CANopen

A technical introduction

Version 3.1 / September 2005

Merlin Gerin
Square D
Telemecanique

Schneider Electric

Table of Content

- History of CAN / CANopen
- Technical features
- CANopen characteristics
- CANopen protocol
- Diagnostic
- Error detection
- What you should remember

History of CAN / CANopen

Origin of CAN

- 1983 Bosch developed the CAN protocol
- 1986 Official introduction of CAN protocol
- 1987 First CAN controller chips from Intel and Philips Semiconductors
- 1989 Low-cost controller chips implementing the CAN data link layer protocol in silicon
- 1991 Bosch’s CAN specification 2.0 published

Origin of CANopen

- **1992-93** **CiA Working Group „Higher Layer Protocols“ specifies CAL (CAN application layer)**
- **1993-94** **Specification and prototyping of a distributed system for production cells**
- **1995** **Formation of CiA Interest group CANopen, specifies CiA Draft**
  - → „Communication profile for industrial automation systems“ (DSP-301)
  - → „Device profile for I/O-modules“ (DSP-401)
- **1996-98** **Specification of standard device profiles, extensions of the communication profile and standard applications**
- **2003** **CANopen is established as a protocol for different automation applications**

Technical features

At at glance

- Flexibility
- Fault tolerant capabilities
- Increased reliability
- Design change flexibility
- EMC immunity
- Better system integrity
- Simplified diagnostics capabilities
- Powerful error detection capabilities

Technical features

Protocol advantages

- Configuration flexibility
- Prioritization of messages
- System wide data consistency
- Multicast reception
- Error detection and error signaling
- Automatic retransmission of corrupted messages
- Distinction between temporary errors and permanent failures of nodes and autonomous switching off of defective nodes

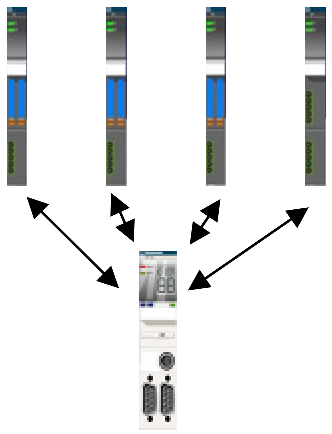
CAN / CANopen specifications

- **CAN Protocol Specification 2.0 A: CAN Controller compliant with this standard handles only standard frames with 11-bit identifiers.**
- **CAN Protocol Specification 2.0 B passive: CAN Controller compliant with this standard transmit only standard frames with 11-bit identifiers, but checks received standard frames as well as extended frames with 29-bit identifiers (even the acknowledge is given).**
- **CAN Protocol Specification 2.0 B active: CAN Controller compliant with this standard can receive and transmit standard and extended frames.**

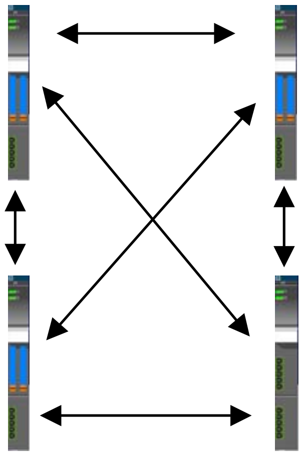
Miscellaneous

number of nodes	• Not limited by layer-2 protocol • Physical limitation by driver capacity of transceivers (64) • Logical limitation when using node specific identifiers (127)			
max. number of message identifier	• Standard format: 2048 (11 bit identifier 2^{11}) • Extended format: 536.870.912 (29 bit identifier 2^{29})			
Length of data field [bytes]	8	4	2	1
max. number of messages / s at 1Mbit/s	8772	12195	15150	17240
max. latency time of highest priority message / µs	135	88	78	68

Communication relations



1 : N



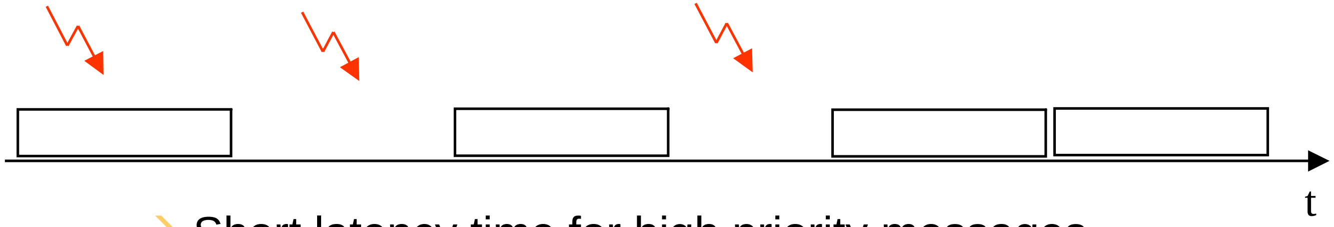
N : M

- **Multimaster Protocol**
  - → Any communication structure is possible
  - → Event-Oriented Message Transmission
    - – Reduced bus load
    - – Short latency time for real-time data

- **Priority-Based Message Transmission**
  - → Short latency time for high priority messages, even at very high bus load caused by low priority messages.

Data length

- **Limited Data Length**
  - → Sufficient for data communication in cars, machines, lower level automation
  - → Transmission of data also possible in electro - magnetically heavy disturbed environment



  - → Short latency time for high priority messages
  - → Segmented transmission of data more than 8 bytes

CANopen Characteristics

General

- **Bus Topology**
  - → High system configuration flexibility
  - → Restricted drop length
  - → Bus line termination required

- **Synchronous Protocol, NRZ Signal Coding**
  - → Improved usage of transmission bandwidth
  - → Bit synchronization mechanism required

- **Limited Product of Bus-Length*Data-Rate**
  - → Due to bit-wise arbitration the possible maximum bus length decreases with increasing data rate.

Data transmission and error detection

- **Random, Collision free Bus Arbitration**
  - → High priority message may monopolize the bus, appropriate system design required
  - → No collision solving mechanism required, non-destructive arbitration
- **Very Effective Error Detection Mechanisms**
  - → Very high data integrity
- **Error Signaling instead of Message Confirmation**
  - → Very short error recovery time
  - → System wide consistence of data
  - → Reduced bus load
- **Error Confinement Mechanisms**
  - → Switch off of defective nodes

Network length and bit rate

- **Total length at defined bit rate**

Bit rate [kBit/s]	1000	800	500	250	125	50	20	10
max. bus length [m]	20	40	100	250	500	1000	2500	5000

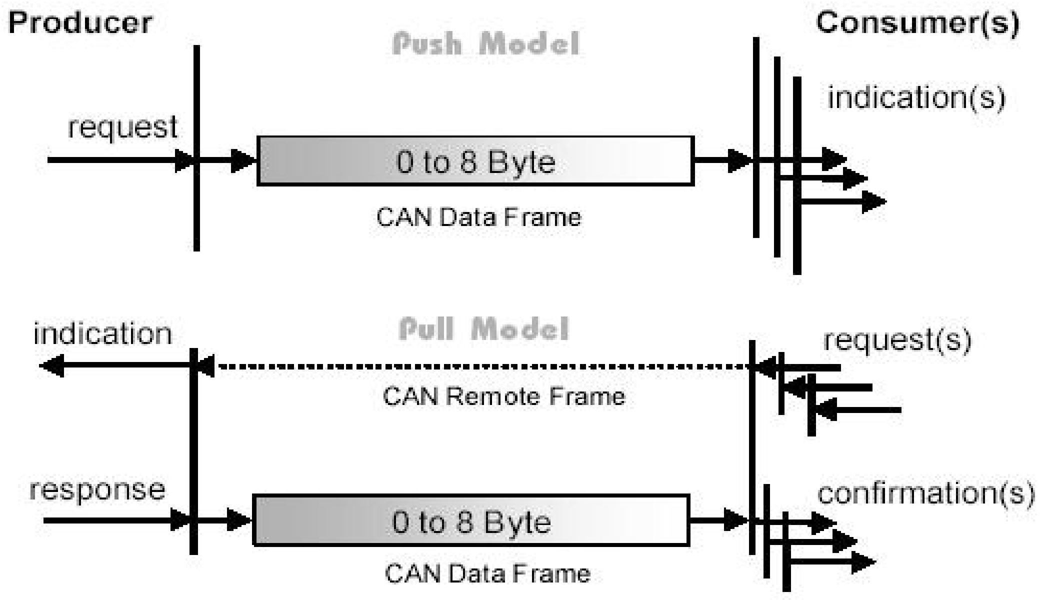
- **Drop length limitation**

Bit rate [kBit/s]	1000	800	500	250	125	50	20	10
Single Branch [m]	0.3	3	5	5	5	60	150	300
Sum per TAP [m]	0.6	6	10	10	10	120	300	600
Min. TAP Spacing* [m]		3.6	6	6	6	72	180	360
Total [m]	1.5	15	30	60	120	300	750	1500

* can be calculated for each TAP. TAP Spacing = 60% of the total of all branches in the TAP

Producer / Consumer

- CANopen is using the Producer/Consumer model. Each station of the network can listen to the messages of the transmitting station and decides, using the bit arbitration, if the messages is accepted or not.
- This model is the bases for the CAN broadcast communication.



The message types

- **Data messages**
  - → Process Data Object (PDO)
    - – fast transmission of process data
  - → Service Data Object (SDO)
    - – transmission of parameters

- **Predefined messages**
  - → Set of messages for synchronization (SYNC), time stamp distribution (TIME STAMP), notification of device failures (emergency message, EMCY).

- **Network Management messages (NMT)**
  - → Set of messages to control the node communication status and for monitoring the communication status of the device

PDO transmission modes

- **(1) Asynchronous Transmission**
  - Event-oriented transmission of a PDO after occurrence of profile - or manufacturer-specific event or after expiration of an event timeout
- **(2) Synchronous Transmission**
  - Transmission of a PDO immediately after reception of a specified number (PDO rate) of SYNC objects.
  - Acyclic synchronous: Single Transmission of a PDO
  - Cyclic synchronous : Repeated transmission of a PDO
- **(3) Transmission on Request**
  - Transmission of a PDO after reception of a request frame.
- **„Transport Capacity“ of a PDO = 8 Byte ⇒ max. 64Bit**

SDO transmission requirements

- **Access to all entries in Object Dictionary Entry**
  - → requires specification of Dictionary Entry by index (16 Bit) and subindex (8 Bit) within SDO protocol
- **Transmission of Object Dictionary Entries > 8 byte**
  - → requires fragmentation (segmentation) in SDO protocol (flow control)
- **Up- and download of data (read and write data)**
  - → requires specification of executed service within SDO protocol
- **Data length of most Object Dictionary Entries is <= 4 bytes**
  - → requires specific way of transfer of up to 4 data bytes within SDO protocol in order to save transmission time
- **A standard SDO is able to transmit 8 bytes**

SDO transmission types

- **Expedited Transfer**
  - → fast SDO transfer type for transferring data with up to 4 bytes
- **Non-Expedited Transfer (Segmented, Fragmented)**
  - → SDO transfer type for transferring data with any number of bytes
  - → Flow control after 7 transmitted bytes
- **Blocktransfer**
  - → SDO transfer type for transferring data with any number of bytes
  - → fast transfer method with flow control only after transmitting a block of data with n*7 bytes (1<=n<=127)
  - → Requires much more resources for implementation and processing than non-expedited transfer

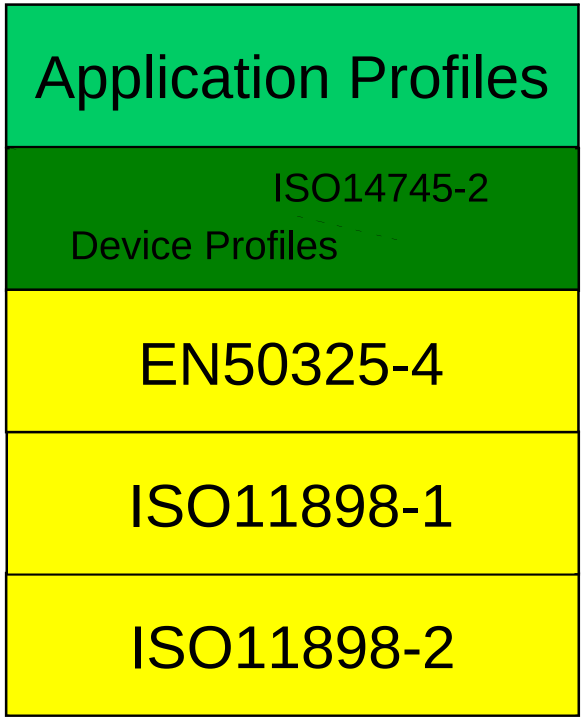
Watchdog mechanism

- **Two watchdog mechanisms exist for CANopen**
  - → Node Guarding
  - → Heartbeat

- **Node Guarding**
  - → The network manager is polling each device to check the health after a configured period of time

- **Heartbeat**
  - → Each device gives a life sign to the network manager or to other devices
    - – less bus load
    - – health check between devices possible

Layer model



Application Profiles

ISO14745-2

Device Profiles

EN50325-4

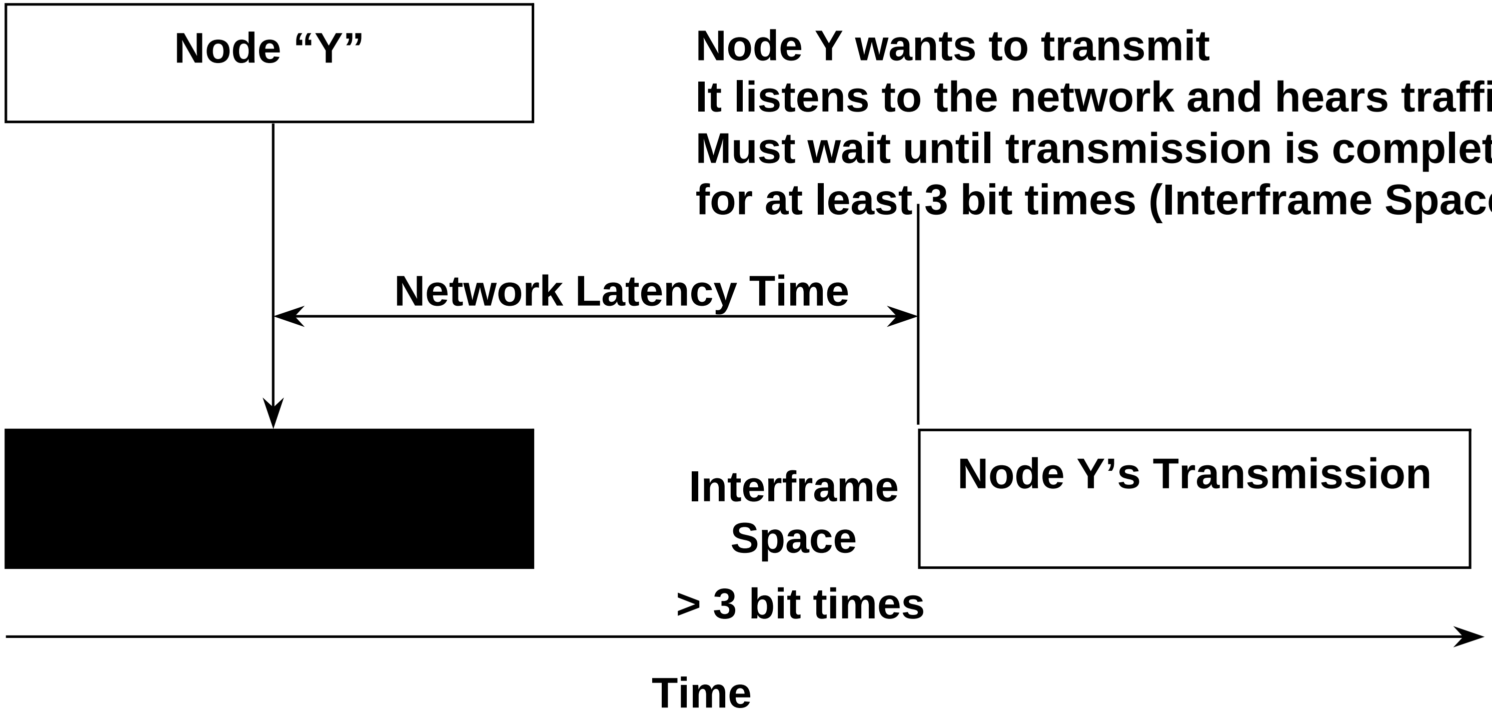
ISO11898-1

ISO11898-2

- **Application Profiles to reflect application needs**
- **Device Profiles & Device Description to support common behavior**
- **CANopen is using Layer 1, 2 and 7 of the ISO/OSI model**
  - → Layer 1 and 2 is CAN
  - → Layer 7 is CANopen

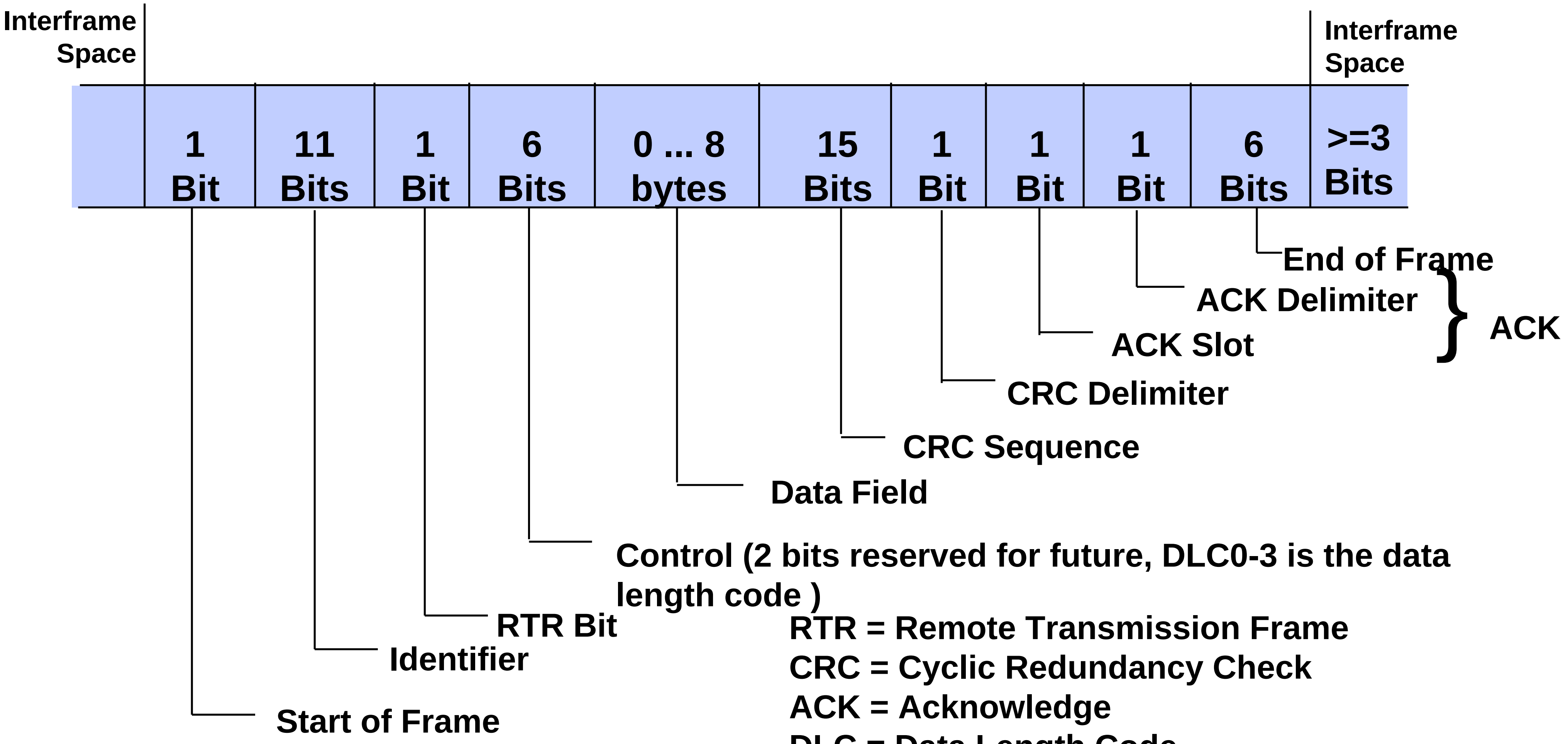
Media access logic

- If a transmission is occurring, a node must wait until it is complete before attempting to transmit



Node Y wants to transmit
It listens to the network and hears traffic
Must wait until transmission is complete
for at least 3 bit times (Interframe Space)

Frame format

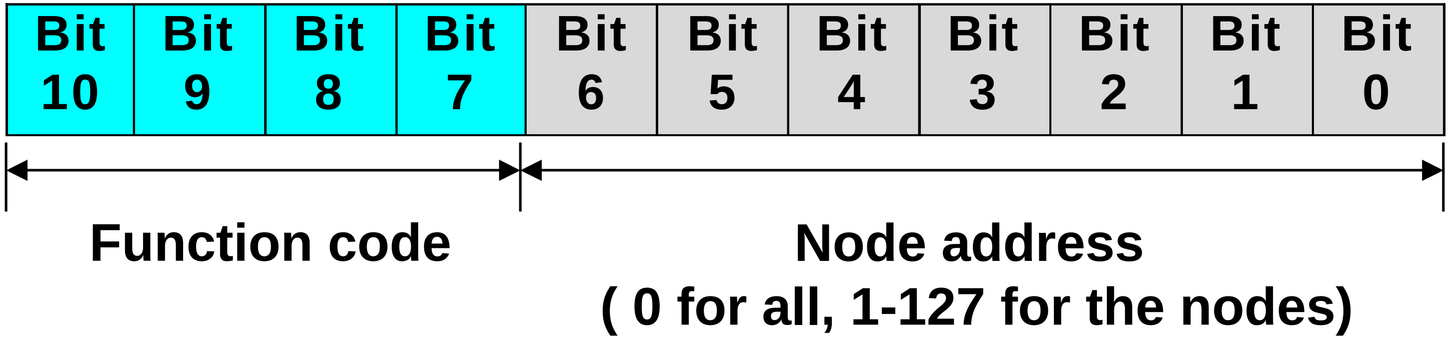


Interframe Space	1 Bit	11 Bits	1 Bit	6 Bits	0 ... 8 bytes	15 Bits	1 Bit	1 Bit	1 Bit	6 Bits	>=3 Bits
	Start of Frame	Identifier	RTR Bit	Control (2 bits reserved for future, DLC0-3 is the data length code)	Data Field	CRC Sequence	CRC Delimiter	ACK Slot	ACK Delimiter	End of Frame	Interframe Space

RTR = Remote Transmission Frame
CRC = Cyclic Redundancy Check
ACK = Acknowledge
DLC = Data Length Code

The CAN identifier (COB-ID)

- **COB-ID: Communication object identifier**
  - → Specification 2.0A or 2.0B passive supports 11 bit identifier



Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Function code

Node address
(0 for all, 1-127 for the nodes)

- **SYNC**
- **NMT Control**
- **EMCY**
- **TPDO**
- **RPDO**
- **TSDO**
- **RSDO**

Physical Signaling

- Bus level 0 = dominant
- Bus level 1 = recessive
- Bus idle = recessive
- The dominant level overrides the recessive level
- Bit coding is NRZ (Non-Return to Zero) w/bit stuffing
- Output of nodes:

#1	#2	#3	resulting bus level
0	0	0	0
0	0	1	0
0	1	0	0
	...		
1	1	1	1

Bus Arbitration Principle (1)

- **Several nodes may start transmission of a CAN frame as soon as they monitor the bus as idle**

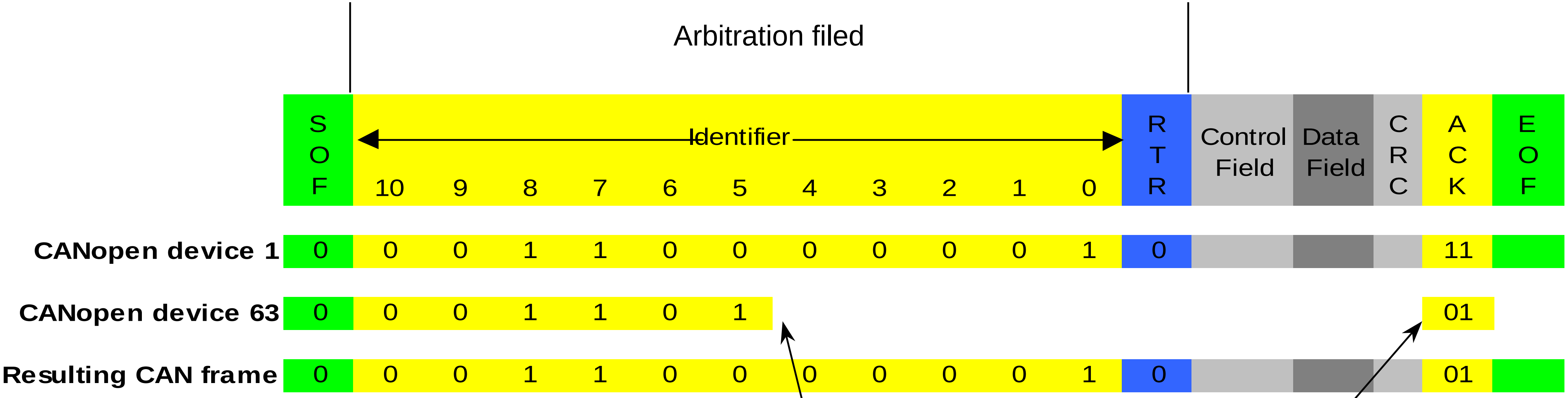
- **During arbitration every node monitors bus line to detect whether its transmitted bit is overwritten by a message of higher priority**
  - → Recessive bit level = 1
  - → Dominant bit level = 0

- **As soon as a transmitting node detects a “dominant” bit while transmitting a “recessive” bit it releases the bus, immediately stops transmission and starts receiving the frame**

Bus Arbitration Principle (2)

- **Within one system each message must be assigned to a unique message identifier**
- **Data frames with a given identifier and a non-zero data length code may be initiated by only one node**
- **Every remote frame should have a system-wide data length code**
  - → Otherwise overwriting of data or data length code
    - – bit errors until bus-off of node(s)

Bus Arbitration Principle (3)



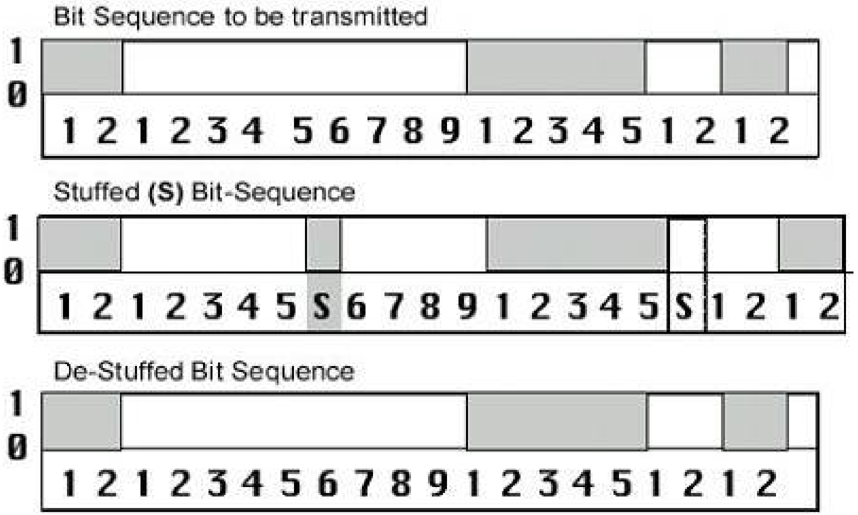
	SOF	10	9	8	7	6	5	4	3	2	1	0	RTR	Control Field	Data Field	CRC	ACK	EOF
CANopen device 1	0	0	0	1	1	0	0	0	0	0	0	1	0				11	
CANopen device 63	0	0	0	1	1	0	1										01	
Resulting CAN frame	0	0	0	1	1	0	0	0	0	0	0	1	0				01	

Node 63 losing arbitration
and stops transmitting!

Node 63 still ACKs message.

Bit Stuffing Principle

- CAN will detect an error after 6 consecutive bits of the same value. Therefore a so called “Stuff Bit” will be inserted in the frame after 5 consecutive bits.
- The CRC, ACK and EOF are of fixed form and will be not stuffed.
- The receiver will automatically remove the “Stuff Bits” from the frame



Frame Types

- **DATA Frame**
  - → Used for sending normal messages
- **ERROR Frame**
  - → ERROR ACTIVE frame:
    - – Six consecutive dominant bits
  - → ERROR PASSIVE frame:
    - – Six consecutive recessive bits
- **REMOTE Frame**
  - → Used to request a DATA Frame with the same Identifier
- **OVERLOAD Frame**
  - → Used for flow control purposes
    - – provides a delay between the transmission of frames

Acknowledgments

- **ALL nodes check all messages for validity**
  - → Each node will acknowledge valid messages in the ACK Slot
    - – this indicates to the sending node that at least one node has received its message correctly
  - → Each node will flag invalid messages with an error frame
    - – this indicates to all nodes that at least one node did not receive the message correctly

- **There is no separate acknowledge frame**

Error handling (1)

- **CAN defines a Error State Machine with three error states**
  - → Error Active
  - → Error Passive
  - → Bus Off

- **Reaction to errors is different in each state**

- **Thresholds are defined by CAN and are part of the CAN chip**

Error handling (2)

- **Error counters track Tx and Rx errors**
  - → Transmit errors count more heavily than receive errors.
- **Good messages will decrement error counters**
  - → Accounts for temporary disturbances
  - → Allows transitions to/from Error Active and Passive states
  - → CAN does not allow a transition from the BUS OFF state
  - → Once the BUS OFF state is reached, the node normally requires a power cycle to return to any other state.

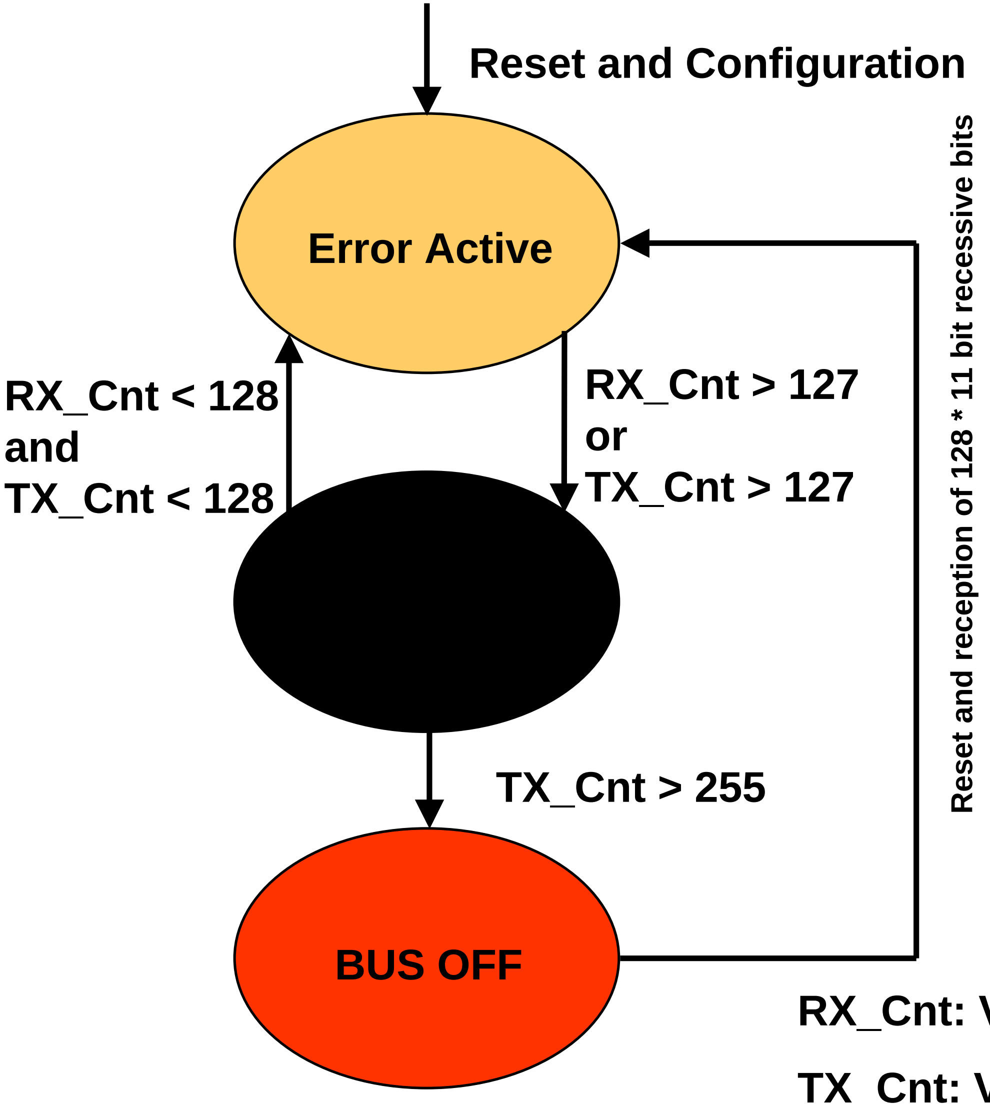
Error State (1)

- **Error Active**
  - Assumption is that network errors are not this node’s fault
  - Upon detection of an error:
    - node will immediately transmit an ERROR ACTIVE FRAME
    - causes all nodes to abort the current message
- **Error Passive**
  - Assumption is that error may be this node’s fault
  - Upon detection of an error:
    - node will immediately transmit an ERROR PASSIVE FRAME
    - this will not affect other nodes unless the node that detects the error was the transmitting node (ie: this was a bit error)

Error State (2)

- **BUS OFF**
  - Assumption is that this node is faulty
  - This node is not allowed access to the network
  - Normally the node needs to be power cycled
    - Exception (normally not implemented but mentioned here for completeness): A node which is ‘bus off’ is permitted to become ‘error active’ (no longer bus off) with its error counters set to zero (0) after 128 occurrence of 11 consecutive ‘recessive’ bits have been monitored on the bus

Error State (3)



RX_Cnt: Value Receive Error Counter

TX_Cnt: Value Transmit Error Counter

- **The value of two error counters determine a node’s error handling status**
- **Increment/Decrement of error counters according to sophisticated rules, e.g.**
  - → Increment error counter by 8 on a given error
  - → Decrement error counter by 1 with each successful operation.

Error detection

Facilities

- **Bit Errors**
  - → Transmitting node checks bit on bus versus what it sent, and finds it to be different
- **Stuff Error**
  - → Occurs after the 6th consecutive bit of the same value (See Error Flags)
- **Acknowledgment Error**
  - → Transmitting node did not detect a dominant bit value in the Ack Slot
- **CRC Error**
  - → 16 bit value recalculated by receiving node did not match transmitted value
- **Form Error**
  - → delimiter and other packet format violations

Capability

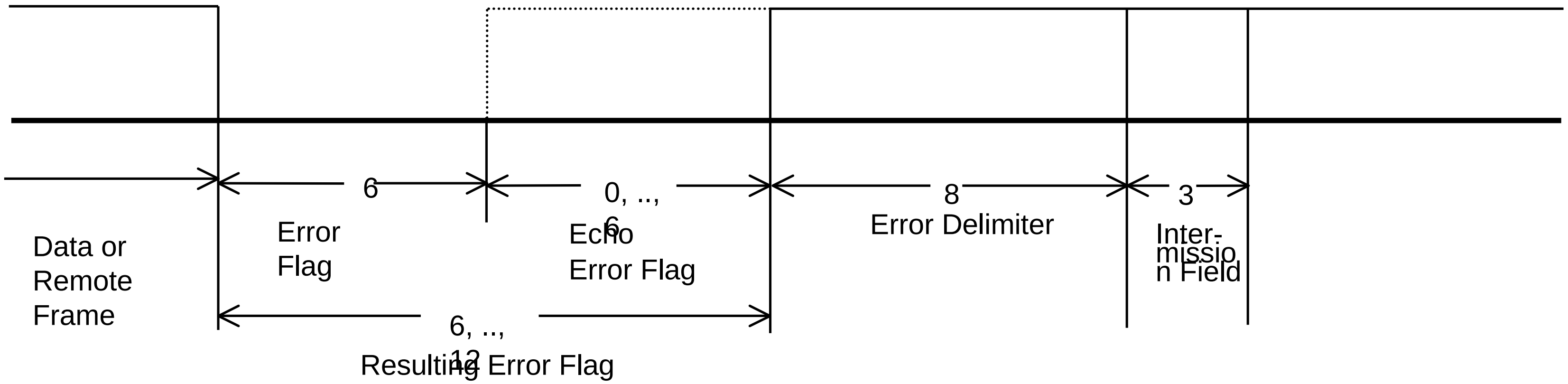
- **Resulting “Error Detection Capability” measured by probability for non-detection of a disturbed message:**
  - – <10-10 * Message Error Rate

- **Example:**

• Data Rate	500 Kbit/s
• Bus load	25 %
• Average message length	80 Bit
• Operating time	2000h/ Year
• Average error rate	10^{-3}
• Probability of a message with an undetected error:	10-13
• Mean time between this event:	1000 years

Error Signaling Error Frame

If any node detects an error it starts the transmission of an error frame



Error-detecting node transmits 6-Bit (dominant) error flag at next bit time

[1] Other nodes detect bit-stuffing error within 6th bit of error flag

[2] and start transmission of own error flag.

⇒Destruction of actual message; secure network-wide data consistency.

[1] Not for CRC-error; Transmission starts after following ACK-delimiter bit

[2] Unless an error flag for a previous error condition (form error) has already been started

What you should remember

- Differential signal transmission
- transmission rate up to 1 Mbit/s
- bus length and transmission rate are depending
- max. 127 devices on the network
- max. 8 byte per PDO
- No telegram loss in case of collision
- Multi master capability
- Synchronization of messages possible

What you should remember

- Produce Consumer model
- PDO for process data
  - → several transmission modes possible
- SDO for service data and large data
- Node address influence the priority of the message
  - → see the arbitration method
- Two watchdog mechanisms possible
  - → Heartbeat
  - → Node Guarding
- CANopen is one application layer on CAN

Thanks

for your attention