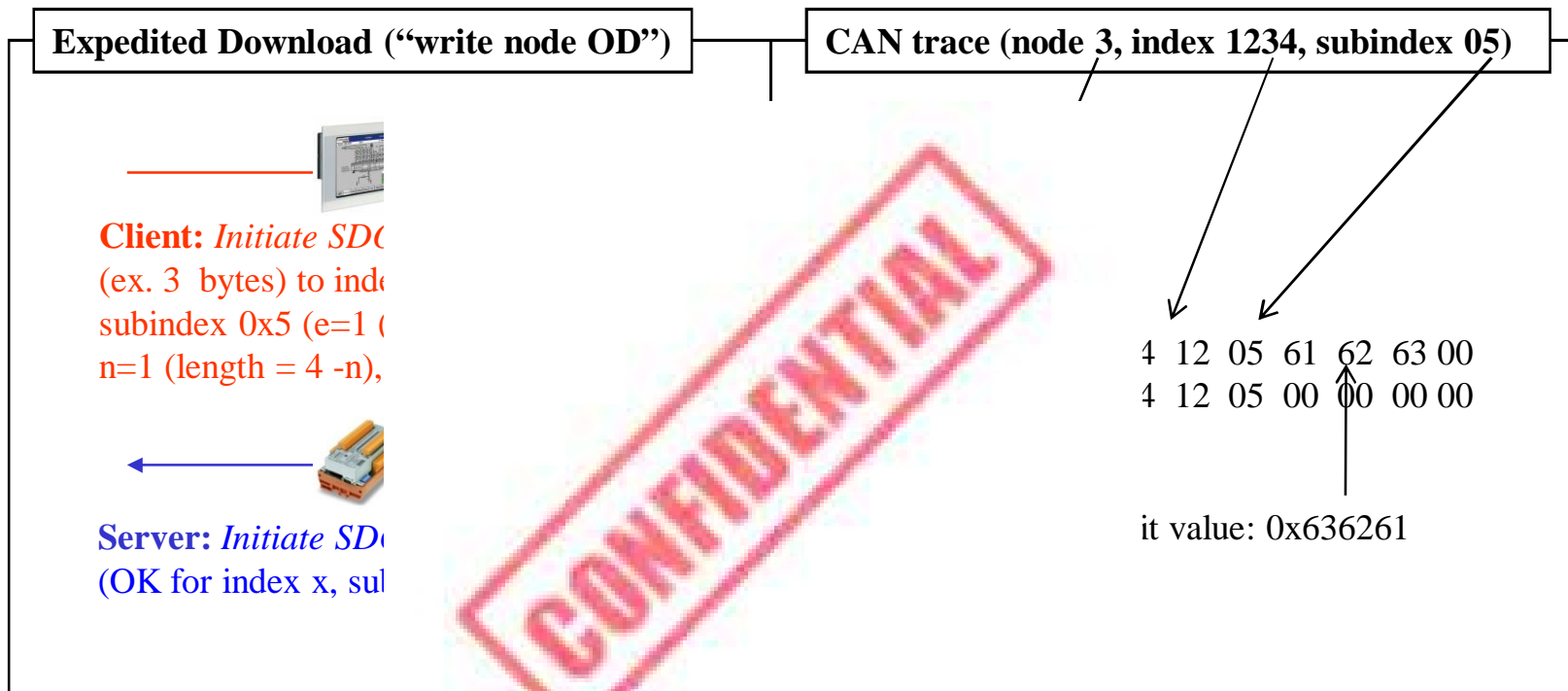
More than 4 bytes (64 bits value and string for example)

Block transfer

More bandwidth efficient than segmented transfers but “do the same thing”.

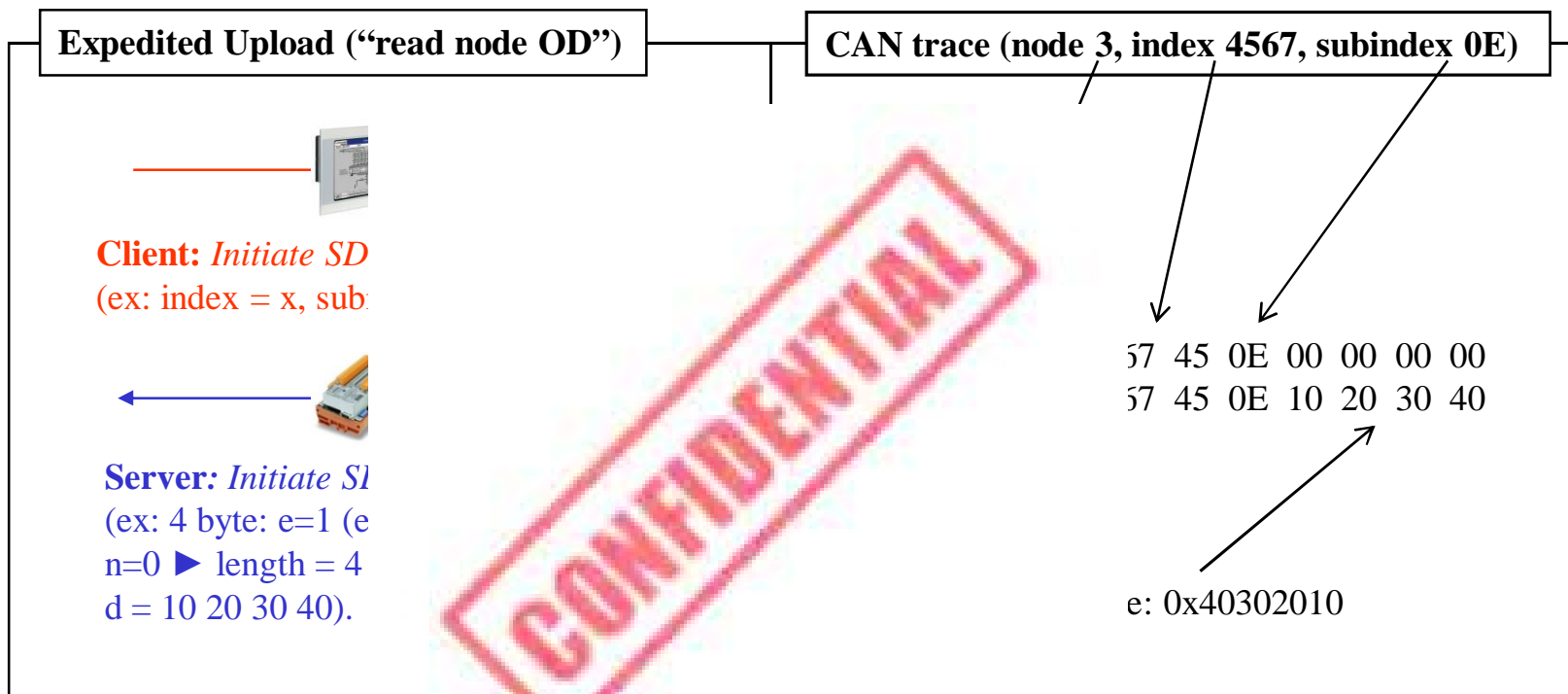
The SDO transfer protocols

(Expedited-, Segmented- or Block-transfer)



The SDO transfer protocols

(Expedited-, Segmented- or Block-transfer)



The SDO transfer protocols

(Expdited-, Segmented- or Block-transfer)

Segmented Download (“write node OD”)

CAN trace (node 3, index 1235, subindex 05)



Client: Initiate SDO download request
index 0x1235 and ;
bytes - e=0 (not ex
given), d = 9 (leng

Server: Initiate SL
(OK).



Client: Download
(t = 0 (toggle) c = (

Server: Download
response (t = 0 (tog



Client: Download
(t = 1 (toggle) c = 1
n=2).

Server: Download segment
response (t = 1 (toggle))

CONFIDENTIAL

```

35 12 05 09 00 00 00
35 12 05 00 00 00 00
31 32 33 34 35 36 37
00 00 00 00 00 00 00
38 39 00 00 00 00 00
00 00 00 00 00 00 00
    
```



The SDO transfer protocols

(Expdited-, Segmented- or Block-transfer)

Segmented Upload (“read node OD”)

CAN trace (node 3, index 0x5678, subindex 0x07)



Client: *Initiate SDO*
(ex: index = 0x5678)

←
Server: *Initiate SDO*
(ex: 5 bytes: e=0 (no
d = 5).



Client: *Upload SDO*
(ex: t = 0 (toggle)).

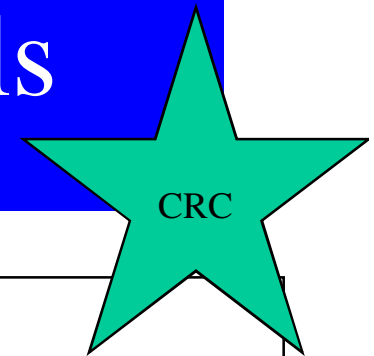
←
Server: *Download*
(t = 0 (toggle) c = 1
n=5, data = 00 01 02 03 04 00 00)

CONFIDENTIAL

```
78 56 07 00 00 00 00
78 56 07 05 00 00 00
00 00 00 00 00 00 00
00 01 02 03 04 00 00
```

The SDO transfer protocols

(Expdited-, Segmented- or Block-transfer)



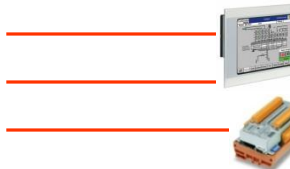
Block transfer Download (“write node OD”)

Client: *Initiate SDC request (ex: index =*



Server: *Initiate SDC response with given segments per block.*

Client: *SDO block segment (sequence 1*

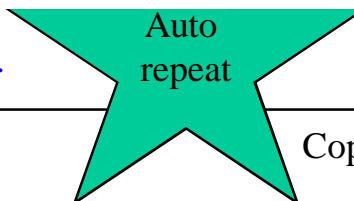


Client: *SDO block download response.*



```

2 34 12 05 3E 00 00 00
0 34 12 05 05 00 00 00
0 54 68 69 73 20 73 74
1 72 69 6E 67 20 68 61
2 73 20 62 65 65 6E 20
3 73 65 6E 74 20 76 69
4 61 20 43 41 4E 6F 70
2 04 05 00 00 00 00 00
0 65 6E 20 42 4C 4F 43
1 4B 20 54 52 41 4E 53
2 46 45 52 20 50 52 4F
3 54 4F 43 4F 4C 00 00
2 03 05 00 00 00 00 00
5 00 00 00 00 00 00 00
1 00 00 00 00 00 00 00
    
```

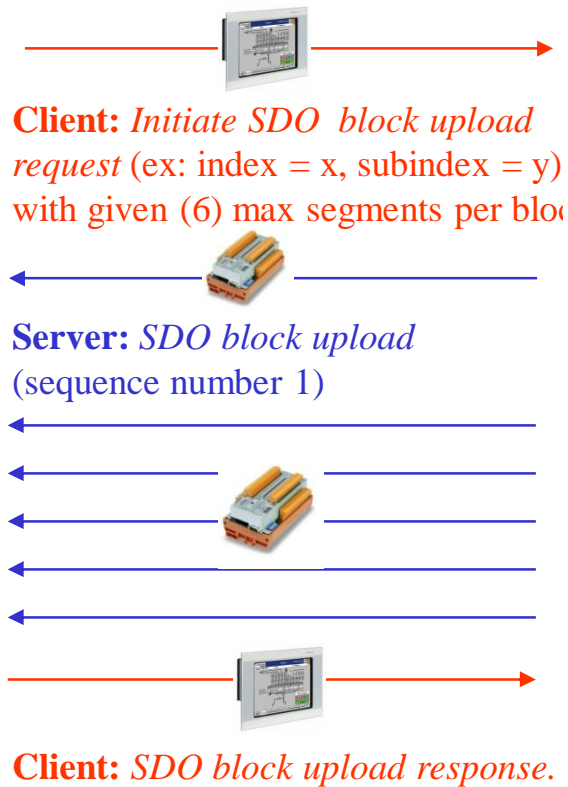


The SDO transfer protocols

(Expdited-, Segmented- or Block-transfer)

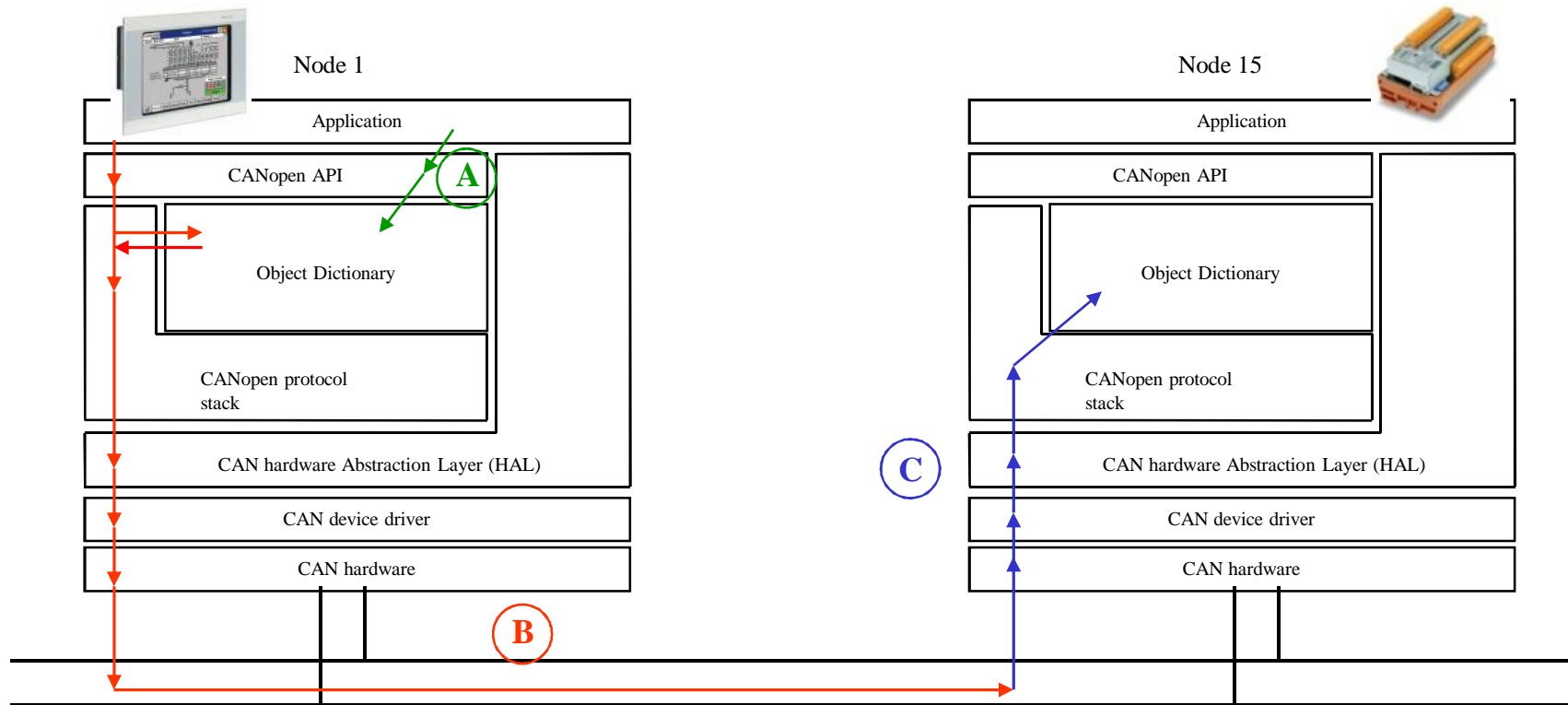
Block transfer Upload (“read node OD”)

CAN trace N/A



CONFIDENTIAL

Map Client SDO to Server SDO



- A)** Node 1 configures one of its SDO client (in this example SDO client 1) to connect to the SDO server 1 (default SDO server) on node 15.
- B)** Node 1 starts the read/write operation via the CANopen API using SDO client no.1.
- C)** Node 15's Server SDO 1 responds to the request.

Default connection set

Object	COBID
Broadcast Network Management (NMT)	0x000
Synchronization (SYNC)	0x080
Emergency (EMCY)	0x080 + NodeId
Transmit PDO 1	0x180 + NodeId
Receive PDO 1	0x200 + NodeId
Transmit PDO 2	0x280 + NodeId
Receive PDO 2	0x300 + NodeId
Transmit PDO 3	0x380 + NodeId
Receive PDO 3	0x400 + NodeId
Transmit PDO 4	0x480 + NodeId
Receive PDO 4	0x500 + NodeId
Server SDO (TX)	0x580 + NodeId
Server SDO (RX)	0x600 + NodeId
Module error control, boot-up protocol, heartbeat etc.	0x700 + NodeId

Object Dictionary

Object Index	Sub Index	Data Type	Bit contents	Description
0x1020				Verify configuration.
	0x1	UINT32		Configuration Date
	0x2	UINT32		Configuration Time
0x1028				Emergency Consumer
	0x1 – 0x7f	UINT32		Emergency Consumer COBID (1 – 127)
0x1200 – 0x127f		SERVER SDO		SERVER SDO 1 – 128
	0x1	UINT32		COBID Client to server (RX)
	0x2	UINT32		COBID Server to client (TX)
	0x3	UINT8		Node Id of the SDO client
0x1280 – 0x13ff		CLIENT SDO		CLIENT SDO 1 – 128
	0x1	UINT32		COBID Client to server (TX)
	0x2	UINT32		COBID Server to Client (RX)
	0x3	UINT8		Node Id of the SDO server

Connect masters client SDO 1 to default server on node 15.

Object Index	Sub Index	Data Type	Bit contents	Description
0x1028				Emergency Consumer
	0x1 – 0x7f			D (1 – 127)
0x1200 – 0x127f				
	0x1)
	0x2)
	0x3			
0x1280 – 0x13ff				
	0x1)
	0x2)
	0x3			



Client SDO 1 is found at object index **0x1280**.

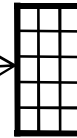
we shall configure or Client SDO 1 to TX on that COBID.

15
 default connection set
 fault SDO server was TX
0x580 + 15 and therefore

we shall configure or Client SDO 1 to RX on that COBID.

Result of SDO client configuration

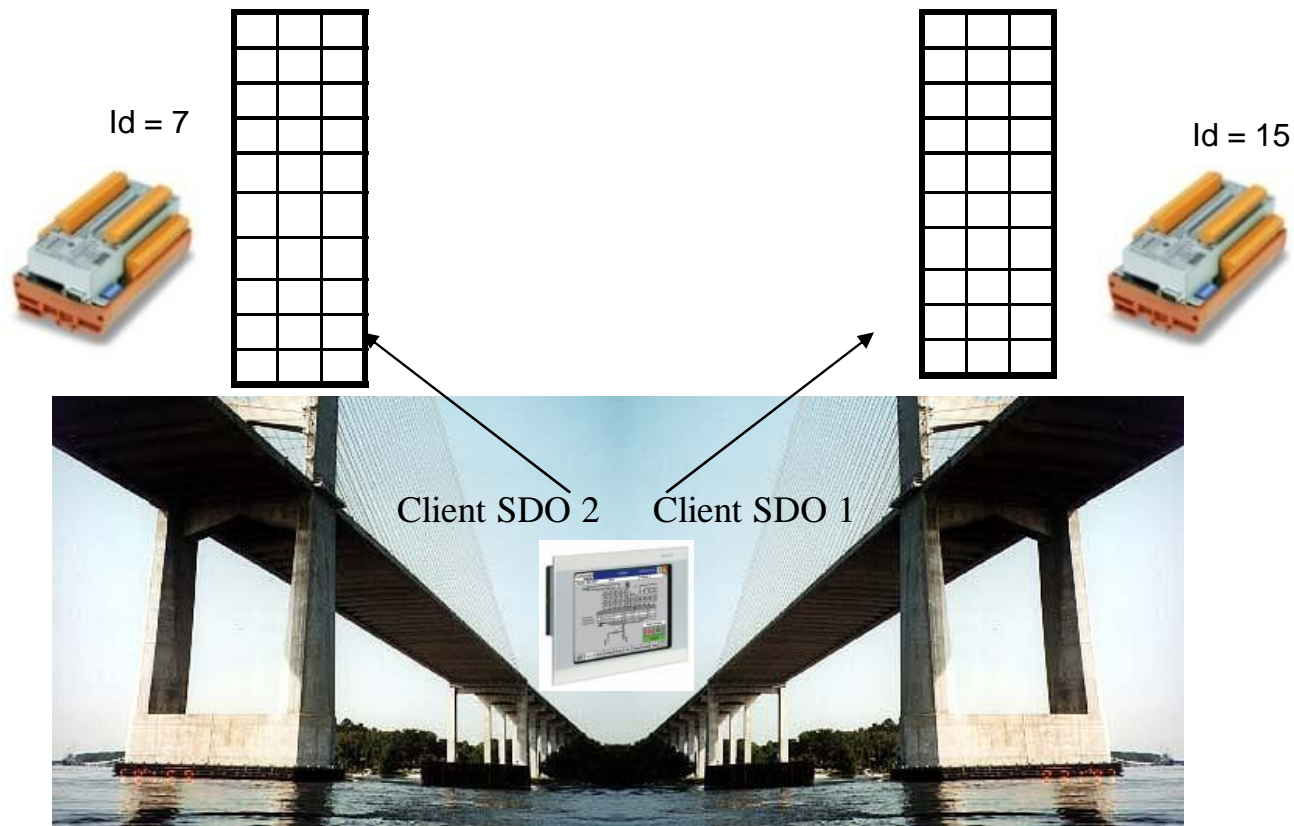
Client SDO COBID TX == Server SDO COBID RX
Client SDO COBID RX == Server SDO COBID TX



Node id 15



If more nodes and more Client SDOs are available...



2010-11-20

CANopen master can have up to 128 client SDOs

(=128 connections)

Copyright Ulrik Hagström 2010

52

Configure a SDO connection

Connect Client SDO to *connect to default SDO of remote node 15 & 7.*



Process Data Object (PDO)

- Sent in run-time to control the running process.
 - Carry the real time process data

2 types of PDOs

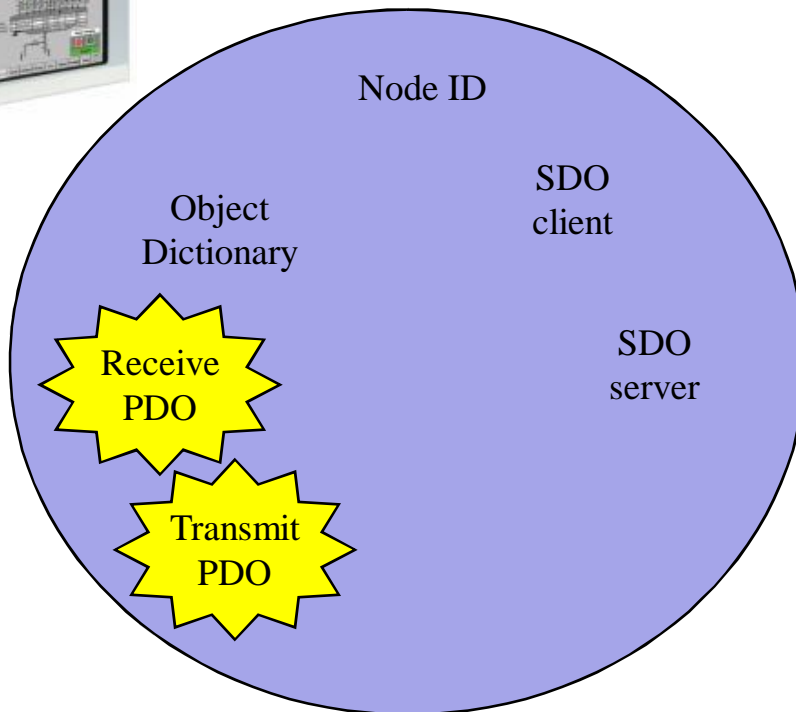
Transmit PDO

Receive PDO

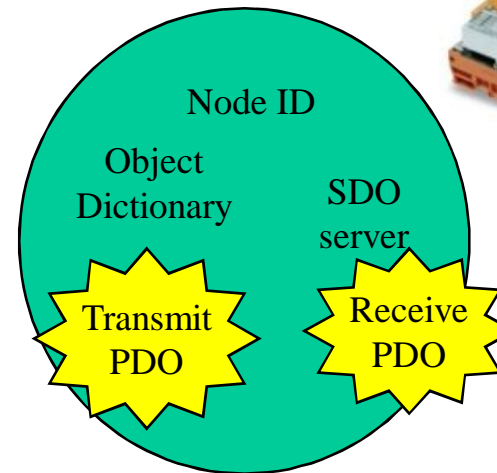
Node functionality



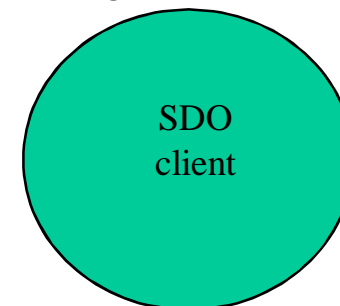
Process computer



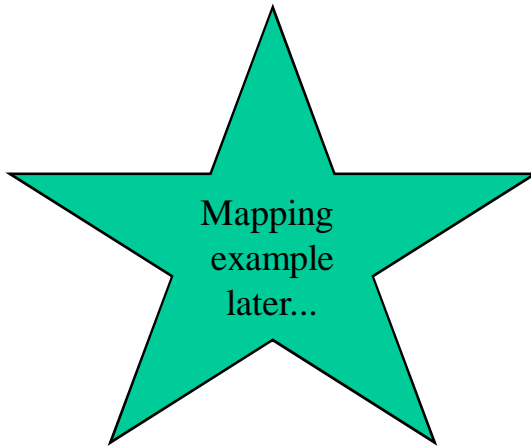
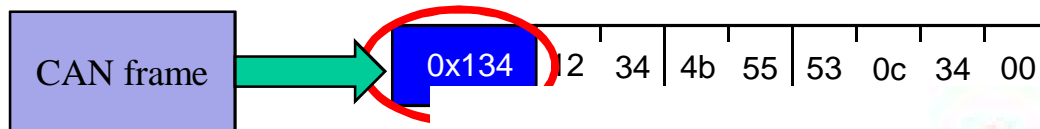
Network node



Configuration tool



PDO decoding example



2010-11-20

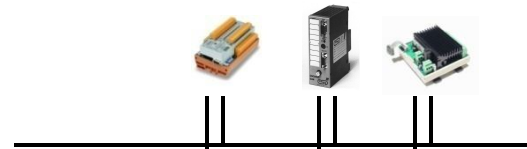
Types of PDOs

- Event driven
- Timer driven
- Remotely requested
 - Synchronized

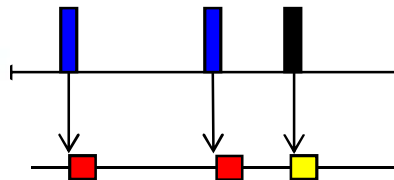
Event driven TPDO

(can cause delay problem)

CAN bus configuration



- = Events that trig to be sent
- = Events that trig to be sent
- = Events that trig to be sent



CONFIDENTIAL

ige)
d)

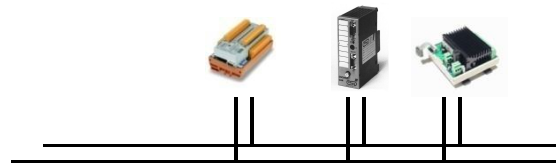









ge (Event)



Event driven TPDO with inhibit time

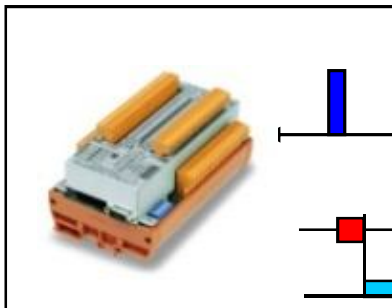
(solves the event burst problem)

CAN bus configuration

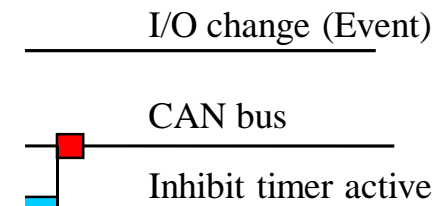


-  = Events that trigger  to
-  = Events that trigger  to
-  = Events that trigger  to
-  = Pending inhibit

Event (I/O input change)
 is luckily not delayed
 and, sent from node 

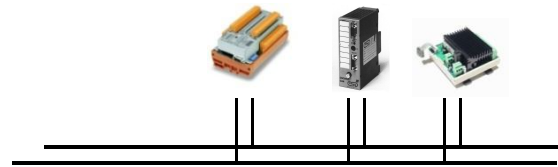





CONFIDENTIAL



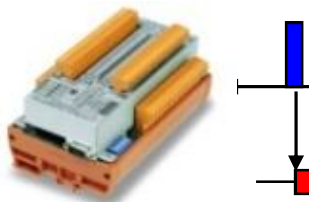
Timer driven PDO transfer

CAN bus configuration



-  = Events that trigger
-  = Events that trigger
-  = Pending Event

event (I/O input change)



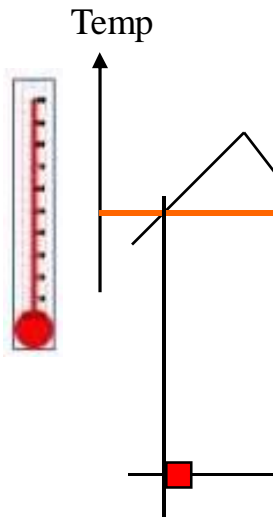
CONFIDENTIAL

I/O change (Event)

CAN bus

Event Timer pending

Remotely requested PDO



Remotely requested

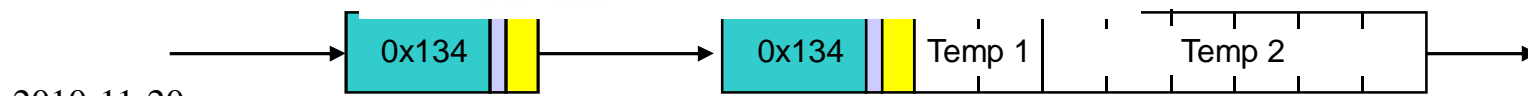


t gets requested by master.

- = Temp PDO sent from
- = Remote request P



R1



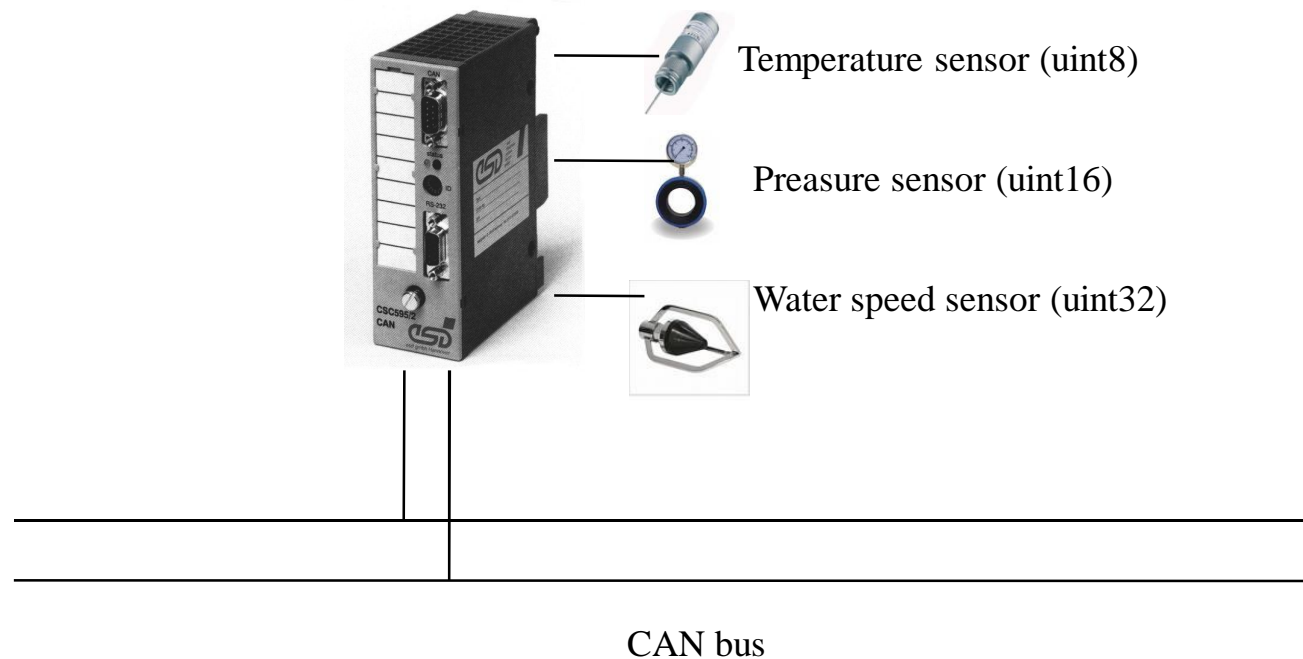
2010-11-20

Copyright Ulrik Hagström 2010

62

PDO mapping example (1/6)

CANopen network node



PDO mapping example (2/6)

Register the programming variables in the Object Dictionary of the node.

Object Index	Sub Index	Data Type	Bit contents	Description
0x2200	0	UINT8		TEMP
0x2201	1	UINT16		PREASURE SENSOR
0x2201	2	UINT32		WATER SPEED

PDO mapping example (3/6)

Object Index	Sub Index	Data Type	Bit contents	Description
0x2200	0	UINT8		TEMP
				PREASURE SENSOR A1
				WATER SPEED

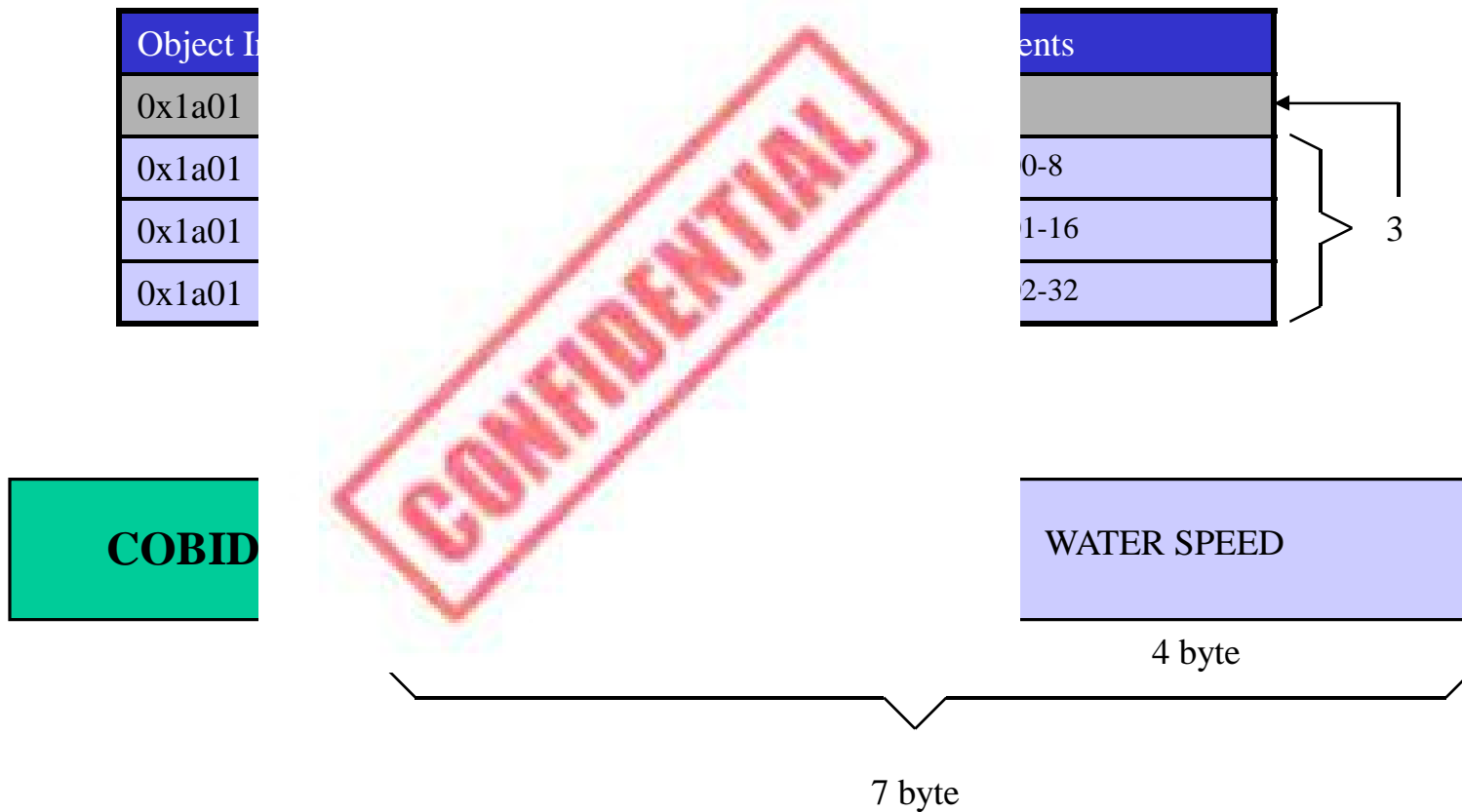
CONFIDENTIAL



Description
Transmit PDO1 mapping parameters.
Transmit PDO2 mapping parameters
Number of objects to map.
0x2200-00-08
0x2201-01-16
0x2201-02-32
Transmit PDO3 mapping parameters
Transmit PDO4 mapping parameters
Transmit PDO _n mapping parameters



PDO mapping example (4/6)



PDO mapping example (5/6)

Object Index	Sub Index	Data Type	Description
0x1600 – 0x17ff			RPDO1 - ...)
	0x1		Object to be mapped.
0x1800	0x0		Communication Parameters.
0x1801	0		Communication Parameters (SubIdx 0 == 5.)
	0x1		
	0x2		
	0x3		
	0x4		
	0x5		
0x1802			Communication Parameters.
...	0x1		Communication Parameters.

COBID	WATER SPEED
--------------	-------------

1 byte
2 byte
4 byte

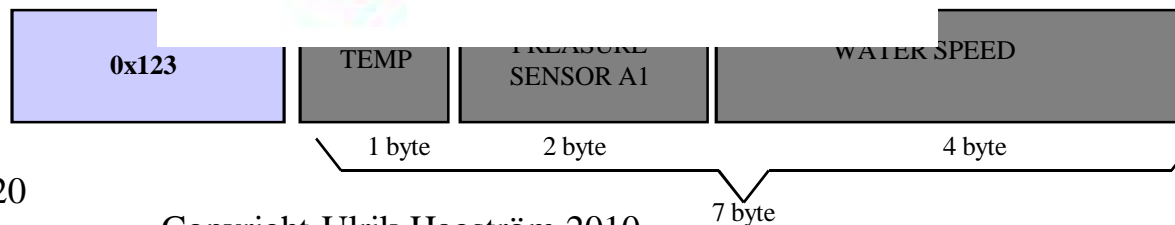
7 byte

PDO mapping example (6/6)

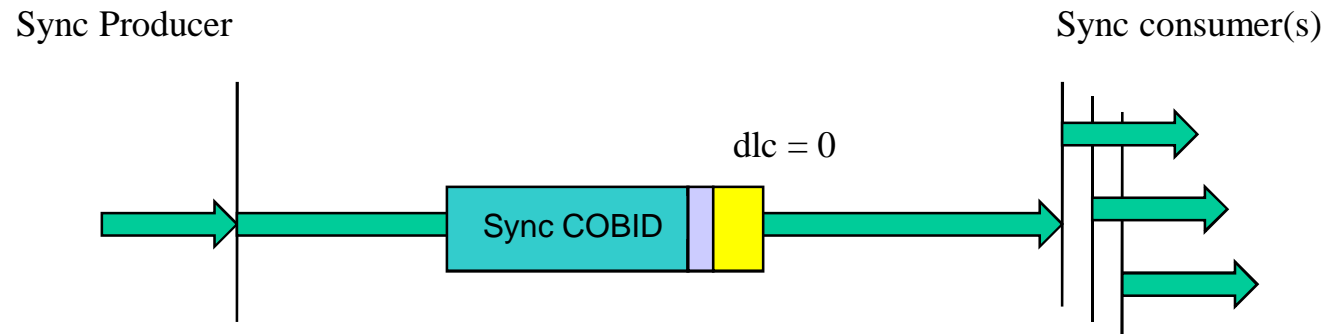
Object Index	Sub Index	Data Type	Bit contents
0x1a01		UINT8	0-8
0x1a01		UINT8	9-16
0x1a01		UINT8	17-32

} 3

Object Index	Bit contents
0x1801	0x123
0x1801	254
0x1801	100 (*10us)
0x1801	0
0x1801	1000 (*1ms)



Sync “pulse” for sync PDO

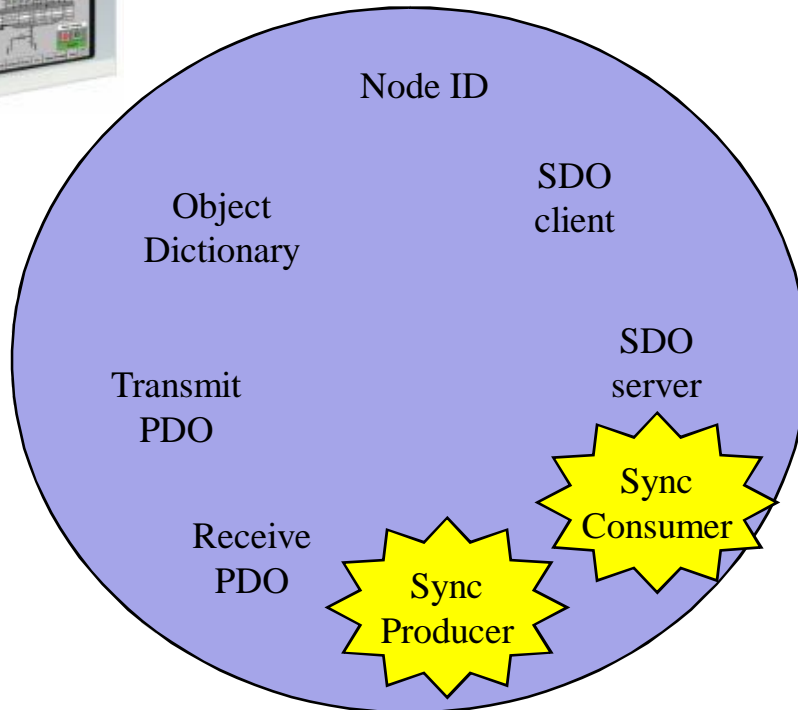


Object	Parameter
0x1005	COB-ID (Bit 30: consumer/producer flag)
0x1006	Communication cycle period

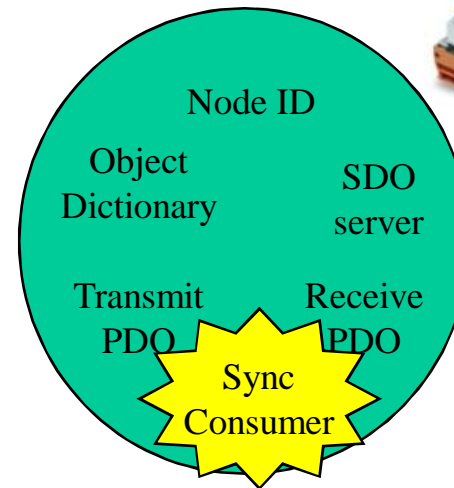
Node functionality



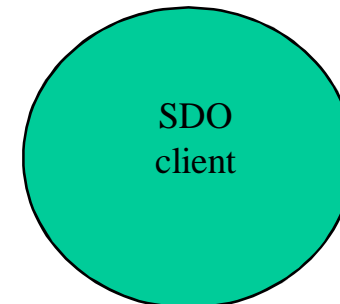
Process computer



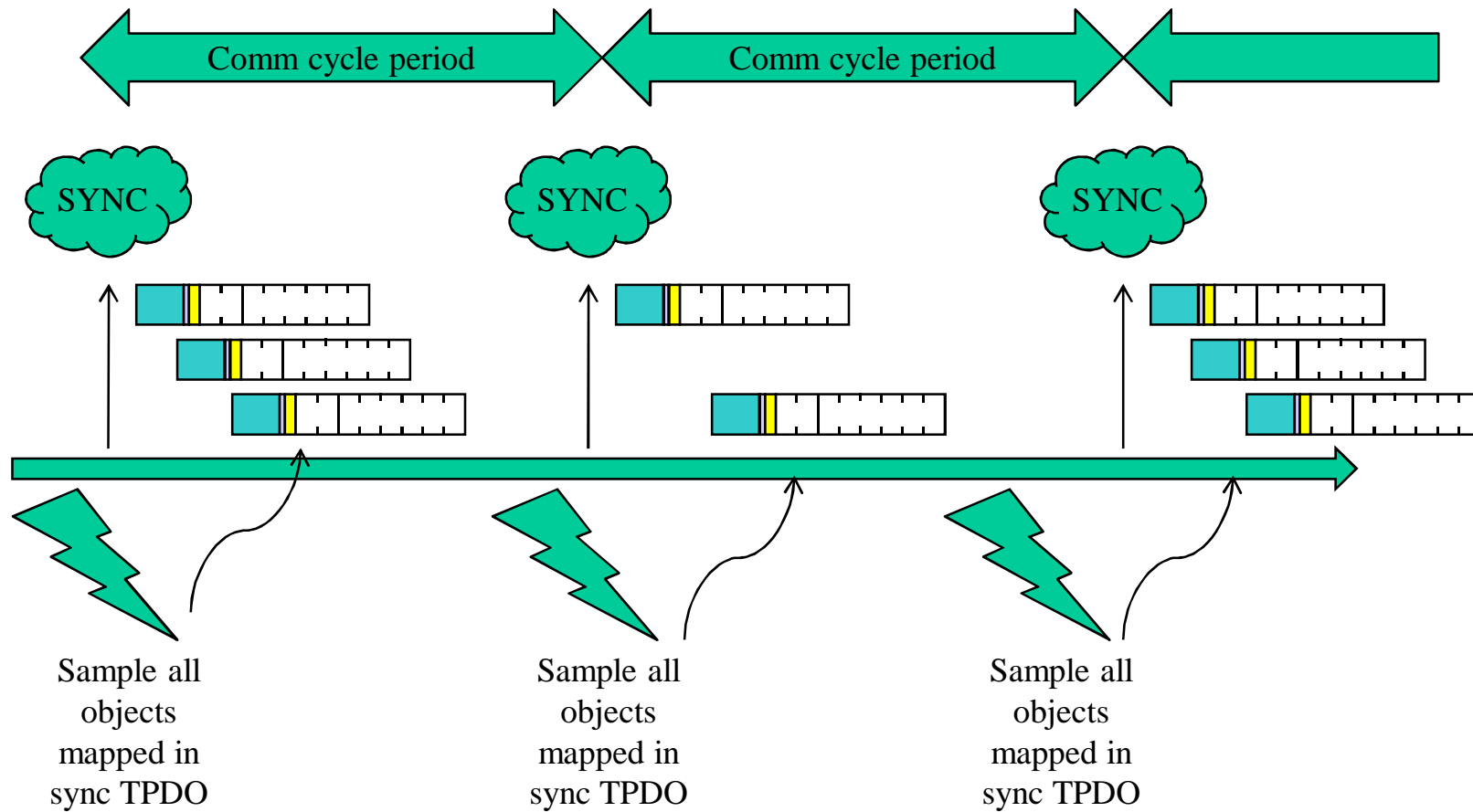
Network node



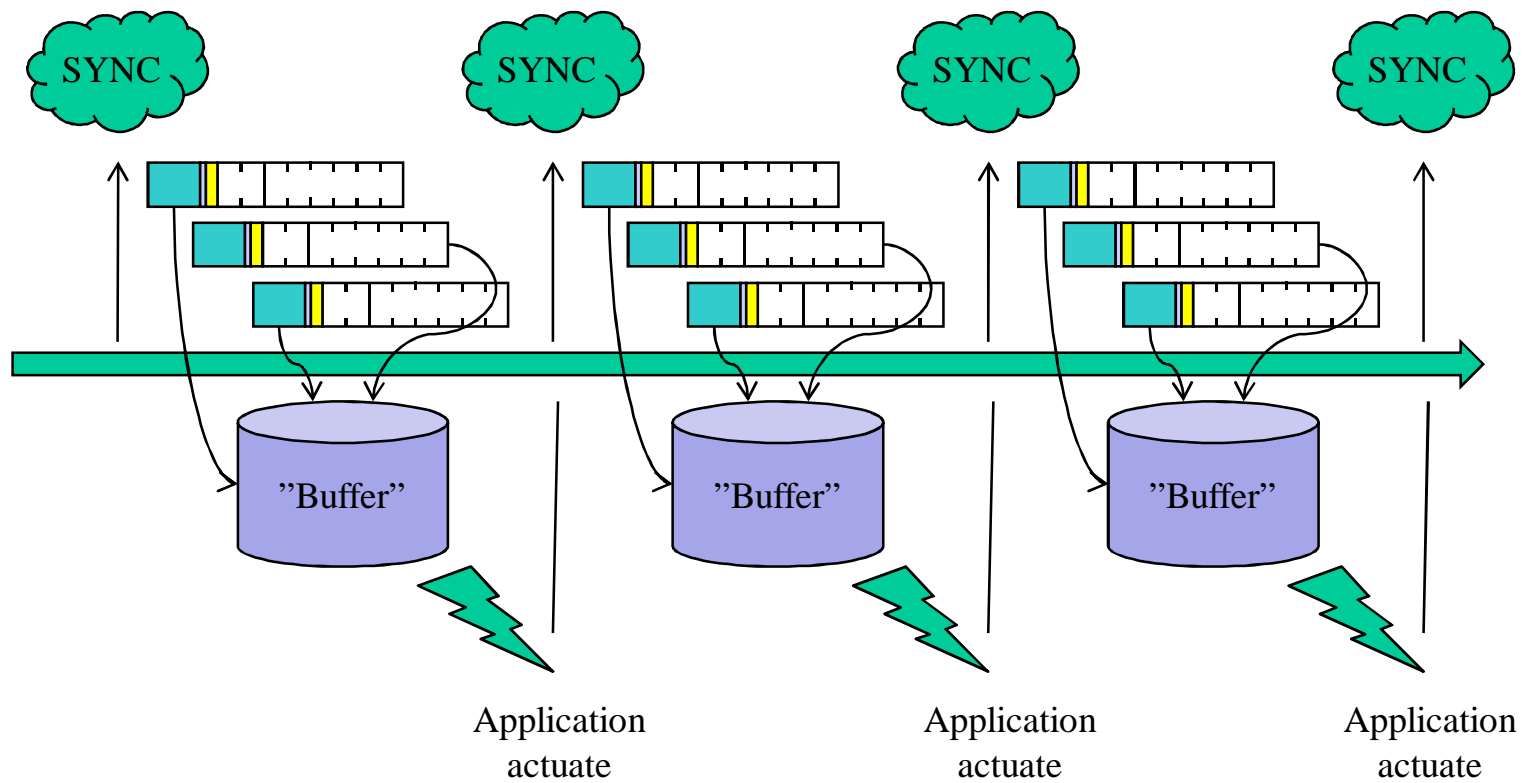
Configuration tool



Sync of transmit PDO



Synchronized RPDO



PDO transmission type

Transm. Type	Meaning for a transmit PDO	Meaning for a receive PDO
0	Sent on next SYNC if event or request has been made.	Application updated on next SYNC.
$1 < n < 240$	Sent on every n SYNC	Application updated on next SYNC.
$241 \leq n < 252$	UNDEFINED	UNDEFINED
252	Sent on next SYNC if PDO has been requested.	UNDEFINED
253	Sent independent of SYNC upon request.	UNDEFINED
254 -255	Sent independent of SYNC in all cases	Application is updated upon reception of PDO

Error Control Protocols

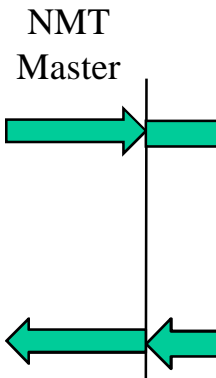
-]

ocol

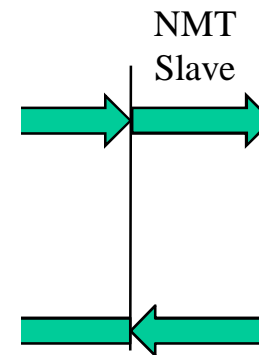
ol

CONFIDENTIAL

Node Guarding Protocol



dlc = 1



CONFIDENTIAL

Name	
Node guard time	
Node life time	
Life Guard Event	Event triggered if NMT slave has not been polled within "Node Life time" (jumps to pre-operational mode).
Node Guard Event	Event triggered if node guard time time has elapsed and no response from slave.

CAN trace (node 2)

```

00000702 R [no data]
00000702 81
00000702 R [no data]
00000702 01
00000702 R [no data]
00000702 81
    
```

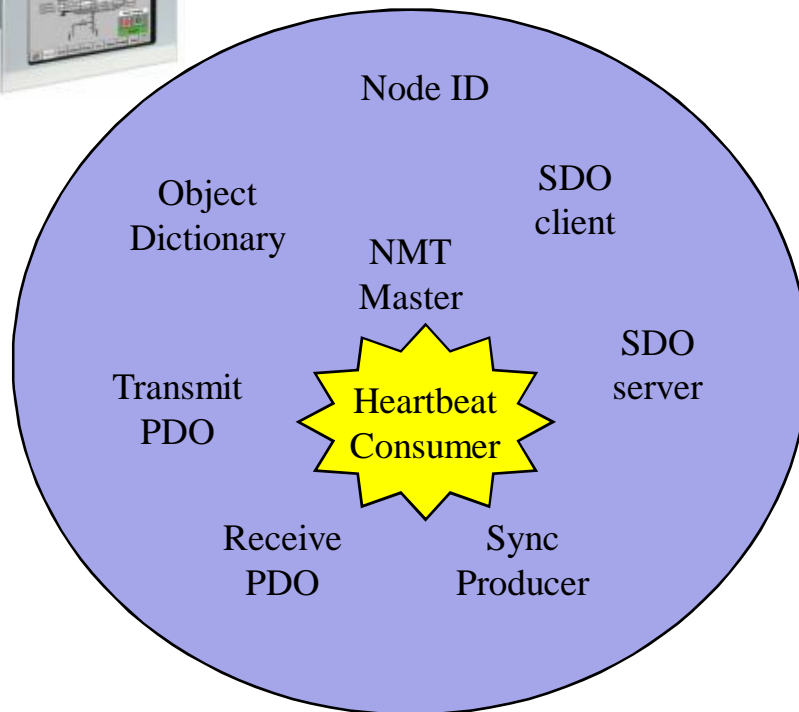
Heart Beat Protocol



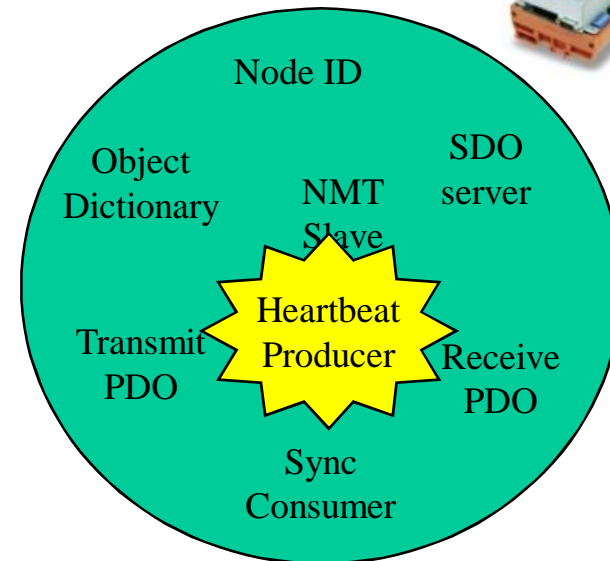
Node functionality



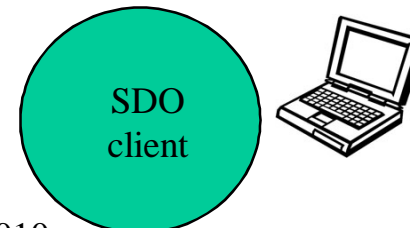
Process computer



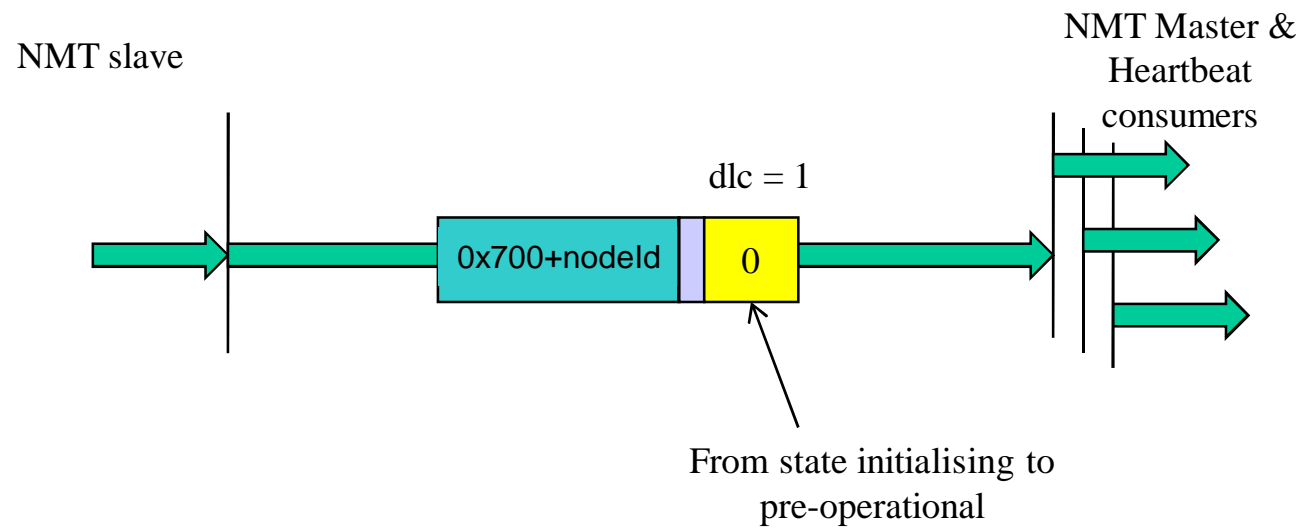
Network node



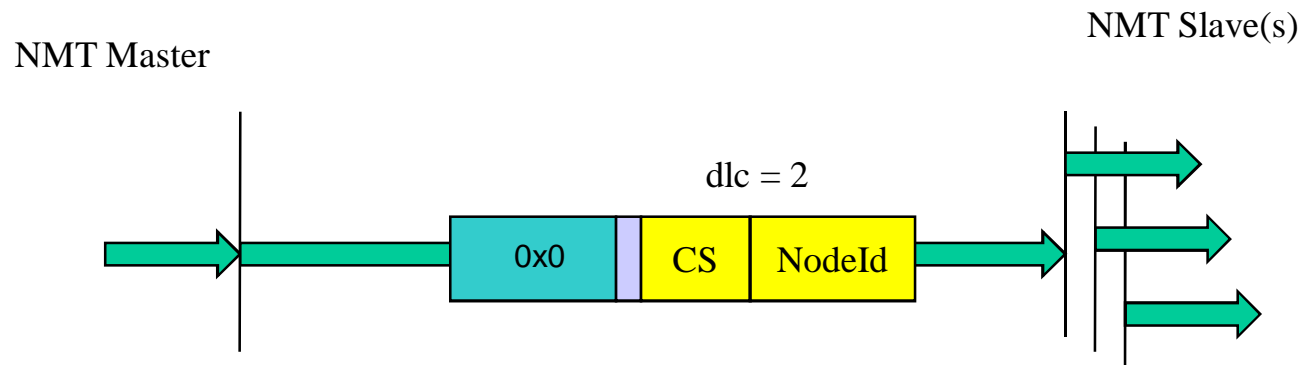
Configuration tool



Bootup Protocol



Module Control Protocols

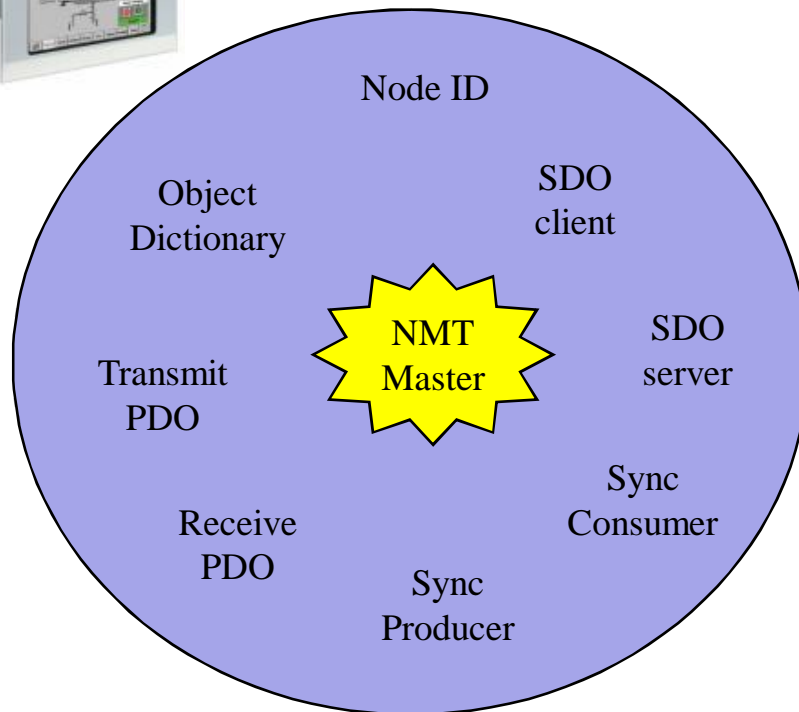


CS (Command Specifier)	Command
1	Start Remote Node
2	Stop Remote Node
128	Enter Pre-Operational Mode
129	Reset Node
130	Reset Communication

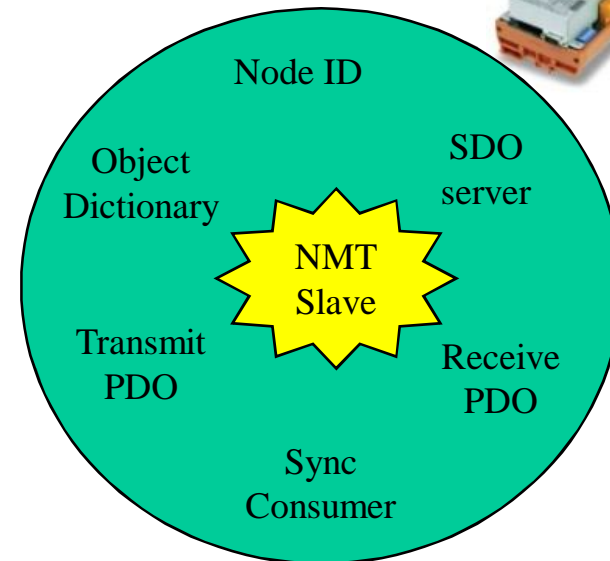
Node functionality



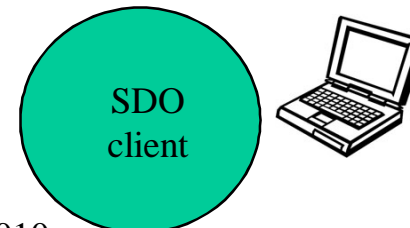
Process computer



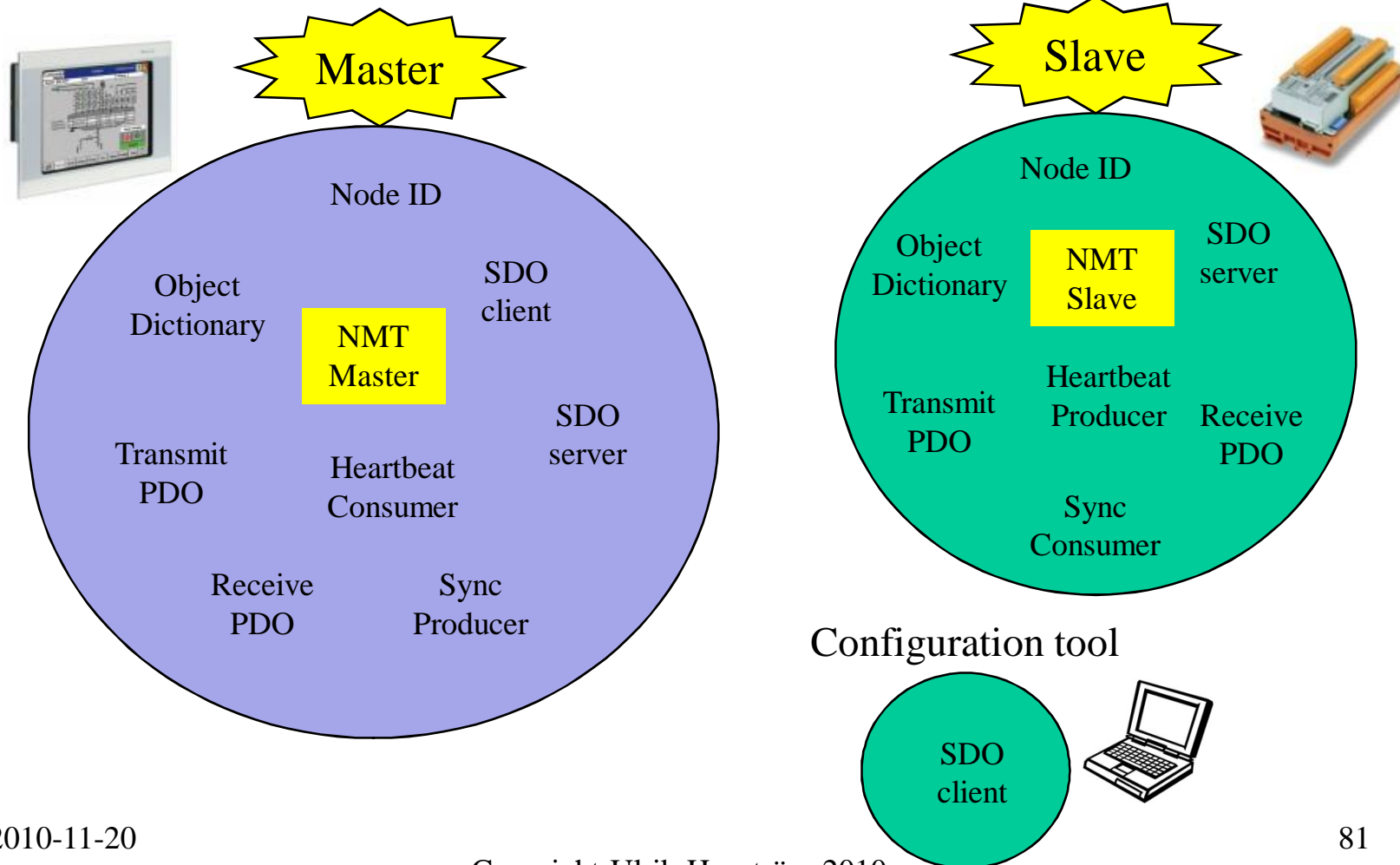
Network node



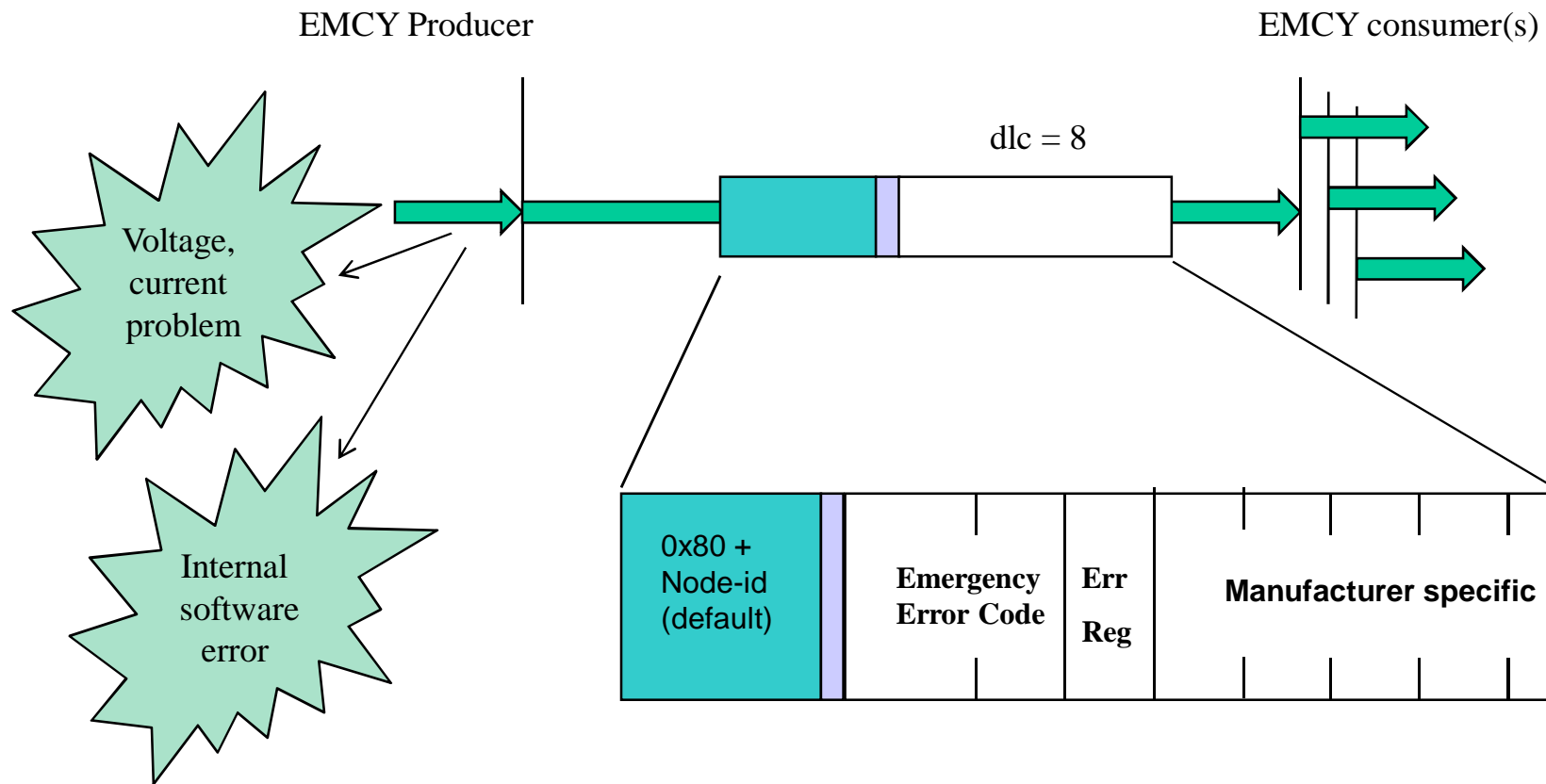
Configuration tool



Node functionality



Emergency Protocol



2010-11-20

Copyright Ulrik Hagström 2010

82

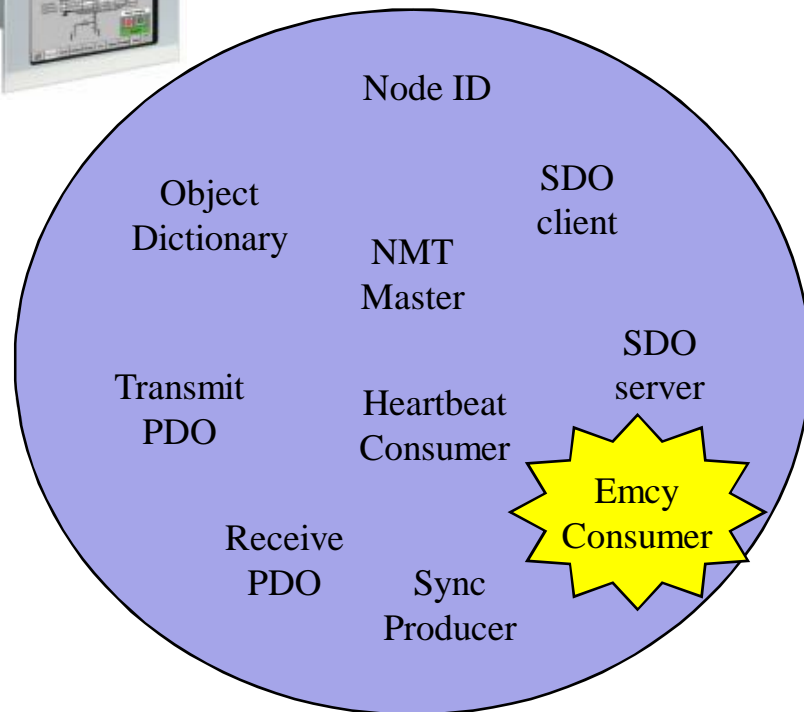
Pre-defined error field (0x1003)

- The object at index 0x1003 holds the errors that have occurred on the device and have been signaled via the **Emergency Object**.
- The entry at sub-index 0 contains the number of actual errors
- Every new error is stored at sub-index 1, the older ones move down the list.
- Writing a 0 to sub-index 0 deletes the entire error history (empties the array).

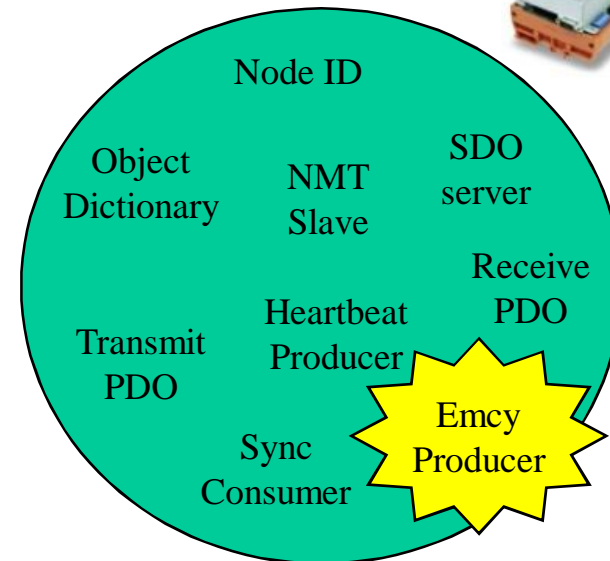
Node functionality



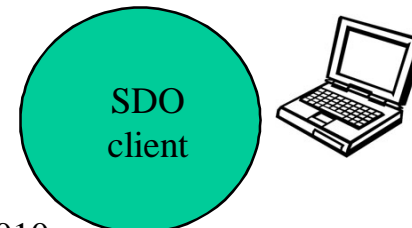
Master



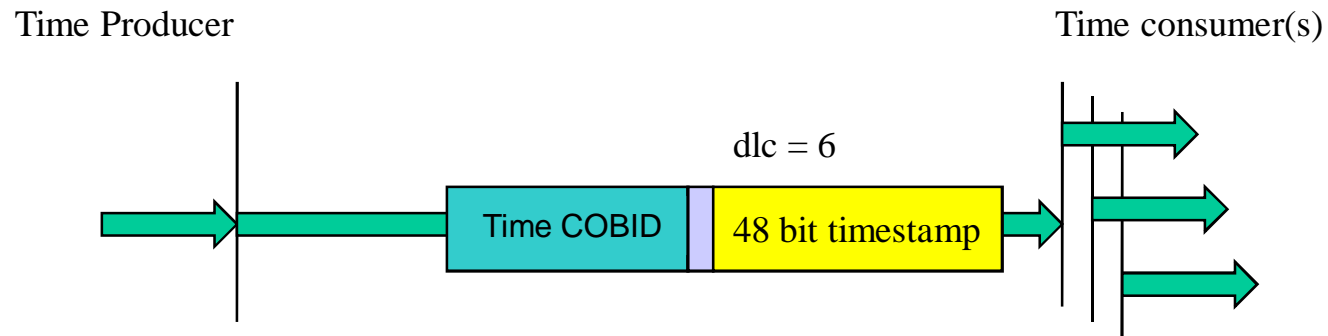
Slave



Configuration tool



Time Stamp Object

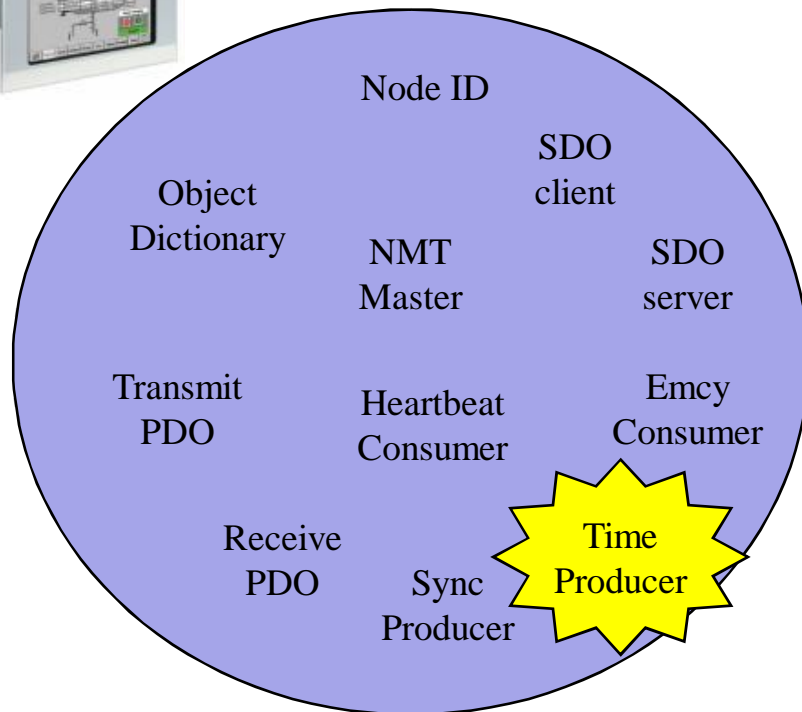


Object	Parameter
0x1012	COB-ID (Bit 30: consumer/producer flag)

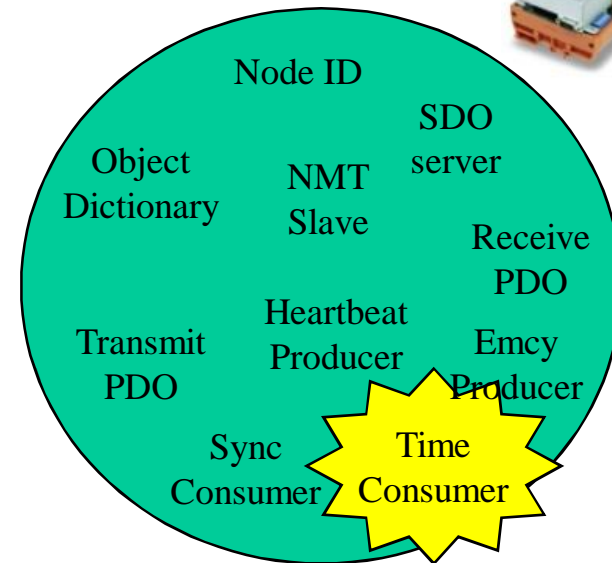
Node functionality



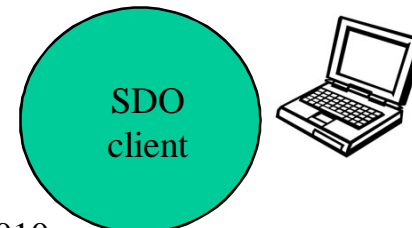
Master



Slave



Configuration tool



Press release for Device Profile 445 (RFID) example

Erlangen, Germany, September 4, 2007 -- The CAN in Automation (CiA) nonprofit organization has released the CiA 445 CANopen device profile for RFID readers/writers. The objective of the profile is to enable easy system integration of RFID readers into networks in factory automation, laboratory automation, medical systems, product and asset management, identification systems, etc. *The device profile will make CiA 445-compliant RFID readers from different manufacturers interchangeable with a minimum of time and configuration effort...*

...The following companies have participated in the development of the profile specification within the CANopen Special Interest Group RFID:

DeLaval International, FH Regensburg, Hans Turck, ifm electronic, Ixxat Automation, RM Michaelides, Schneider Electric, Sick, Siemens Medical Solutions, Vector Informatik, and others.

CANopen in practice



CONFIDENTIAL



EDS (Electronic data Sheet) file

- CiA-306


- The EDS belongs to the standard d supplied with a CANopen dev
- A proper EDS file is required to pas CANopen conformance test
- In the future these files will be rep XML device descriptions according to IS (CiA-311)
- CANeds free-of-charge EDS generator/editor (www.canopen-forum.com)

```
EDS example  
[FileInfo]  
FileName=example1.eds  
FileVersion=?
```

CONFIDENTIAL

```
DefaultValue=0x1  
PDOMapping=0
```

DCF (Device configuration File)

- EDS file containing the device configuration parameters.
- Used by Configuration Manager (either stand alone tool or CANopen Master node).
- Stand alone tool for service engineers: www.tke.fi 
- Configuring node standalone prevents node-id overlapping, faulty bitrate etc. that would cause bus chaos.

DS-302

Framework for Programmable CANopen Devices

CONFIDENTIAL

Manager, Configuration Manager (opt) }

Object Dictionary:

Configuration (Am I the NMT master ?)

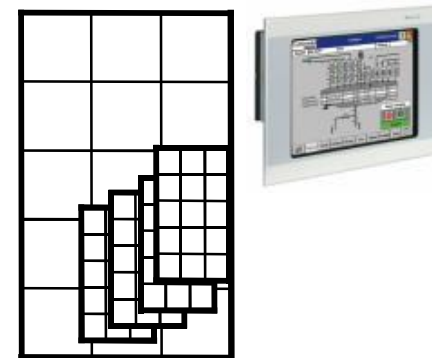
Work list, device type, serial number,

keep alive (node guard),

FW version (optional upgrade) }

Segment (CMT)

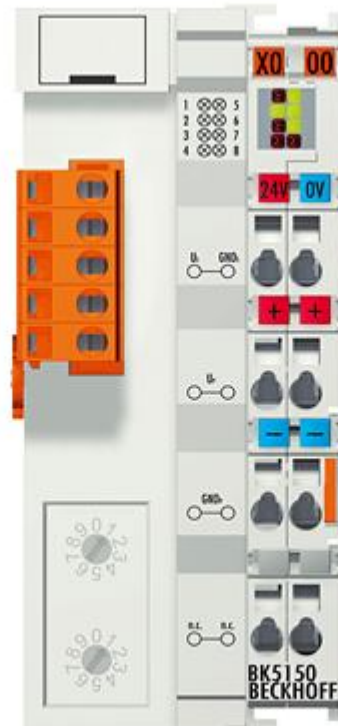
This makes it possible
to use third party CANopen
design tools.



Object Dictionary

Object Index	Sub Index	Data Type	Bit contents	Description
0x1f26		ARRAY		Expected configuration time.
0x1f27		ARRAY		Expected configuration date.
0x1f50		ARRAY		Download program data.
0x1f51		ARRAY		Program Control.
0x1f52		APPL SW		Verify application software.
	0x1			Application software date.
	0x2			Application software time.
0x1f53		ARRAY		Expected application sw date.
0x1f54		ARRAY		Expected application sw time.
0x1f80		UINT8		NMT Startup
			BIT 0	0 = NMT slave, 1 = NMT master.
			BIT 1	0 = Start node by node, 1 = start all nodes.
			BIT 2	0 = Automatic start, 1 = Application allows start (node local)
			BIT 3	0 = Start slaves, 1 = Application starts slave.
			BIT 4	0 = slave specific err handling, 1 = Err control event handling
0x1f81 – 0x1f89		ARRAY		DS302 Boot Parameters...
0x1fa0 – 0x1fcf		ARRAY		Object Scanner List (Multiplexed PDOs)
0x1fd0 – 0x1fff		ARRAY		Object Dispatcher List (Multiplexed PDO)

Lab



- Boot-up
- Emergency Error
- Read/Write parameters from OD using SDO protocol.
- PDO mapping
- Timer driven PDO transfer
- Module control protocol (operational states).

Advanced CAN

- Controller types.
- Signal levels.
- Oscilloscope pictures.
- Error detectoion
- Connectors

Three types of CAN controllers

Part A and Part B comp ability

There are three types of CAN controllers: Part A, Part B passive and Part B. They are able to handle the different parts of the standard as follows:

CAN chip type	Part A	Part B passive	Part B
11 bit identifier	OK.	OK.	OK.
29 bit identifier	ERROR!	Tolerated on the bus, but ignored.	OK.

Recessive & Dominant levels

When a logic '1' is written to the bus, the two wires sit at 2.5V and is termed a "recessive" bit.

CAN_HI

CAN_LO

Recessive signal
(= 2.5V)

Dominant signal
(= 1 – 1.5V)

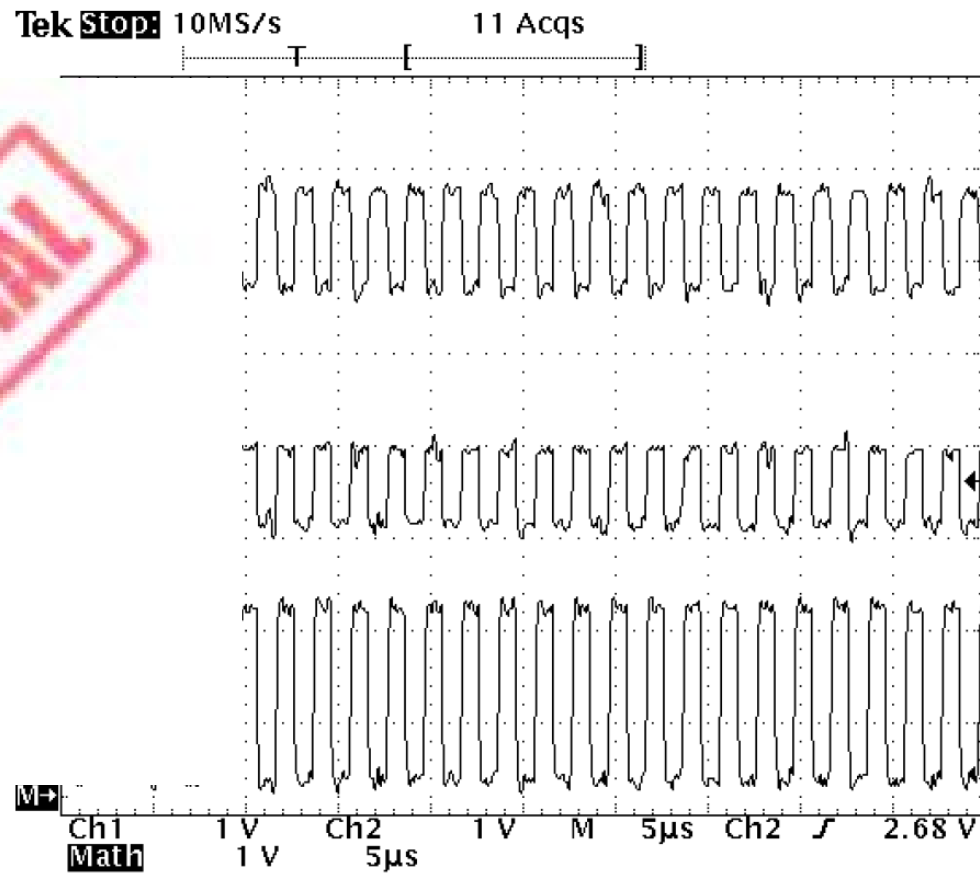
CONFIDENTIAL

is written to the bus,
| to 5V (CAN HI)
ulled down to
)

If both lines are at the same voltage, the signal is a recessive bit. **If the CAN_HI line is higher than the CAN_LO line by 0.9V, the signal line is a dominant bit.** If just one node is driving the bus to a logical 0 (=dominant bit), then the whole bus is in that state regardless of the number of nodes transmitting a logical 1. Copyright Ulrik Hagström 2010

CAN signals

CONFIDENTIAL

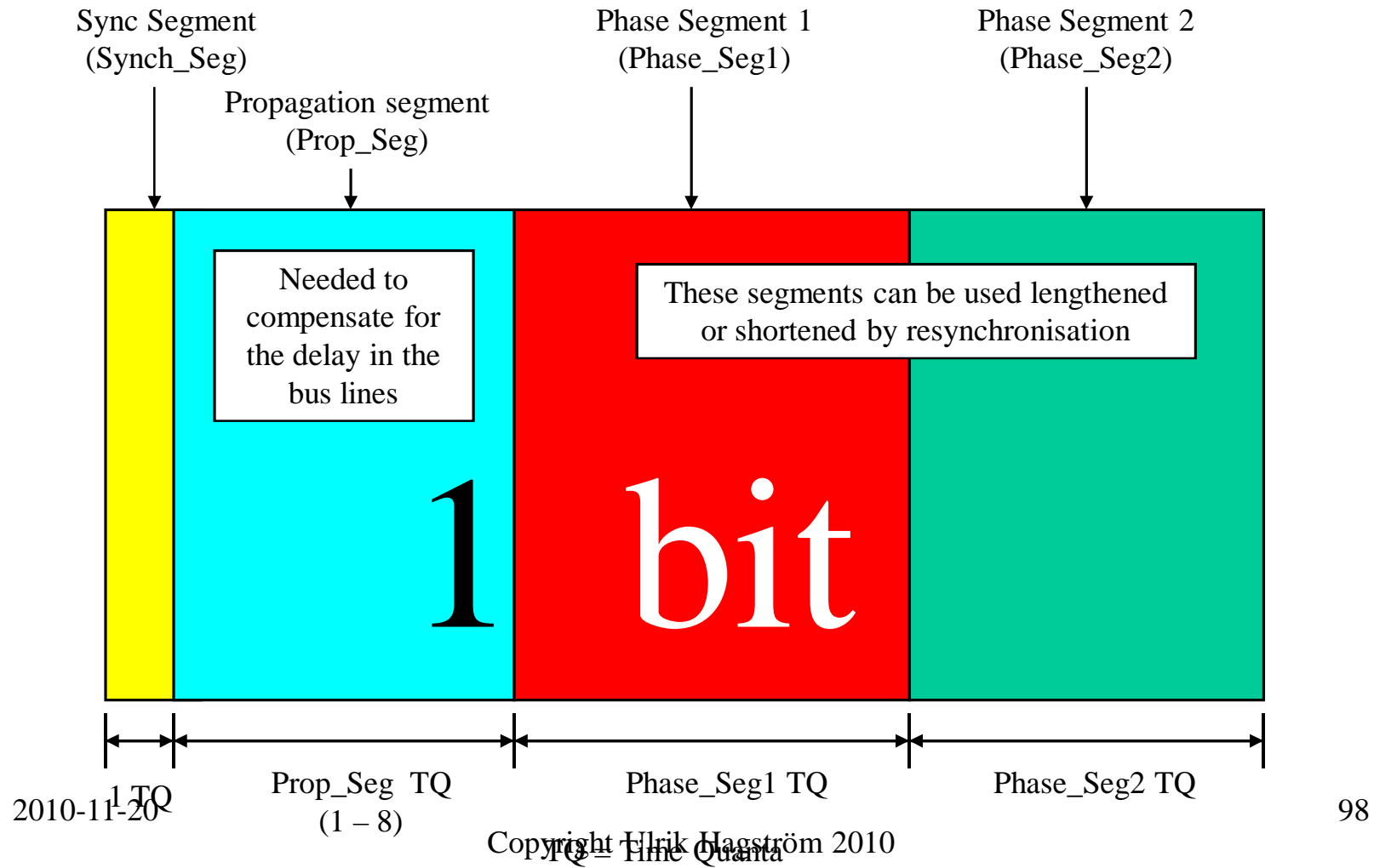


2010-11-20

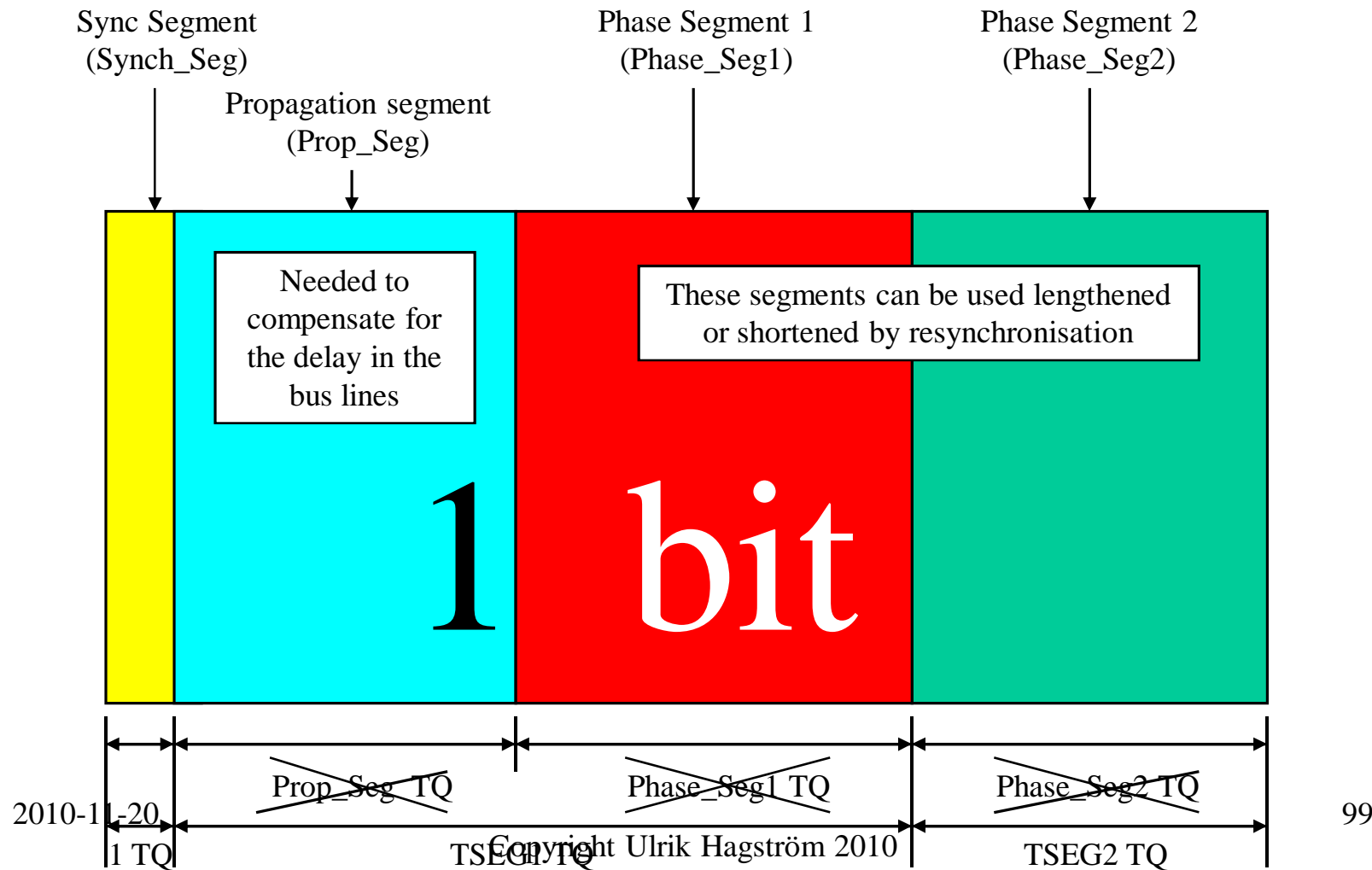
Copyright Ulrik Hagström 2010

97

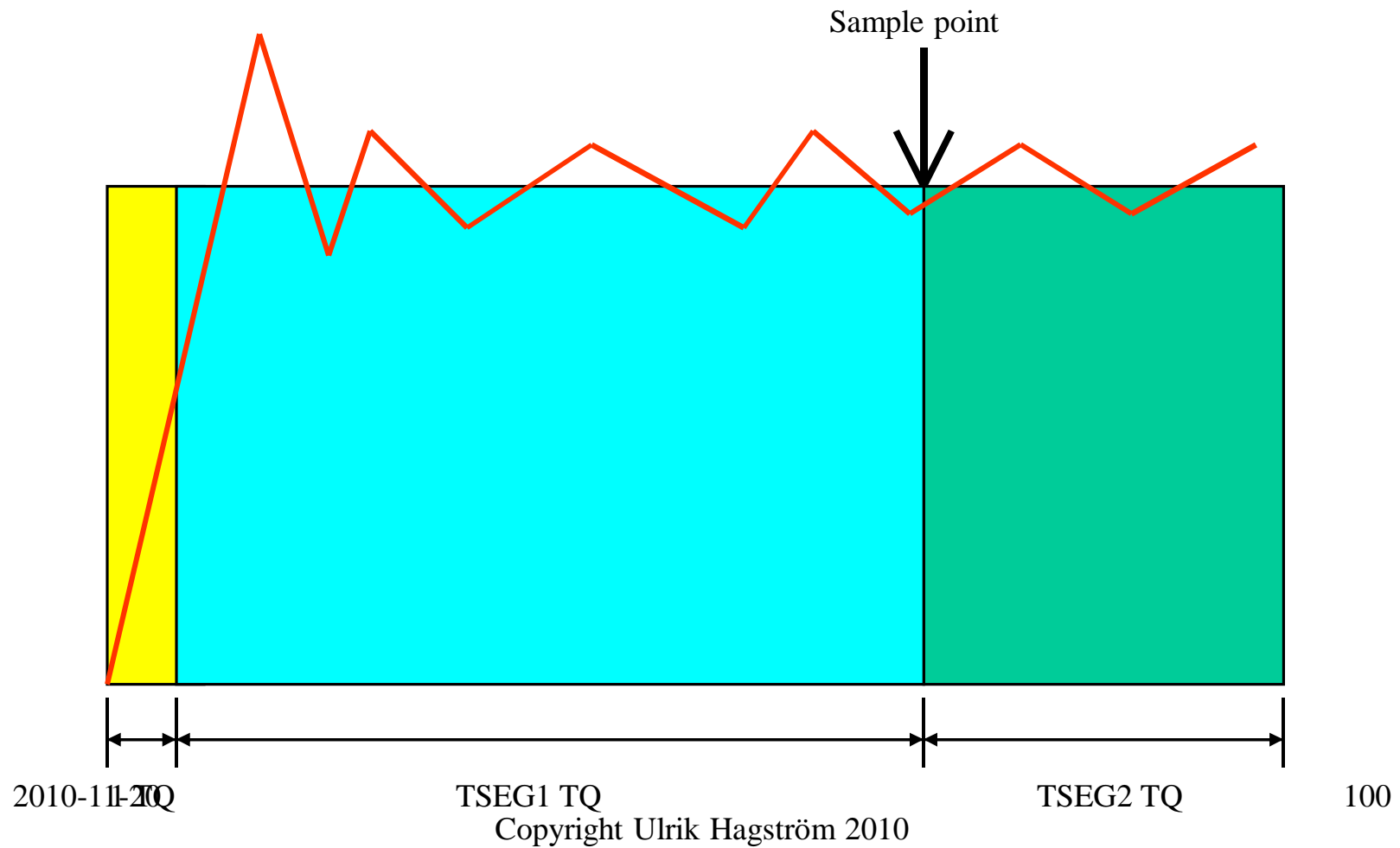
1 bit BOSCH-specification



1 bit on ISO11898-specification



Sample point per bit



Bitrate settings

$n = SY$

BRP = val
(re

Bitra



Resync and SJW

Hard resynchronization

Resynchronization within a frame

CONFIDENTIAL

Typical settings...

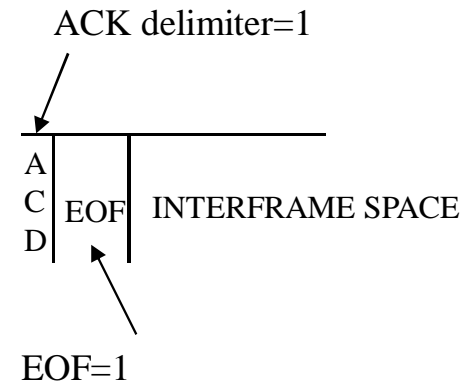
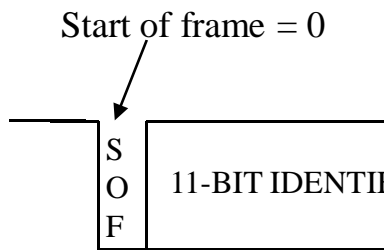
Bit rate Bus length ⁽¹⁾	Nominal bit time t_b	Number of time quanta per bit	Length of time quantum t_q	Location of sample point
1 Mbit/s 25 m	1 μ s	8	125 ns	6 t_q (750 ns)
800 kbit/s 50 m	1,25 μ s	10	125 ns	8 t_q (1 μ s)
500 kbit/s 100 m	2 μ s	16	125 ns	14 t_q (1,75 μ s)
250 kbit/s 250 m ⁽²⁾	4 μ s	16	250 ns	14 t_q (3,5 μ s)
125 kbit/s 500 m ⁽²⁾	8 μ s	16	500 ns	14 t_q (7 μ s)
50 kbit/s 1000 m ⁽³⁾	20 μ s	16	1,25 μ s	14 t_q (17,5 μ s)
20 kbit/s 2500 m ⁽³⁾	50 μ s	16	3,125 μ s	14 t_q (43,75 μ s)
10 kbit/s 5000 m ⁽³⁾	100 μ s	16	6,25 μ s	14 t_q (87,5 μ s)

The five error checks...

- **Bit monitoring (read back)**
 - **Bit stuffing (toggle required)**
 - **Frame check (predefined values)**
 - **Acknowledgement check (received?)**
 - **CRC check.**
- Error Frame → Automatic retransmission

Form- and biterror

- Form

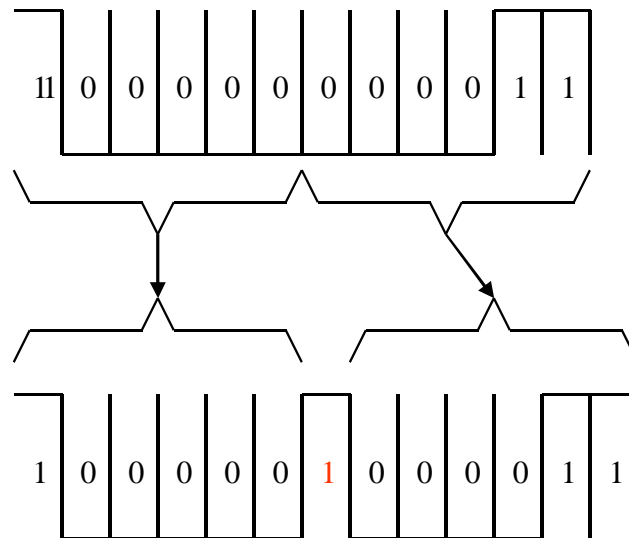


- Bit error

If a bit is written incorrectly, a “Bit error” is generated (DOES NOT APPLY to IDENTIFIER or ACK-bit)

s read back

Bit stuffing error



Acknowledgement check

CRC check and the acknowledge slot ("Form error",
"bit stuff" "Ack Error")

Transmitter:



Receiver:



CONFIDENTIAL

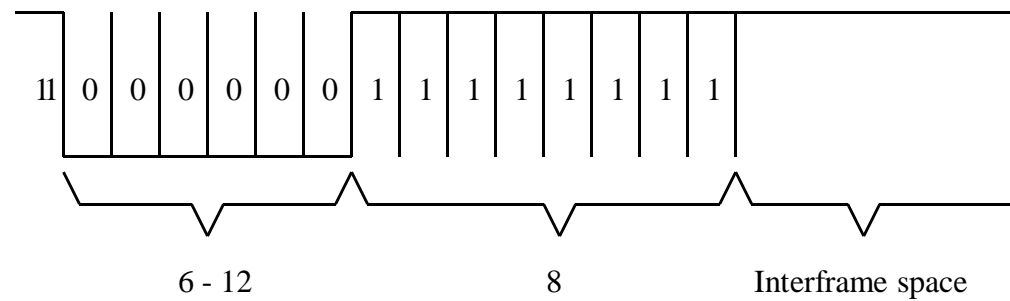
Recessive.



Dominant if OK.



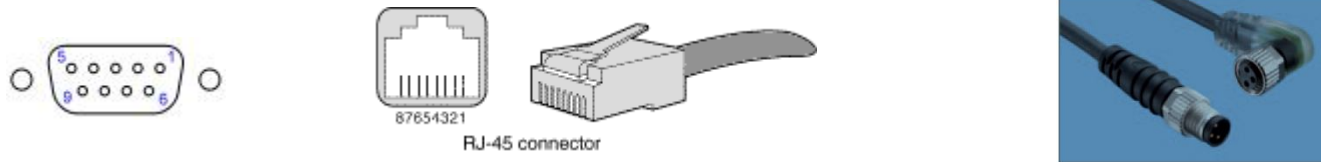
CAN error frame



CAN controller error modes

- **Error active**
Tx error counter ≤ 127 AND Rx error counter ≤ 127
- **Error passive**
(Tx error counter > 127 OR Rx error counter > 127) AND Tx error counter ≤ 255 .
- **Bus off**
(Tx error counter > 255)

CAN connectors



<http://www.erni.com/DB/PDF/M8M12/ERNI-M8M12-e.pdf>

Pin #	Signal Names	Signal Description	RJ45 Pin #	RJ10 Pin #	Signal Name	Signal Description
1	Reserved	Upgrade Path	1	2	CAN_H	CAN High
2	CAN_L	CAN Low	2	3	CAN_L	CAN Low
3	CAN_GND	Ground	3	4	CAN_GND	Ground
4	Reserved	Upgrade Path	4	-	Reserved	Upgrade Path
5	CAN_SHLD	Shield, Optional	5	-	Reserved	Upgrade Path
6	GND	Ground, Optional	6	-	CAN_SHLD	CAN Shield, Optional
7	CAN_H	CAN High	7	-	CAN_GND	Ground
8	Reserved	Upgrade Path	8	1	CAN_V+	Power, Optional
9	CAN_V+	Power, Optional				

One logic to several physical



Contact: sales@datalink.se

Rx error counter rules

- Receiver detects error (any): the Rx error counter will be increased by 1, except when the detected error was a bit error during the sending of an active error flag or an overload flag (=this specific node did not see the error that another node saw).
- Receiver detects a dominant bit as the first bit after sending an error flag: the Rx error counter will be increased by 8.
- If a receiver detects a bit error (“what was written was not read”) while sending an active error flag or an overload flag the Rx error counter is increased by 8.
- *After the successful reception of a message (reception without error up to the acknowledge slot and the successful sending of the acknowledge bit), Rx error counter is decreased by 1 if it was between 1 and 127. If Rx error counter was 0 it stays 0, and if it was greater than 127, it will be set to a value between 119 and 127.*

Tx error counter rules

- When a transmitter sends an error flag, the Tx error counter is increased by 8. Important exception: If a node is the only one on the bus (or during start-up the only one that has become active), and it transmits a message, it will get an acknowledgement error, and will retransmit the message. This may lead to that node going to error passive mode – but it will not go bus off (=“oscillate”)
- If a transmitter detects a bit error while sending an active error flag or an overload flag, the Tx error counter is increased by 8.
- *After the successful transmission of a message (getting ack and no error until end of frame is finished) Tx error counter is decreased by 1 unless it was already 0.*

Advanced CANopen

- Multiplex PDO.
- Object Dispatcher List.
- Object Scanner List.

Multiplexed PDO

- Multiplexed PDOs are SDO/PDO hybrids for objects with a size of 1 – 32bits.
- Write to any OD entry (1-32 bits) on remote node without using SDO transfer.

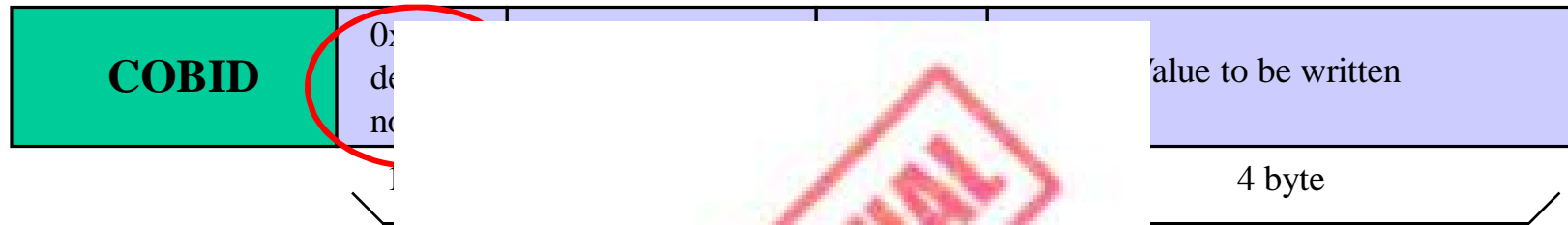
Multiplexed PDO types

- Destination Addressing Mode MPDO
(DAM MPDO)

- Source Addressing Mode MPDO
(SAM MPDO)

DAM MPDO

Destination Addressing Mode Multiplexed PDO



PDO mapping parameters R

Object Index
1x1600 – 0x17f
...
0x1a00 – 0x1bf

PDO mapping parameters TPDO.

CONFIDENTIAL

Value
255 (=DAM MPDO)
...
255 (=DAM MPDO)

Normal PDO:s are only 0..64

SAM MPDO

Source Addressing Mode Multiplexed PDO



CONFIDENTIAL

PDO mapping parameters

Object Index
1x1600 – 0x
...
0x1a00 – 0x

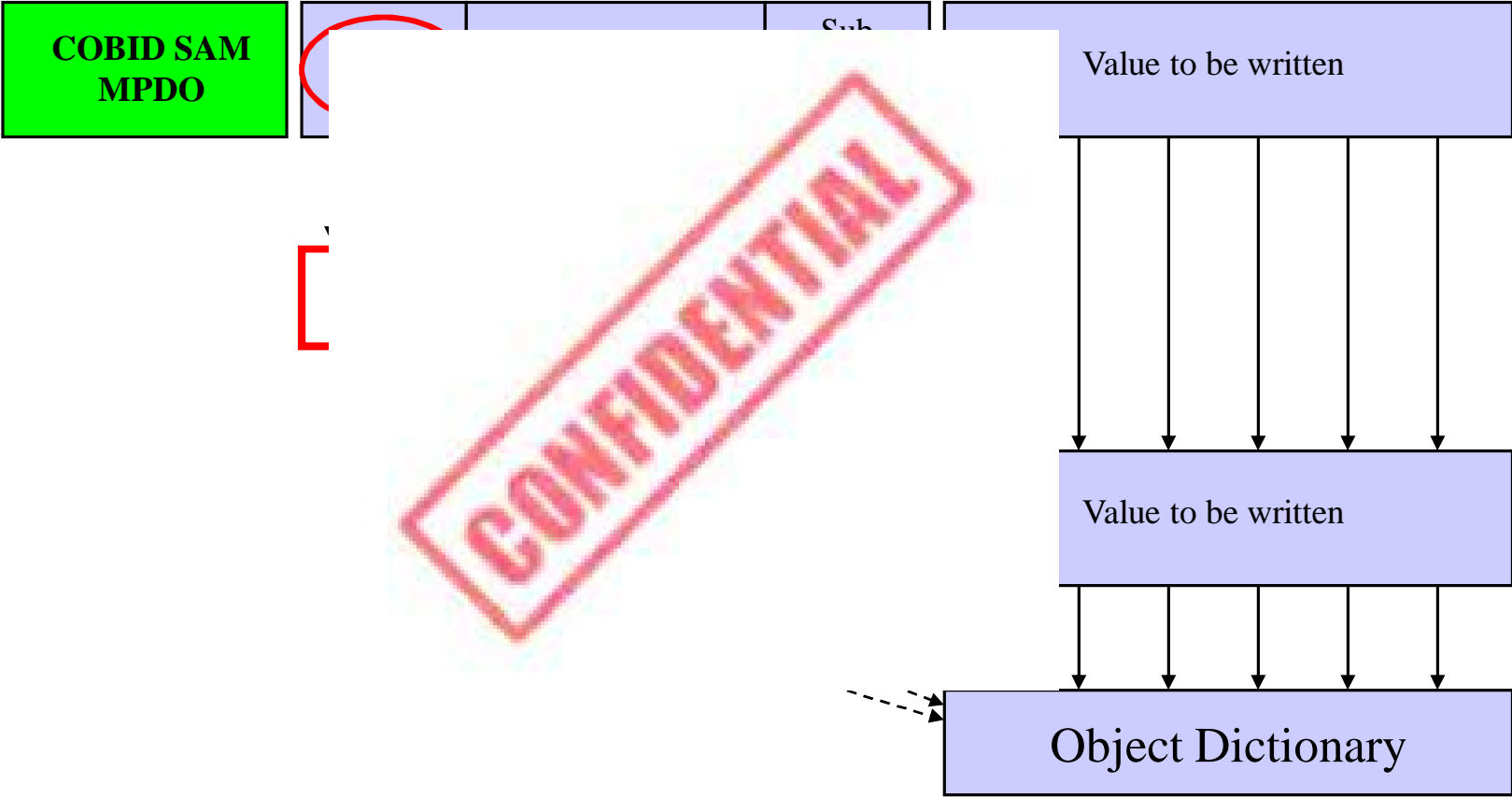
PDO mapping parameters TPDO.

Value
254 (=SAM MPDO)
...
254 (=SAM MPDO)

Normal PDO:s are only 0..64

Object Dispatcher List

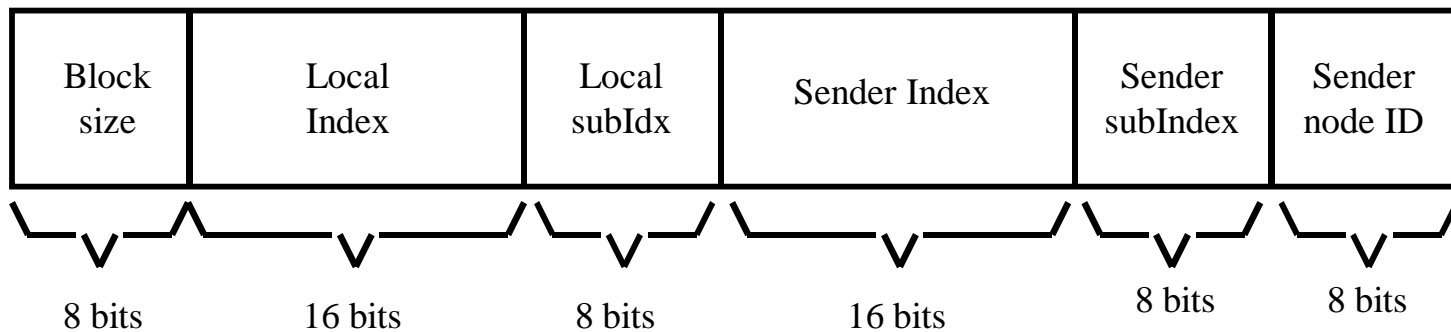
(used when node receive a SAM-MPDO)



Object Dispatcher List

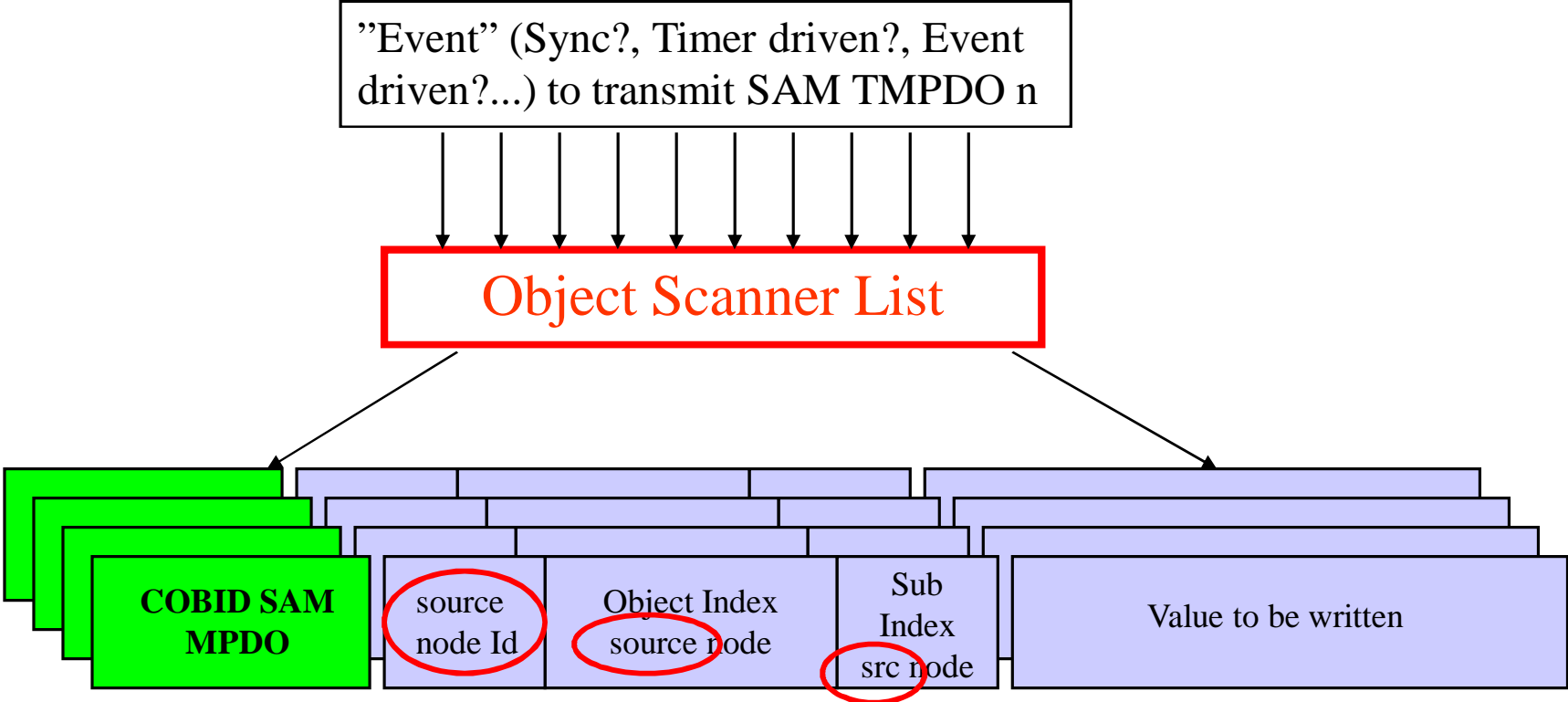
(object used for configuring)

Object Index	Sub Index	Data Type	Description
0x1fd0 – 0x1fff	0	UINT8	Number of configured dispatchers
	0x1	UINT64	Object Dispatching 1
	-- 0xfe	UINT64	Object Dispatching ...254



Object Scanner List

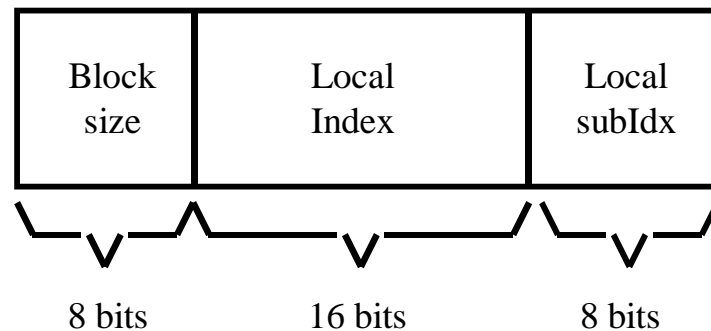
(used when node transmits SAM-MPDO)



Object Dispatcher List

(object used for configuring)

Object Index	Sub Index	Data Type	Description
0x1fd0 – 0x1fff	0	UINT8	Number of configured dispatchers
	0x1	UINT32	Object Dispatching 1
	-- 0xfe	UINT32	Object Dispatching ...254



Transmission Type

Object Index	Sub Index	Data Type	Bit contents
0x1801	1 (COBID)	UINT32	0x123
0x1801	2 (Transmission time)	UINT8	254
0x1801	3 (Inhibit time)	UINT16	100 (*10us)
0x1801	4 ()	0	0
0x1801	5 (Event timer)	UINT16	1000 (*1ms)

Transmission Type	Meaning for a transmit PDO	Meaning for a receive PDO
0	Sent on next SYNC if event or request has been made.	Application updated on next SYNC.
1 < n < 240	Sent on every n SYNC	Application updated on next SYNC.
241 <= n < 252	UNDEFINED	UNDEFINED
252	Sent on next SYNC if PDO has been requested.	UNDEFINED
253	Sent independent of SYNC upon request.	UNDEFINED
254 -255	Sent independent of SYNC in all cases	Application is updated upon reception of PDO