

CANopen Fundamentals Seminar

Duration:	1 Day
Target group:	Decision makers and developers
Prerequisites:	CAN fundamentals

1 Introduction to the CAN protocol (1,3 h)

Goal: Provide an overview of CAN communication mechanisms. Convey the fundamental properties of CAN and resulting requirements for CAN-based systems

Contents: Physical layer, arbitration, frame formats, bit-stuffing, error handling

2 Introduction to the CANopen world (1,0 h)

Goal: Provide an overview of components and correlations

Contents: Standardization documents, network properties, switching on a CANopen device, general device model, structure of the object dictionary

3 Accessing the Object Dictionary (SDO) (1,5 h)

Goal: Introduce the Service Data Object as a means of configuring CANopen nodes

Contents: Properties of SDO communication, differences between „expedited transfer“, „segmented transfer“ and „block transfer“, error handling in SDO transfer

4 Parameter Handling / Parameter Storing (0,3 h)

Goal: Standardized configuration storing of a CANopen device

Contents: Definition/addressing of single parameter areas, initiation of storing resp. erasing of the parameter areas

5 Exchange of Process Data (PDO) (1,5 h)

Goal: Use of PDOs in a network, communication configurability

Contents: Attributes of a PDO, PDO communication types (synchronous, event-based), configuration of PDOs by entries in the object dictionary

6 Network Management and Error Detection (0,7 h)

Goal: Control and monitor CANopen nodes in the network

Contents: States in the state machine and their properties, node monitoring by Guarding or Heartbeat, Emergency messages

7 CANopen Master and CANopen Slave (0,3 h)

Goal: Differentiation of the terms

Contents: Related base functionalities, extended functionalities, Boot-Up process

8 Conformance Testing (0,3 h)

Goal: Overview of the test tool

Contents: Principles, procedure and coverage of the tests

9 Questions, suggestions, wishes

Goal: Clarification of remaining questions and open discussion as feedback for Vector