SYS TEC ELECTRONIC

sysWORXX Automation Series

sysWORXX CANopen I/O modules

User Manual

Document number:
L-1070e_07

The following supplement is part of this documentation:

none

This page was left empty intentionally.

Safety Guidelines

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol; notices referring to property damage only have no safety alert symbol. These notices shown below are graded according to the degree of danger.

Danger

indicates that death or severe personal injury will result if proper precautions are not taken.

Warning

indicates that death or severe personal injury may result if proper precautions are not taken.

Caution

with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

Caution

without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

Note

indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by qualified personnel. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

Warning

This device may only be used for the applications described in the catalog or the technical description and only in connection with devices or components from other manufacturers, which have been approved or recommended by SYS TEC. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

In this manual are descriptions for copyrighted products, which are not explicitly indicated as such. The absence of the trademark (®) symbol does not infer that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual.

Disclaimer

The information in this document has been carefully checked and is believed to be entirely reliable. However, SYS TEC electronic GmbH assumes no responsibility for any inaccuracies. SYS TEC electronic GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. SYS TEC electronic GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages that might result.

Additionally, SYS TEC electronic GmbH offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. SYS TEC electronic GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

Contact information

Address:	SYS TEC electronic GmbH August-Bebel-Str. 29 D-07973 Greiz GERMANY
Ordering Information:	+49-3661-6279-0 sales@systec-electronic.com
Technical Support:	+49-3661-6279-0 support@systec-electronic.com
Fax:	+49-3661-6279-99
Web Site:	http://www.systec-electronic.com

This page was left empty intentionally.

Preface

Purpose of this manual

The information provided in this manual enables you to operate the distributed I/O modules of the sysWORXX Automation Series.

Basic knowledge required

To understand the manual, you require general experience in the field of automation engineering and a general understanding about CANopen.

Scope of this Manual

This manual is applicable to the devices of the CANopen I/O modules of the sysWORXX Automation Series.

This manual contains a description of the devices, which were valid at the time the manual was published. We reserve the right to issue a Product Information, which contains up-to-date information about new components and new versions of components.

Guide

The manual's navigation features outlined below support quick access to specific information:

- The manual begins with a table of contents and a list of tables.
- Important terms are explained in the glossary.
- Navigate to the most important topics in our documents using the index

Special notes

In addition to this manual, you also might need the manual of the CANopen Master (in general a PLC device) and the manual of the CANopen configuration software you are using.

Recycling and disposal

The modules of the sysWORXX Automation Series can be recycled due to its ecologically compatible equipment. For environmentally compliant recycling and disposal of your electronic waste, please contact a company certified for the disposal of electronic waste.

The sysWORXX I/O devices shipped out after July 1st, 2006 comply with RoHS regulations (Category 9 of the Germany law “Gesetz über das Inverkehrbringen, die Rücknahme und die umweltverträgliche Entsorgung von Elektro- und Elektronikgeräten (Elektro- und Elektronikgerätegesetz – ElektroG)*) Vom 16. März 2005”, Einordnung in die Kategorie 9) issued by the European Union.

Further information

If you have any questions relating to the products described in this manual, and do not find the answers in this documentation, please contact your technical support.

The portal to our technical documentation and support for all SYS TEC products and systems is available at:

http://www.systec-electronic.com/support

Technical Support

You can reach technical support for all SYS TEC products:

Using the Support Request form on the web:
http://www.systec-electronic.com/support

Phone: + 49-3661-6279-0

Fax: + 49-3661-6279-99

For further information about our products and services, please refer to our Homepage at:

http://www.systec-electronic.com

Service & Support on the Internet

There you will find:

- Our Newsletter, which constantly provides you with the latest information about your products.
- The right documentation and latest drivers for use with our products
- A list of our distributors and partners for our products your inquiries

Table of Contents

This page was left empty intentionally.

Index of Tables

Index of Figures

1 General description

1.1 What are distributed I/O systems?

Process I/Os are often installed as a central integral in the automation system configuration. Greater distances between the process I/O and the automation system may require extensive and complex wiring, which could make the system susceptible to electromagnetic interference and thus impair its reliability. Distributed I/O forms the ideal solution for such systems. While the master CPU is located centrally the distributed I/O systems (inputs and outputs, intelligent preprocessing using intelligent CANopen slaves) operate locally at a remote location the highly efficient CANopen protocol and high data transmission rates of the CAN-bus provide a smooth flow of communication between the CPU and the distributed I/O systems

1.2 Main characteristics of CAN

In the following the main features of the CAN protocol as standardized by the ISO 11898-1 and ISO 11898-2 are introduced.

Bus Topology, message rate, and number of nodes

CAN is based on a linear[1] topology usually utilizing a two-wire bus media with differential signal transmission.

Hierarchical network structures are possible using repeaters or routers. The maximum number of nodes is limited by capability of the deployed driver chips, not by the protocol itself.

Repeater can be used to increase the number of nodes on the network. The maximum network extension possible at a specific bit rate is limited by the signal propagation time[2] along the bus medium.

Message-oriented protocol

The CAN protocol is not based on addressing the message receiver, but uses the CAN-identifier for identification of transmitted messages. Based on the CAN-identifier, each node checks whether the received message is relevant for itself. Therefore, a message can be received and accepted by one or multiple nodes at the same time (broadcasting).

Priority of messages, Short latency time for high-priority messages

The CAN-identifier of a CAN message directly represents its priority with regards to bus access. This allows for preferential transmission of

[1] also known as “Bus Topology”

[2] At 1MBit/s a network length of 40m is possible. At 80 kBit/s up to 1000m bus length is possible

important messages with a low latency time regardless of the actual busload; even in exceptional situations (transmission peaks or disturbances) the transmission is ensured.

Multi-master capability

On CAN, bus access does not depend on a supervisory control unit. Each node can start transmitting a message as soon as the bus becomes idle. In case of simultaneously access of several nodes, the node that wants to transmit the message with highest priority obtains access to the bus.

Transmissions are initiated by the message source. Thus, the bus is occupied only if new messages are to be sent (event controlled transmission). This results in a significant lower average busload in comparison to a system with deterministic bus access.

Loss-free bus arbitration

The CAN protocol uses the CSMA/CA[1] access method to guarantee the transmission of the highest prior message in case of simultaneous access attempt without destruction.

Short frame length

The maximum data length of a CAN message is limited to 8 bytes to guarantee a short latency time for bus access. Short messages are important to increase reliability of transmission in a distorded environment, as the probability of a coincidence with a disturbance increases proportionally with the frame length.

Transmission of data with size higher than 8 bytes is handled by services provided with the higher layer protocol such as the SDO[2] in CANopen.

High data integrity and very short recovery time

The CAN protocol features several complementary mechanisms for detection of corrupted messages with a very high probability including automatic re-transmission of incorrectly transmitted or received messages. Unlike node-oriented protocols, CAN provides a very short error detection, signaling and correction time.

Network wide data consistency

A system wide data consistency is fundamental for data integrity in distributed systems. In process control applications the operation of several nodes need to be synchronized frequently. This requires the data and synchronization messages to be received correctly and

[1] Carrier Sense Multiple Access / Collision Avoidance

[2] Service Data Object

simultaneously by involved nodes. Thus, locally disturbed messages must be known to be invalid by all nodes. The error signaling mechanism defined within the CAN protocol provides this basic requirement.

Detection and de-activation of defective nodes

Within the CAN protocol a monitoring of the communication-specific functions is defined. If a node exceeds pre-defined error rates, measures are taken to prevent defective nodes from continuously disturbing the data communication.

International standardization

The international standards ISO 11898 Part 1, 2 and 3 specify CAN as OSI-Layer 1/2 protocol. As a higher layer protocol for general industrial application, CANopen was specified by the CAN in Automation (CiA) and applied as European standard EN 50325-4.

1.3 What is CANopen?

What is CANopen?

CANopen is a standardized CAN-based protocol for industrial distributed automation systems. In Europe CANopen can be regarded as the de-facto standard for implementation of industrial CAN-based systems.

In 1995, the CANopen specification was handed over to the CAN in Automation (CiA) international users' and manufacturers' group and is now standardized as CENELEC EN 50325-4.

CANopen offers the following performance features:

- Transmission of time-critical process data (*see Section 8.2*) according to the producer consumer principle
- Standardized device description (data, parameters, functions, programs) in the form of the so-called "object dictionary". Access to all "objects" of a device with standardized transmission protocol (SDO protocol) according to the client-server principle. (*See Section 8.2*)
- Standardized services for device monitoring (node guarding/heartbeat), network management ("NMT messages , boot-up messages") and error control (Emergency messages) (*see Section 8.2, 9.5*)
- Standardized system services for synchronous operations (SYNC messages), central time stamp message (*see Section 8.2, 8.6*)
- Standardized functions for remote configuring of bit rate and device identification number via the bus (*see Section 5.3*)

- Standardized CAN identifier assignments based on the node-ID simplify the system configurations in the form of the so-called "predefined connection set"

The Object Dictionary concept

The central element of the CANopen standard is the description of all device-specific functionality, parameters and data-types by an “Object Dictionary” (OD). Thereby, the Object Dictionary can be seen as a lookup table with a 16-bit Index and an 8-bit Subindex. This allows for up to 256 Subentries per Index. Each entry can hold one variable of any type (including a complex structure) and length. In the following sections the terms Object and Subindex will be used when describing such Object Dictionary entries.

All process and communication related information is stored as entries in predefined locations of the Object Dictionary. Therefore the Object Dictionary is divided in several sections containing general specifications about the device such as identification data and manufacturer, a section containing communication parameters, and a section with device specific functionality. All entries of the Object Dictionary are accessible from the “outside” via CAN using SDO communication (*see Section 8.2*). Therefore, a CANopen device is completely remote configurable, which provides the basis for the manufacturer independence targeted by CANopen.

CANopen profiles

CANopen is based on a so-called “communication profile” that specifies basic communication mechanisms and services (CiA 301). Further profiles and frameworks exist, specifying extended functionality for use with programmable devices (CiA 302) or safety relevant communication (CiA 304).

In addition to the communication profiles there are so-called “device profiles” for important types of industrial devices, such as generic digital and analog I/O devices (CiA 401), drives (CiA 402), IEC 61131-3 programmable devices (CiA 405) or encoders (CiA 406). The device profiles are add-on specifications that describe all the communication parameters, device-specific features and Object Dictionary entries that are supported by a certain type of CANopen module. A master or configuration tool can read-access the identity object (*see Section 8.5*) of any slave node to receive the information about which device profile a module conforms to.

Sometimes an application requires the implementation of not standardized, manufacturer-specific Object Dictionary entries. This is possible due to the open structure of CANopen. Additional entries that disable or enable a certain functionality that is not covered by one of the existing device profiles can be implemented in any device, as long

as they conform to the structural layout of the Object Dictionary (*see Section 8.4*).

Electronic Data Sheets

In addition, the functionalities and characteristics of each CANopen device are described in a so-called “Electronic Data Sheet” (EDS) stored in ASCII or XML format (CiA 306). The EDS offers a standardized way of specifying supported Object Dictionary entries and can be seen as a template for describing the device configuration. The actual device configuration is stored in a so-called “Device Configuration File” (DCF) and, for example, contains the resolved communication and mapping parameters for process data communication (*see Section 5.4*).

A CANopen master or configuration tool can directly load the EDS into its set of recognized devices. Once the device was found on the network, all supported Object Dictionary entries are known by the master or configuration tool.

The Device Profile specifies the minimum entries that need to be supported by a device conforming to the profile. However, the EDS might only specify objects that are specific to a certain manufacturer or sub-type of module.

Device Profiles and Electronic Data Sheets are the basic functionality needed to meet the requirement for "off-the-shelf' availability of network devices. From the communication point of view, any two nodes that conform to the same EDS are interchangeable, their Object Dictionaries are identical and they have the same communication behavior.

What are CANopen Slaves, CANopen Masters and CANopen Managers?

Within a distributed system the application process is divided into several parts running on different nodes. From the applications point of view usually one node is responsible for the control and management of the distributed control system. This node (e.g. a PLC) is called application master. CANopen devices without management functions are generally supposed to be CANopen slave devices (e.g. I/O modules).

However, it is possible to operate the sysWORXX I/O modules without having a master on the network. Therefore the sysWORXX I/O modules feature a so-called “simple boot-up NMT master”[1] implemented as manufacturer specific extension. See *Section 8.4* for more information.

[1] The device can switch to NMT state OPERATIONAL autonomously and sends out the corresponding NMT message to start other devices as well (*see Section 9.5*).

The term CANopen Manager is used to specify more clearly the network functionality of a network-controlling device in a CANopen network:

- Definition of the Boot-Up process for each device that is to be managed.
- Configuration of unconfigured nodes during system boot-up.
- The dynamic establishment of SDO connections between devices. The SDO Manager handles dynamic SDO connections.
- The definition of dynamically allocated entries (Network Variables) in an object dictionary which can be used for the representation of I/O data e.g. on programmable nodes like PLCs.
- Provides services for downloading program data and functions for the control of programs on a device.

The PLC devices of the sysWORXX Automation Series provide CANopen Manager functionality and therefore can be used as Application Master for your CANopen network. Please refer to Section 1.4 for more information about the sysWORXX Automation Series.

Which devices can be connected to a CANopen network?

CANopen devices that at least comply with the CANopen specification CiA 301 can be connected to a CANopen network.

Furthermore, all devices connected to the same CAN-bus segment have to support the same physical layer and an identical bit rate.

All sysWORXX I/O modules support high-speed CAN according to ISO 11898-2. Coupler devices and gateways may be used to extend the network size or to connect CAN-bus segments with different physical layer and/or bit rate.

Within a CANopen network each device has a so-called “node-ID”, which is used to identify a specific node. The valid range for node-IDs is from 1 to 127.

By this schema, a CANopen network can have 127 nodes theoretically. Practically, this number is limited by the CAN transceivers used, which typically support up to 100 nodes on the same bus. The CAN transceivers used on the sysWORXX I/O modules support up to 110 nodes.

To put a CANopen network into operation, two basic conditions must be fulfilled:

- All nodes must be configured to the same bit rate and
- The assigned node-IDs are unique.

The system integrator needs to ensure these conditions are fulfilled, as there are no off-shelf mechanisms that can ensure this conditions automatically.

Usually the node-ID is configured directly on the device via hardware switches (*see Section 5.2*). Alternatively, the node-ID and bit rate can be configured via the so-called "Layer Setting Service" (LSS). Please refer to *Section 5.3* for detailed information.

Configuration of a CANopen network

The figure below illustrates the typical configuration of a CANopen network. The CANopen masters are integrated in the corresponding device. CANopen slaves form the distributed I/O systems, which are connected to the CANopen masters via CAN-bus.

Figure 1: Simple CANopen network configuration

The PLCmodule-C14, for example, features two CAN-bus interfaces, which allows for connecting the PLC to two different CANopen networks.

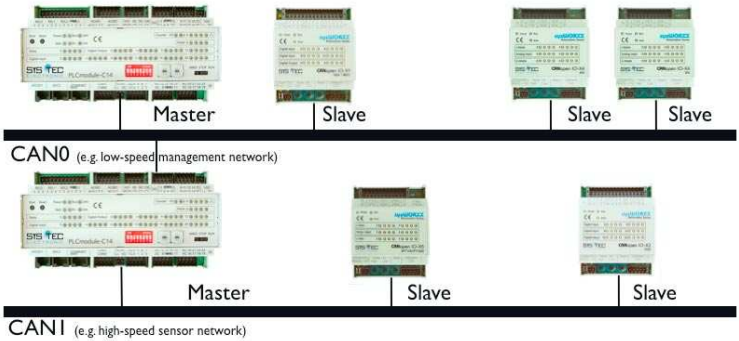


Figure 2: Complex CANopen network configuration

1.4 The sysWORXX Automation Series

The sysWORXX Automation Series combines harmonized devices and software tools, needed to create sophisticated industrial distributed automation solutions.

The sysWORXX Automation Series includes:

- IEC 61131-3 compliant controls
- CANopen I/O modules
- CANopen Human Machine Interfaces (HMI)
- Integrated IEC 61131-3 Development System
- CANopen Configuration Tools
- CAN-bus interfaces and gateways

IEC 61131-3 controls:

PLCmodule-C14 and CANopen PLC

The sysWORXX PLC modules are high-performance and versatile compact PLCs. They have a number of communications interfaces and a large selection of industry-proven inputs and outputs.

Feature	PLCmodule-C14	CANopen PLC
Order No.	phyPS-412-Z5	3000001
CAN-bus interface according to ISO 11898-2	2, galvanic isolated, each can be operating in CANopen Master or Slave mode	1, galvanic isolated, can be operating in CANopen Master or Slave mode
RS232	3	2
Ethernet	10baseT Ethernet interface for uplink to management PC (for program download, monitoring)	
Digital Inputs	24, isolated, 24VDC	24, 24VDC
Digital Outputs	16, isolated, 24VDC,500mA, high-side switches	16, 24VDC,500mA, low-side switches
PWM/PTO Outputs	2, isolated, 24VDC, 500mA, 15kHz	2, 24VDC, 500mA, 70kHz
Analog Inputs	4 channels, 0..10V, 10-bit	4 channels 0..10V and 4..20mA 12-bit or 14-bit
Counter/Encoder Inputs	3 counter (pulse/dir), isolated, 24VDC, 70kHz	2 encoder, a/b and pulse/dir, 24VDC, 70kHz 1 counter, 24VDC, 70kHz
Relay outputs	4 channels, 230VAC/3A, NO	4 channels, 230VAC/3A, NO
Power supply	24VDC	

Integrated IEC 61131-3 development environment

OpenPCS is an comprehensive IEC 61131-3 workbench certified by PLCopen.

- PLCopen certified IEC 61131-3 compiler
- Sequential Function Charts (SFC)
- Continuous Function Charts (CFC)
- Ladder Diagrams (LD)
- Structured Text (ST) and Instruction List (IL) for function block programming (IL&ST Base Level certified)
- Function block libraries for configuration of sysWORXX I/O modules
- Comprehensive CANopen function block library
- Extended function library (e.g. RTC, non-volatile memory, process control, string manipulations)
- Supports multiple controllers in one project file
- Complete support of CANopen Network variables for data exchange
- Online monitoring and power-flow for easy troubleshooting
- Online change for fast debugging cycles
- Offline simulator
- Integrated OPC Server
- Device access possible via Ethernet, USB, Parallel Port or RS232
- Project documentation support

CANopen configuration tools

The CANopen Configuration Suite is a powerful, intuitive and user-extendable tool chain for configuration and management of CANopen networks and devices. It includes the CANopen Configuration Manager, the CANopen Device Monitor as well as the SYS TEC CAN-driver. Various CAN-bus interfaces of different manufactures are supported. By using the CANopen Configuration Suite your workload will be reduced significantly. Especially when the system becomes more complex. The risk of configuration errors is minimized, and the quality and reliability of the system is enhanced.

- Project-oriented management of all configuration and device data based on EDS and DCF
- Supports download of the device and network configuration via CAN-bus

- Simple and intuitive interface for all configuration tasks in your network
- Quick access to the device parameters and network structure
- Automatic PDO mapping and PDO linking
- Optional PDO linking based on Pre-Defined Connection Set – thus it is not necessary to configure every CANopen slave device
- Strip-chart visualization of PDO data
- Selective readout of the object directory from connected CANopen nodes
- Automatic scanning for CANopen nodes in the network
- Support of network variables in accordance with the CiA 302 and CiA 405 specification
- Export of all assigned network variables as IEC 61131-3 conformant variable declaration
- Script functionality with comprehensive high-level CANopen API for easy realization of automated processes and extension of functionality
- Expert console window for quick command line access to the CANopen functionality

CAN-bus interfaces and gateways

USB-CANmodul1

- Low-cost USB2.0/CAN interface in table case
- PC driver supports up to 64 devices simultaneously
- Power via USB

USB-CANmodul2

- USB2.0/CAN interface in table case
- 2 CAN interfaces, optional one LIN possible
- PC driver supports up to 64 devices simultaneously
- High-precision transmission timer
- 8-bit I/O port (TTL-level)
- Power via USB

USB-CANlog

- CAN-bus data logger with USB/CAN interface
- Stand-alone operation
- 2 CAN interfaces
- Supports SD-cards up to 1GB size
- Selective Triggers and message filters
- External power-supply 9…30VDC

CAN-Ethernet Gateway

- Supports all higher-layer CAN protocols
- Up to 4 connections per device
- High-speed transmission of CAN messages (bulk)
- High-precision timestamps for CAN frames
- ASCII based device configuration (Telnet or RS232)
- Configurable message filters

1.5 CANopen I/O modules

Definition of the sysWORXX CANopen I/O modules

The sysWORXX CANopen I/O modules are compact distributed I/O devices, with degree of protection IP20.

Fields of application

Its compact design and its high I/O density make the sysWORXX CANopen I/O devices suitable for use in machine automation. With IP20, the sysWORXX CANopen IO-X devices are protected against the ingress of foreign particles greater diameter 12.5mm. The sysWORXX CANopen I/O modules support communication with other CANopen devices, which are compatible to CiA 301 and/or CiA 302 standard.

CANopen I/O modules overview:

The following sysWORXX I/O devices are offered:

Name	Description	Order number
CANopen IO-X1	8 DO and 16 DI, 24VDC	3001000
CANopen IO-X1	8 DO (pulsed) and 16 DI, 24VDC	3001010
CANopen IO-X2	24 DI, 24VDC	3001001

Name	Description	Order number
CANopen IO-X3	24 DO, 24VDC 500mA	3001002
CANopen IO-X4	8 AI, 12-bit ADC	3001003
CANopen IO-X5	8 RTD, 12-bit ADC	3001004
CANopen IO-X6	8 AO, 10-bit DAC	3001005
CANopen IO-X7	8 TC, 12-bit	3001006

Table 1: CANopen I/O module overview

Installation

The sysWORXX I/O modules were designed for DIN-rail mounting and always include the complete set of terminal plugs.

You can thus set the focus of your configuration on local requirements. The comfortable handling features of the sysWORXX I/O modules ensure quick commissioning and easy maintenance.

1.6 Components of the sysWORXX CANopen I/O modules

The list below introduces the vital parts and components delivered with the sysWORXX I/O modules:

Component	Function	
I/O device	The I/O devices incorporates the device electronics, LED interface and socket connectors. The following subsystems are included: – Embedded microcontroller – Reset & watchdog circuit – CAN-bus interface – Configuration units – Non-volatile memory for storage of configuration data – Device specific I/O circuitry – Voltage regulator for 24VDC power supply – LED interface	
I/O plug	Removable spring-type plug	

Component	Function	
connector	connector, used to connect the sensors and actuators. The connector block can be fixed on the socked by screw. There are 3 types used: 30-pin for IO-X1 to IO-X3 24-pin for IO-X4 to IO-X6 16-pin for IO-X7	
CAN-bus connector	Removable 5-pin screw-type plug connector, used to connect the CAN-bus lines. The connector pinout complies with DS 102.	
Power supply connector	Removable 2-pin screw-type connector to connect power supply.	

Table 2: Component overview

1.7 HMI elements on the sysWORXX I/O modules

Introduction

This section describes the HMI elements on the sysWORXX I/O modules.

Overview

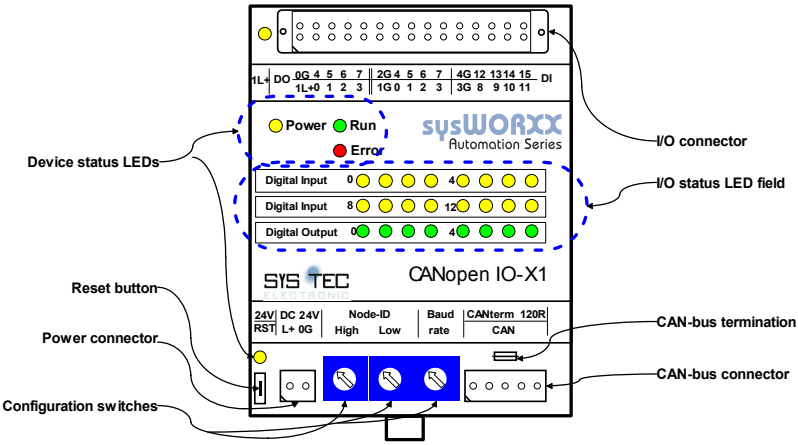


Figure 3: Overview of HMI elements on the sysWORXX I/O modules

2 Application planning

2.1 Compact system design

A compact system design in the context of the sysWORXX I/O modules means: You can adapt the configuration to meet the requirements of your application by means of combining various I/O modules by CAN-bus.

2.2 Selection guide for I/O modules

Help for the selection of I/O modules

The table below helps you to select the right I/O module for different applications:

Application	I/O module	
Evaluating signals of switches, proximity switches, digital sensors	16 DI, 24V DC	CANopen IO-X1
	24 DI, 24V DC	CANopen IO-X2
Switching solenoid valves, DC-contactors, signal lamps	8 DO, 24V DC, 0.5 A	CANopen IO-X1
	24 DO, 24V DC, 0.5 A	CANopen IO-X3
Switching/piloting proportional valves, servo drives, proportional actuators	8 AO 0…10 V; 4…20 mA; 0…20 mA	CANopen IO-X6
Voltage measurement	8 AI, ±10 V; 0…10 V	CANopen IO-X4
Current measurement	4 AI 4…20 mA; 0…20 mA	CANopen IO-X4
Measuring low to medium temperatures	8 RTD PT100, PT1000	CANopen IO-X5
Measuring medium to high temperatures (up to 1870°C)	8 thermo-couple sensors of type J, K, L, R, S, T, E	CANopen IO-X7

Table 3: Selection Guide for I/O modules

2.3 Maximum configuration

Maximum number of nodes on the CAN-bus

The CAN-bus drivers used on the sysWORXX I/O modules support up to 110 nodes simultaneously connected to the same CAN-bus segment.

Electrical maximum configuration

Electronic supply L+:

Supplies power to the internal electronic circuit of the modules. Supplies the digital outputs on CANopen IO-X1 and CANopen IO-X3.

There are additional connection points (1L+, 2L+, 3L+, ect.) to supply power to the outputs.

Warning

If the digital outputs are connected to the process, the additional supply points for power (1L+ … 3L+) must be connected. Otherwise a shortcut might lead to damages on the PCB or power supply connector.

On sysWORXX I/O modules without isolated CAN-bus, L+ is connected to the supply lines of the CAN-bus (CAN_V+).

Properties	Limitations
Electronics supply L+	1A (if additional supply points for outputs 1L+...3L+ are not used)
Digital supply 1L+ … 3L+	4 A each supply point

Table 4: Electrical maximum configuration

Mechanical maximum configuration

The I/O connector used supports connection of cables up to a diameter of 1mm^2. Do not connect more than one cable to a single I/O point. It is recommended to use flexible cable types for wiring to the terminal block.

3 Mounting

3.1 Requirements

Pre-assembly

You can pre-assemble the modules on a DIN-rail before you install it on site.

Mounting position

The modules can be installed in any mounting position.

Mounting dimensions

See also Appendix Module Dimensions on page 187.

Mounting dimensions	Comment	Dimension
Mounting width		71 mm
Mounting height	with I/O terminal block assembled	96 mm
	without I/O terminal block assembled	94.8 mm
Mounting depth	starting from DIN-rail	53.58 mm

Table 5: Mounting dimensions

3.2 Installing the I/O modules

Introduction

The module features removable terminal blocks to connect to the CAN-bus, I/O wiring and power-supply. The module can be installed without terminal blocks assembled.

Requirements

If the devices are mounted on a DIN-rail, it must be mounted on the rack or solid surface.

Required tools

Slotted screwdriver with 4 mm blade.

Procedure

(1) Place the module onto the DIN-rail as shown below. Use a slotted screwdriver to lift the lug (1), and then push it on until it engages with an audible click.

(2) Slide it into the working position.

Caution

Do not twist the screwdriver while it is placed in the lug. It will lead to the destruction of the lug.

Do not use the enclosure as a support point for the screwdriver. Otherwise the hardware switches or the enclosure might be damaged.

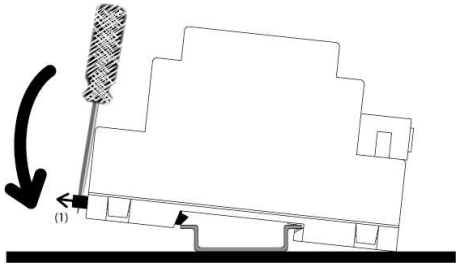


Figure 4: Installation of the sysWORXX I/O modules

See also

Section 6, Commissioning the sysWORXX I/O modules at page 51

3.3 Installing the wiring to the connectors

Introduction

The module has two kinds of removable terminal blocks:

(1) Spring-clamp type for I/O and CAN-bus

(2) Screw-type for power-supply

Requirements

Before you wire any of the modules, either switch off power or remove the relevant connector terminal blocks.

Required tools

Slotted screwdriver with 2,5 mm blade.

Procedure for placing the connector

(1) Place the connector on the socket then push it on until it engages with an audible click.

(2) If necessary fix the I/O connector by the two screws located on both sides.

Procedure for placing the wires

Spring-clamp type (*see Figure 5*)

(1) Push the screwdriver into the rectangular hole of the I/O point you intend to wire. Make sure not to exceed the physical dimensions of the connection point.

(2) Insert the cable end and remove the screwdriver.

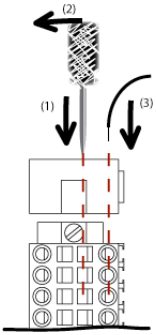


Figure 5: Handling of spring-type connectors

Screw type (*see Figure 6*)

(1) Open the screw and insert the cable.

(2) Close the screw.

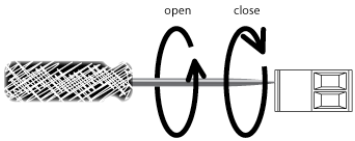


Figure 6: Handling of screw-type connectors

See also

Section 6, Commissioning the sysWORXX I/O modules at page 51

3.4 Setting the CANopen node-ID, bit rate and the terminating resistor

Introduction

After mounting the module you need to set the CANopen node-ID and terminating resistor at the module.

- The CANopen node-ID defines the address of the node within the CANopen network. The node-ID directly represents the message priority of this particular node.
- A CAN-bus segment must be terminated at both ends, i.e. on the first and last segment node, with its characteristic impedance. Enable the integrated terminating resistor if the device is the last node on the CAN-bus (*see Section 4.3*).

Requirements

The set node-ID must correspond with the definition in the Device Configuration File of this device.

Configure the node-ID and bit rate before you power-on the module. If you change while the module is powered-on, the changes become effective after reset or on next power-on.

Required tools

Screwdriver with 2,5 mm blade

Setting the CANopen node-ID and CAN-bus bit rate

(1) Set the node-ID using the hex-encoding switches (see the example below).

(2) Set the CAN-bus bit rate using the hex-encoding switches (see the example below).

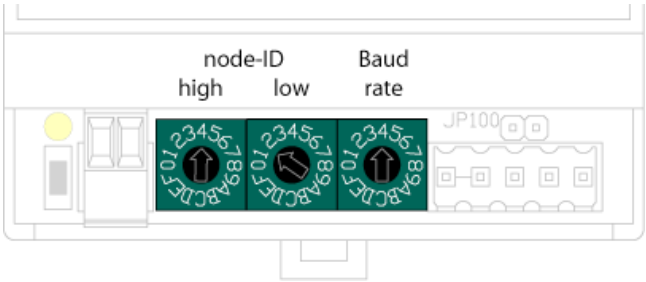


Figure 7: Location of configuration switches

For further information on how to set the node-ID and bit rate refer to *Section 5.2*.

Enabling the terminating resistor

If this device is the first or last node on the CAN-bus, enable the internal terminating resistor.

You need to remove the CAN-bus terminal connector before you can set or remove the jumper.

(1) To enable the termination set the jumper.

(2) To disable the termination remove the jumper.

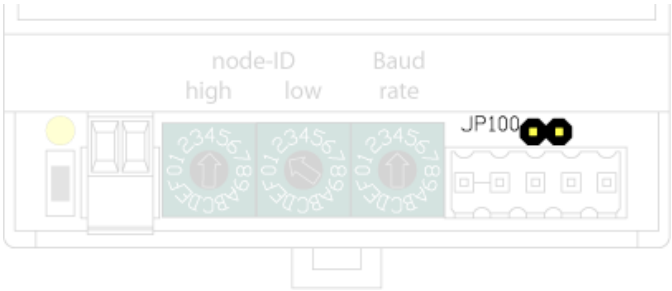


Figure 8: CAN-bus termination jumper

See also

Section 4.3, Connecting the CAN-bus

Section 5.1, General rules for configuring CANopen networks

This side was left empty intentionally.

4 Connecting

4.1 General rules and regulations for operating the sysWORXX I/O modules

Introduction

The distributed I/O modules represent a component of plants or automated systems, and thus is subject to special rules and regulations based on its application.

This section provides an overview of the most important rules you have to observe when integrating the sysWORXX I/O modules into a plant or system.

Specific application

Observe the safety and accident prevention regulations for specific applications, for example, the machine protection directives.

EMERGENCY-OFF equipment/components

EMERGENCY-OFF equipment must remain effective in all operating states of the plant or system.

System startup after certain events

The table below shows what you have to observe when restarting a plant or system as a result of specific events.

If …	then …
there was a restart following a voltage drop or power-fail	dangerous operating states must not develop. If necessary, force an “EMERGENCY STOP”.
there was a startup after interruption of bus communication.	the system must never perform an uncontrolled or undefined restart.

Table 6: System startup after certain events

24VDC power supply

The table below shows essential aspects of the 24VDC power supply.

For …	you need to observe …	
buildings	external lightning protection	lightning protection precautions (e.g. lightning protection elements)
24VDC power supply cables and signal cables	internal lightning protection	
24VDC power supply	safe (electrical) isolation of the safety extra-low voltage (SELV)	
Daisy-chaining the power supply	voltage drop when daisy-chaining the power supply.	

Table 7: Considerations for 24VDC power supply

Protection from external electrical interference

The table below shows how to protect your system against electromagnetic interference or faults.

For …	make sure …
all systems or plants that contain a sysWORXX I/O device	that the system is properly grounded in order to allow the EMC-conformant discharge of electromagnetic interference.
power supply, signal cables and bus cables	that the cables are properly routed and the installation is free of faults.
signal and bus cables	that cable or wire break does not lead to undefined states of the system.

Table 8: Protection from external electrical interference

4.2 Operation of sysWORXX I/O modules on grounded reference potential

Introduction

This section provides information on the overall configuration of a sysWORXX I/O module on a grounded TN-S power supply. The following topics are covered:

- Disconnecting devices, short-circuit and overload protection according to VDE 0100 and VDE 0113
- Load power supplies and load circuits

Grounded mains

The neutral of grounded mains is always bonded to ground. A short-circuit of a live conductor or grounded part of the system to ground trips the protective devices.

Safe electrical isolation (SELV/PELV to IEC 60364-4-41)

The sysWORXX I/O modules require power supplies or power supply modules with safe electrical isolation.

Installation with grounded reference potential

in an system with grounded reference potential, any interference current is discharged to protective earth. The terminals need to be interconnected externally (G ↔ PE.)

Components and protective measures

Regulations stipulate the implementation of diverse components and protective measures when installing the plant. The type of components and the binding character of protective measures depends on the DIN regulation which applies to your application. The table refers to *Figure 9* below.

For …	Reference to Figure 9	DIN VDE 0100	DIN VDE 0113
Disconnecting devices for control systems, signal generators and final control elements	①	Part 460: main switch	Part 1: mains disconnect switch

For …	Reference to Figure 9	DIN VDE 0100	DIN VDE 0113
Short-circuit and overload protection	②	Part 725: Single-pole fusing of circuits	Part 1: grounded secondary power circuit: single-pole fusing
Line protection	⑤	Part 430: Protection of cables and lines against over-current	

Table 9: Protective measures

Overall configuration of a sysWORXX I/O device

The *Figure 9* below shows the overall configuration of a sysWORXX I/O device (load voltage supply and grounding concept) which is operated on TN-S mains.

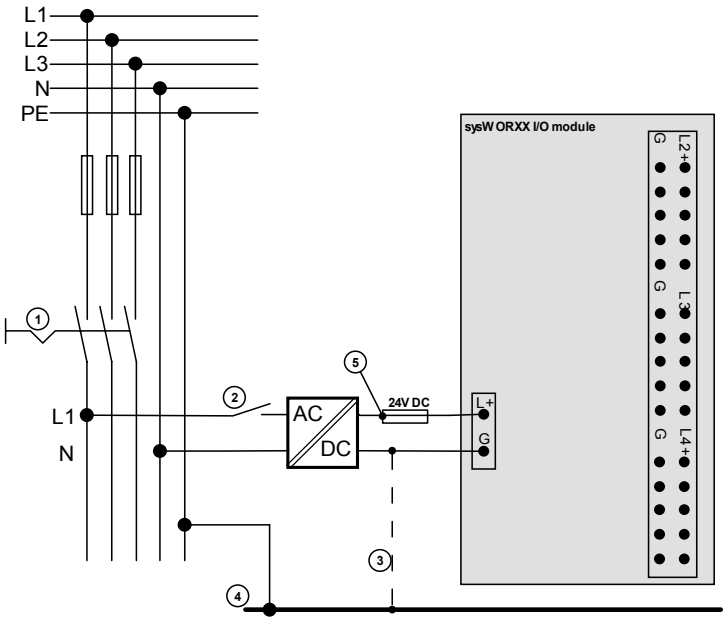


Figure 9: Electrical configuration of the sysWORXX I/O modules

① To disconnect devices for the control system, signal generators and final control elements

② For short-circuit and overload protection

③ The connection between G and PE is not applicable in a configuration with ungrounded reference potential.

④ Grounding bus bar

⑤ Fuses for line protection

4.3 Connecting the CAN-bus

Introduction

This section provides an overview on how the sysWORXX I/O modules are connected to the CAN-bus and gives hints for wiring and cabling the bus.

Wiring and cabling

Although CAN is supposed to be a 2-wire network an additional common ground is required for reliable operation, especially if the network spreads over a longer distance.

CAN-bus cable

Using screened twisted-pair cables (2x2x0,25mm^2) with a characteristic impedance of between 108 and 132Ω is recommended for the CAN-bus wiring.

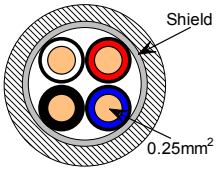


Figure 10: CAN-bus cable cross-view

SYS TEC electronic has high quality CAN-cables in its scope of delivery. Please contact our sales for a proper quotation.

CAN-bus interface connector pinout on the sysWORXX I/O modules

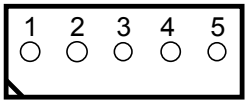


Figure 11: CAN-bus interface connector pinout

Pin	CAN-bus signal / Description
1	CAN_GND
2	CAN_L
3	n.c.
4	CAN_H
5	CAN_V+ (connected to L+ on modules without galvanic isolation, not used on modules with galvanic isolated CAN)

Table 10: CAN-bus interface connector pinout

CAN-bus signal description

Signal	Description
CAN_L	Bus line that is driven lower during the dominant bus state.
CAN_H	Bus line that is driven higher during the dominant bus state.
CAN_GND	This is the common ground used by the CAN nodes. This might not be needed if the nodes have a common ground anyway.
CAN_SHLD	Optional shield around CAN_L and CAN_H (not used on sysWORXX I/O modules)
CAN_V+	If a CAN node is supplied with its operating power via the CAN cable this line is connected to the positive line of the power supply. The voltage levels are not specified and depend on the application. For sysWORXX I/O modules, the used voltage should be 24V DC. **Note** The maximum current should not exceed the specified limit of the cable used.

Figure 12: CAN-bus signal description

The wiring schema of a sysWORXX I/O device is shown in *Figure 13*.

Note

For reliable operation CAN_L, CAN_H and CAN_GND must be wired.

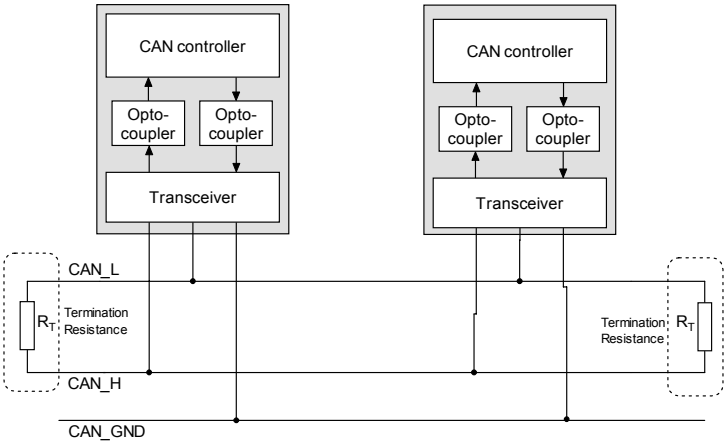


Figure 13: Wiring schema of galvanic isolated sysWORXX I/O devices

Physical layout

Typically the layout of a CANopen network is that of a linear bus. The main trunk consisting of the CAN_L and CAN_H signals must have termination resistors (typically 120Ω) **at each end of the line**. Please refer to *Section* ***Fehler! Verweisquelle konnte nicht gefunden werden.*** for more detailed information about cable length and termination resistance depending on the bit rate.

If Y-junctions are used, the drop lines (aka Trunk lines) must not exceed a maximum length in order to avoid reflections resulting in bus errors. This length depends on the bit rate used on the bus. The higher the bit rate the shorter the drop lines. At 1Mbps the drop line may not exceed 30cm.

A rule for estimation of the maximum allowable length of a drop cable length $\mathbf{L_{dc}}$ is given below.

$$L_{dc} < \frac{t_{\text{Prop\_seg}}}{50 \cdot t_p},$$

The total drop line length is calculated as following:

$$\sum_{i=1}^{n} L_{dc_i} < \frac{t_{\text{Prop\_seg}}}{10 \cdot t_p}$$

With: t_p Specific line propagation delay per length unit

t_{Prop_seg} Time of the propagation delay segment

This effectively leads to a reduction of the maximum trunk cable length by the sum of the actual cumulative drop cable length at a given bit rate. If the above recommendations are met, then the probability of reflection problems is considered to be fairly low. Drop lines must not have terminating resistors!

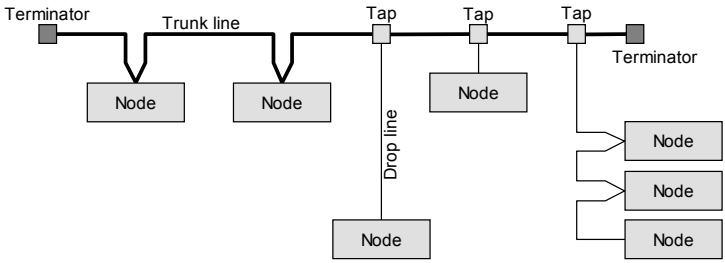


Figure 14: Physical layout of a CANopen network

The tables below show drop line length for single drop lines and star-shaped multiple drop lines for quick reference.

Note

The tables given below do not substitute a careful dimensioning and calculation of the application specific CAN-bus network.

Bit rate	Drop line length (single drop line)	Total length of all drop lines
1000 kbps	< 1m	< 5 m
500 kbps	< 5 m	< 25 m
250 kbps	< 10m	< 50 m
125 kbps	< 20m	< 100 m
50 kbps	< 50m	< 250 m

Table 11: Drop cable length (single drop line)

Bit rate	Drop line length (multiple drop lines, star shaped)	Drop line length (without drop lines)
1000 kbps	< 0,3 m	< 25 m
500 kbps	< 1,2 m	< 66 m
250 kbps	< 2,4 m	< 120 m

Bit rate	Drop line length (multiple drop lines, star shaped)	Drop line length (without drop lines)
125 kbps	< 4,8 m	< 310 m

Table 12: Drop cable length (multiple drop lines)

References

CiA 303-1

ISO 11898 part 1 and 2

See also

Section **Fehler! Verweisquelle konnte nicht gefunden werden.**, **Fehler! Verweisquelle konnte nicht gefunden werden.** on page **Fehler! Textmarke nicht definiert.**

This side was left empty intentionally.

5 Configuring

5.1 General rules for configuring CANopen networks

At the beginning of each system design the overall requirements must be evaluated. This includes, of course, the number and distribution of I/O points and implies the selection of the modules accordingly. On the communication side the evaluation should include response times, bandwidth usage, distances, as well as number and type of communication nodes.

Defining the system

This section focuses on how to setup the communication between devices. Thus, any needed control algorithms for PLCs are assumed to be implemented already.

At stage of system design with sysWORXX I/O modules involved, the following points should be considered:

	Means:
Participating devices	A participating devices must support the same physical layer and need to be at least conformant to CiA 301 V4.02. Some devices have multiple input/output options (e.g. voltage or current output). These device features should be defined in advance and be configured before connecting the sensors and actors. 3rd party devices (e.g. sensors with CANopen interface) can be integrated seamlessly if they fulfill the above requirements.
	Note Some devices (esp. some small CANopen sensors) only support configuration via LSS[6] and therefore must be considered as being unconfigured at first power-on. A LSS Master is required for configuration. All sysWORXX I/O devices support remote configuration via LSS according to CiA 305 V1.1.

[6] Layer Setting Services according to CiA 305

	Means:
Distribution and selection of node-IDs	Each node gets an unique node-ID between 1 and 127. The node-ID directly represents the message priority of this node. Assign a lower node-ID to nodes with high priority I/O connected (e.g. position sensors or drives). A higher node-ID (lower priority) can be assigned to nodes with I/Os connected to slow processes (e.g. temperature sensors).
Bit rate / bus speed	All devices on the network must support the same bit rate. The highest possible bit rate depends on the bus length and length of drop lines. See CiA 305-1 for detailed information. *Page **Fehler! Textmarke nicht definiert.*** shows some standard values for DC parameters for CANopen networks with less than 64 nodes. In general it is recommended not to run the network with a higher bit rate than required. Keeping the bus speed low reduces EMI and increases overall system stability and tolerance.
Synchronized operations	If an accurate timing is required by the application or parts of it (e.g. in motion control), the SYNC mechanism of CANopen is used. **Note** Depending on the amount of synchronized PDO used, SYNC might produce a considerable transmission peak, as synchronized PDOs are transmitted upon reception of the SYNC message. See *Section 8.6* for detailed information on how to use SYNC with the sysWORXX I/O modules.
Communication structure, and device guarding	Define the communication and connections (PDO) for all participating devices: • Master-Slave connections • Slave-Slave connections • Device guarding and network management issues (e.g. heartbeat) (who guards who)

Estimating the bandwidth usage

Calculating the bandwidth of a CANopen network without a simulation tool network is quite a difficult thing. However, a rough estimate bandwidth usage can be calculated as follows:

(1) Calculate the number of data bytes transmitted (e.g. at each SYNC cycle)

Example:

32 Digital inputs (BYTE)	→	4 bytes
4 Analog inputs (INT)	→	8 bytes
8 Digital outputs (BYTE)	→	1 byte
Total:		13 bytes

(2) Calculate the data bandwidth required. Either based on the communication cycle or based on a worst case scenario.

Example:
With an estimated SYNC cycle time of 13 ms and 13 data bytes, about 1000 bytes are transmitted within a second. Multiplying by 8 (to achieve bits per second) results in 8kbps.

(3) **Calculate the total bandwidth**
CAN messages not only contain data bytes but also message ID, control bits a checksum and other overhead information. Unfortunately there is no easy rule describing the relationship between data and overhead. The overhead factor may vary from 2 to 6 depending on the message length. If many short messages are used, a factor 6 could be reasonable.

Example:
Assuming an overhead factor of 4 result in a bandwidth of *4*8kbps = 32 kbps*.

With a chosen bitrate of 125kbps the average bandwidth usage is:
32kbps / 125kbps => 25,6%

About 25% is an acceptable margin for a rough estimation. In case the chosen bit rate would be 50kbps, a more detailed calculation becomes necessary.

Advanced development tools are capable of performing these calculation automatically. Please contact our support team if you need more information.

Determine the Communication Type

Once the bit rate has been chosen it is necessary to specify the PDO communication type(s). These have different advantages and disadvantages:

- **Cyclic synchronous communication** provides an accurately predictable bus loading, and therefore a defined timing behavior.

The main idea behind the synchronized communication mode is to provide motion oriented systems (such as robots) with “parallelized” inputs and outputs. The process values are updated synchronously. To avoid jitter effects and ensure smooth movements, all inputs are read at the same time and output data is applied simultaneously. The SYNC rate parameter determines the bus load globally. Under normal conditions the guaranteed reaction time of the system is at least as long as the cycle time. One drawback is that the CAN-bus bandwidth is not used optimally, since old data (e.g. data that has not changed) is also transmitted continuously. To optimize the network and reduce the bandwidth usage, the synchronization of a PDO can be scaled. SYNC multiples (transmission types 1...240) can be assigned to PDOs, to transmit slowly changing data less often than, for instance, time-critical inputs.

Note

Furthermore it is important to consider, that input states shorter than the SYNC cycle time will not necessarily be transmitted. If this is not possible for your application, associated PDOs must be configured for asynchronous communication.

- **Event-driven asynchronous communication** is quite the optimum in terms of reaction time and the exploitation of bus bandwidth. It uses transmission methods of "pure CAN". However, if a large number of events occur simultaneously, the corresponding delays before a PDO with a relatively low priority can be sent increases. Proper network planning therefore need to include a worst-case analysis. Certain mechanisms, for example the inhibit time, allow for controlling the traffic. Constantly changing inputs with a high PDO priority can be prevented from blocking the bus. This is why event driven communication is disabled by default in the device profile of analog inputs, and must be enabled explicitly. The so-called “Event timer” enables re-transmission of a PDO even without prior change of the I/O state. So the PDO is not sent again before the inhibit time has elapsed, and not later than specified with the “Even timer”.
- The communication type is parameterized by the so-called “Transmission Type” (*see Section 8.2*).

While each PDO can be configured for a single transmission type only, it is possible to combine different transmission types on devices with more than one TPDO. All sysWORXX input modules feature from two to four TPDO depending on module type.

Choosing devices and tools

Once the requirements are set, you probably need to select the devices and tools used to configure and test the devices and the network.

The following table provides an overview about the tools and services available for integration of the sysWORXX I/O devices.

Tool/device	Tasks	Scope of use
CANopen Device Monitor	Device configuration via direct access to Object Dictionary (SDO access) Performing network management tasks (NMT Master) Reading diagnostic data from the device Remote configuration via LSS Access and visualization of I/O data (e.g. process values) and PDOs	Configuration Test & Commissioning Maintenance
CANopen Configuration Manager	Overall CANopen network configuration (DCF generation) and configuration download via CAN-bus Changing PDO linking/mapping and configuration of communication parameters Configuration of SYNC and heartbeat producers Generation of network documentation	Configuration
CAN-REport	Logging CAN-bus messages (to screen or to file). Transmission of CAN messages. CANopen protocol plug-in for direct interpretation of CANopen messages to plain text. Plug-ins for data visualization.	Test & Commissioning Operation Maintenance
OpenPCS + CANopen PLC	Implementation and integration of distributed automation applications CANopen Configuration Manager functions integrated on the PLC when running in master/manager	Test & Commissioning Operation

Tool/device	Tasks	Scope of use
	mode Handling of CANopen network variables. Performing CANopen management tasks during operation. Performing OPC access to PLC variables.	
CANopen OPC Server	Mapping of CANopen network variables to OPC items.	Operation

Table 13: CANopen tools overview

5.2 Basic device configuration

Introduction

This section describes the steps for setting up the device to enable instant operation in a CANopen network (e.g. after deployment or replacement).

The basic device configuration covers:

- Setting of the node-ID
- Setting of the bit rate

Each sysWORXX IO device features three rotary HEX-encoding switches. Two are there for setting the node-ID and one for setting the CAN-bus bit rate.

After setup of bitrate and node-ID the device is ready for operation, e.g. accessible for further configuration via CAN-bus.

Required tools

Slotted screwdriver with 3.0 mm (0.118") blade

Procedure

Configuring the node-ID

Each CANopen device in a CANopen network must have an own unique node-ID from range 1 to 127. After power on the device checks the node-ID on the rotary switches. A configuration error (e.g. invalid node-ID) is displayed with a special LED blinking cycle (*see Section 9.1 on page 81 for details*).

Note

Changes at the hardware switches take effect only after power on or a reset of the device.

A alternative way to configure the device node-ID and bit rate is using the CANopen Layer Setting Services (LSS). Please refer to *Section 5.3* on *page 42* for detailed information on how to use LSS for node-ID configuration. When LSS was used for configuration, the settings on the hardware switches are ignored. The LSS settings are deleted with resetting the module to manufacturer settings.

The node-ID is configured in hexadecimal notation. One configures the high-nibble and the other the low-nibble of the note-ID. Figure 15 shows an example with node-ID 62H (respectively 92D) configured.

Note

Table 119 on *page 184* contains a table for node-ID conversation from decimal to hexadecimal notation.

Position FFh is reserved for resetting the device to factory settings (*see Section 5.6*)

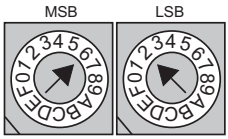


Figure 15: Example for a node-ID setup on hardware switches

Configuring the CAN-bus bit rate

The third hardware switch is used to select the CAN-bus bit rate.

Note

Changes at the hardware switches take effect only after power on or a reset of the device.

Alternatively it is possible to use the CANopen Layer Setting Services for switching the bit rate of a sysWORXX IO device or the CANopen network globally. Please refer to *Section 5.3* on *page 42* for detailed information on how to use LSS. When LSS was used for configuration, the set bit rate on the hardware switch is ignored. The LSS settings are deleted with resetting the module to manufacturer settings.

Table 14 shows the assignment of the CAN-bus bit rate to the position of the switch. A configuration error (wrong position) is displayed with a special LED blinking cycle (*see Section 9.1 on page 81 for details*). The assignment of the bit rates to the positions corresponds to the assignment used with LSS as defined in CiA 305.

Note

Position FH is reserved.

Switch position	Bit rate [kBit/s]
0	1000
1	800
2	500
3	250
4	125
5	100
6	50
7	20
8	10
0EH	1000, reserved for production

Table 14: Supported bit rates of the CANopen IO devices

5.3 Configuring using CANopen Layer Setting Services (LSS)

Introduction

LSS offers the possibility to inquire and change certain parameters of a sysWORXX CANopen I/O node via the CAN-bus.

The following device parameters can be inquired and/or changed using LSS:

- node-ID
- CAN-bus bit rate
- LSS address (Identity Object 1018H)

The sysWORXX I/O modules feature LSS slave functionality compliant to CiA 305 V1.1.

By using LSS a sysWORXX I/O device can be configured for a CANopen network without using the configuration switches[7]. The configured parameters are stored to a non-volatile memory after the configuration process has been finished successfully. The configuration of bit rate and node-ID on the switches is ignored and the configuration data is load from non-volatile memory after power on.

Note

The procedure of LSS access defined in the different versions of specification CiA 305 (e.g. V1.0 to V1.1) are not fully compatible. Thus, the LSS master must provide compatibility with all versions of the LSS specification used in the devices deployed on the CANopen network.

Requirements

The device identification data of the device to be configured must be known in advance. You can derive this information from the corresponding EDS. Furthermore the device serial number is needed. The device serial number is printed on a sticker placed on the module. If the sticker is missing please contact our support team for further assistance.

Required tools

CANopen configuration tool with LSS master function (e.g. CANopen Device Monitor with LSS plug-in)

PC/CAN interface

[7] However, before the module is accessible via LSS, valid values must be configured on the hardware switches to enable a normal startup behavior of the sysWORXX module.

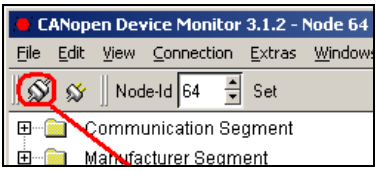
Procedure

In the following the procedures of configuring a sysWORXX I/O device via LSS are shown. It is assumed, that the module was installed properly and is at least connect to the CAN-bus and power. Furthermore the CANopen configuration tool and PC/CAN interface was installed and configured for operation.

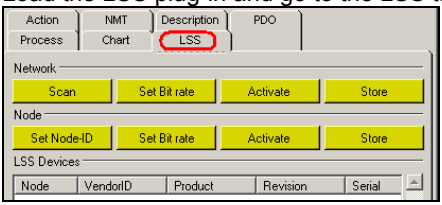
The description given below refers to the CANopen Device Monitor. Any other CANopen configuration tool featuring a LSS master might work as well but may differ in handling.

Configuring a node-ID via LSS

(1) Connect to the CAN network

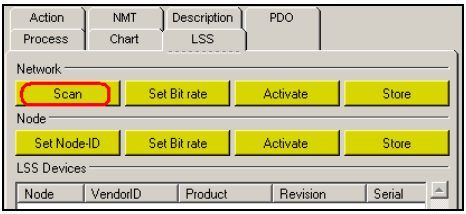


(2) Load the LSS plug-in and go to the LSS tab sheet



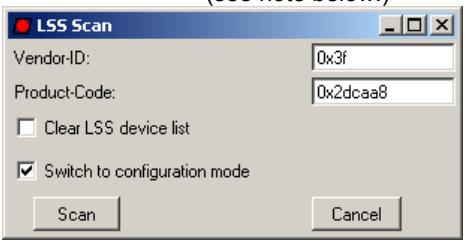
(3) There are two ways of adding nodes to the node list for configuration.
a) Right-click on the node-list to add an already configured node for changing its node-ID.

b) Scan the network for unconfigured devices according to some search criteria



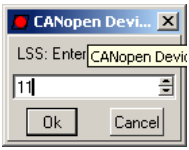
Vendor-ID:	0x3F for SYS TEC electronic GmbH (unique for all SYS TEC products)
Product code:	0x2DCAA8 for sysWORXX IO-X1

(see note below!)

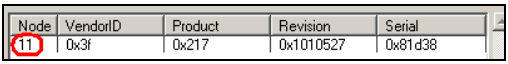


(4) Unconfigured devices appear on the node list. The node entry shows the vendor-ID, product code, revision number and serial number of the device.

(5) Select the node you want to configure and click on button **Set Node-ID**. Enter the new node-ID and confirm with **Ok**.



The device is shown with its new node-ID on the node list.



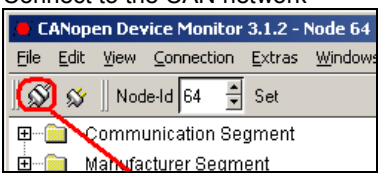
(6) Click on **Store** to save the changes to non-volatile memory. You are done!

Note

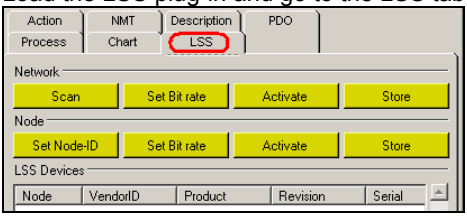
The product code of a sysWORXX IO device is equal to its order number. This means the product codes differs for each type of module and therefore enables a selective search

Configuring the bit rate of a device via LSS

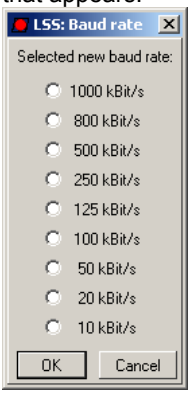
(1) Connect to the CAN network



(2) Load the LSS plug-in and go to the LSS tab sheet



(3) Click on **Set Bit rate** and select the new bit rate from the dialog that appears.



(4) Click on **Store** to save the changes to non-volatile memory.

(5) Click on **Activate** to take the changes into effect.

(6) The hardware configuration dialog appears. Switch your CAN-interface hardware to the new bit rate and reconnect to the network. You are done!

Note

Changing the bit rate of a single node does not make sense if the network consists of more than one node. Bus errors might occur. Switch the bit-rate of the CANopen network globally instead.

Reference

CiA 305 V1.1

5.4 Configuring with using Device Configuration Files (DCF)

Introduction

This section provides an overview about how to configure a sysWORXX I/O device using a Device Configuration Files (DCF).

Requirements

You need to have the DCF in hand before you start. Use a CANopen configuration tool to create the DCF (*see Section 2.2*) or derive it from an EDS manually (not recommended!).

Note

Some parameter modifications require a special sequence of actions (e.g. PDO mapping). The CANopen configuration tool or manager used for configuration should be able to handle this points automatically.

Required tools

CANopen configuration tool (providing SDO access to the Object Dictionary, e.g. CANopen Device Monitor)

PC/CAN interface

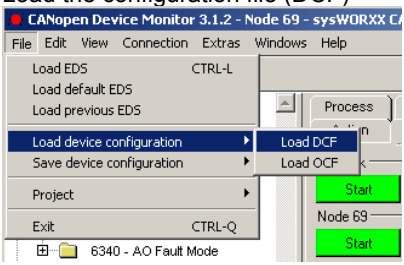
Procedure

This example shows how to configure a device by using the CANopen Device Monitor tool (CDM), which is part of the CANopen Configuration Suite. We assume the hardware components and network (CAN-bus interface and sysWORXX I/O modules) to be ready for operation.

(1) Open the CDM and connect to the network.

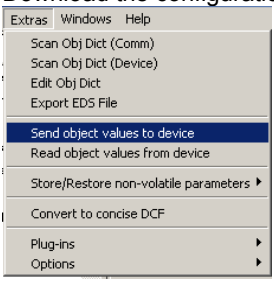
(2) Select the node you want to configure from the NMT tab sheet.

(3) Load the configuration file (DCF)

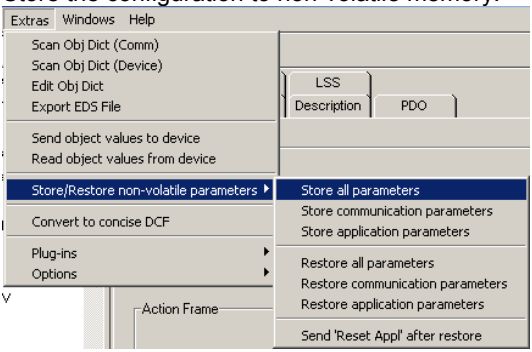


(4) Edit the configuration if needed.

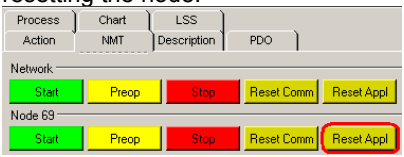
(5) Download the configuration to the device.



(6) Store the configuration to non-volatile memory.



(7) If the I/O configuration has been changed, the device needs to be reset in order to apply the changes.
Go to NMT tab-sheet and click on button **Reset Appl** for resetting the node.



Note

It is also possible to read-back the device configuration from a device by using the menu entries:

Extras->Scan Obj Dict(Comm) to scan the communication profile

Extras->Scan Obj Dict (Device) to scan the device profile

See also

L-1056e, CANopen Device Monitor, Software Manual

L-1055e, CANopen Configuration Manager, Software Manual

Section 5.5, Store/Restore device configuration

5.5 Store/Restore device configuration

Introduction

This section describes how to store a configuration to the non-volatile memory and remotely restore the factory settings.

The store / restore of configuration data is controlled by two object entries. Index 1010H is used for storing the configuration. For restoring the factory default settings, index 1011H is used.

Object Index	Object Subindex	Object name	Data type	Read value	Write value
1010H	1	Save all parameters	Unsigned32	1	"evas"
1011H	1	Restore all default parameters	Unsigned32	1	"daol"

Table 15: Object Dictionary entries for store / restore parameter (1010H/1011H)

Note

The sysWORXX I/O modules only support the “Save all parameters” feature.

Required tools

CANopen configuration tool (providing SDO access to the Object Dictionary, e.g. CANopen Device Monitor)

PC/CAN interface

Procedure

The storing/restoring of parameters is controlled by writing the signatures save/load into the corresponding object index/subindex.

Saving the configuration

Write “save” as hexadecimal value **65766173H** to Object 1010H Subindex 1 via SDO.

Signature	MSB			LSB
/ISO8859/ character	e	v	a	s
hex	65_h	76_h	61_h	73_h

If storing failed the device responds with a SDO abort code according to *Table 16*.

Restoring factory default settings

Write “load” as hexadecimal value **64616F6CH** to Object 1011H Subindex 1 via SDO.

Signature	MSB			LSB
/ISO8859/ character	d	a	o	l
hex	64_h	61_h	$6F_h$	$6C_h$

If restoring failed the device responds with a SDO abort code according to *Table 16*.

The restored default values become valid after the device was reset or power cycle (*see Figure 16*).

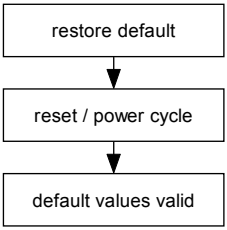


Figure 16: Restore procedure

SDO abort code	Description
0606000H	Store/Restore failed
0800002xH	Wrong signature

Table 16: SDO abort codes for store/restore configuration

Reference

CiA 301 V4.02

5.6 Resetting to factory settings

Introduction

These section describes how to restore the default factory settings locally on the device using the hardware switches.

Required tools

Slotted screwdriver with 3.0 mm (0.118") blade

Procedure

(1) Set the hardware switches for node-ID to value FFH

(2) Perform a hardware reset or power cycle.

(3) The blinking RUN and ERROR LED indicate the end of the restore process (see *Section 9.1* for LED blinking cycles).

(4) Set the original node-ID and perform a hardware reset or power cycle (see *Section 5.2* for basic device configuration).

Reference

CiA 303-3

6 Commissioning

6.1 Commissioning of the sysWORXX I/O modules

Introduction

The procedures for commissioning your automation system are determined by the relevant plant configuration. The procedure outlined below only describes the commissioning of sysWORXX I/O modules.

Requirements

We assume that the following steps have been completed successfully:

Actions	Reference
The module is installed and wired	Section 3, Mounting Section 4, Connecting
The device is configured (node-ID, bit rate)	Section 5, Configuring

Table 17: Commissioning requirements

Commissioning

(1) Switch on the device voltage supply (L+).

(2) Switch on the load voltage supply (supplies) (1L+, 2L+ …) if applicable.

See also

Section 3, Mounting

Section 4, Connecting

Section 5, Configuring

6.2 Startup of the sysWORXX I/O modules

Principle of operation

The diagram below illustrates the startup routine of a sysWORXX I/O module from application level. The module start-up and state from network level is described with the NMT state machine in *Section 9.5*. The steps described below refers to the steps the module is going through during NMT state INITIALIZATION (see the NMT state machine described in *Section 9.5*).

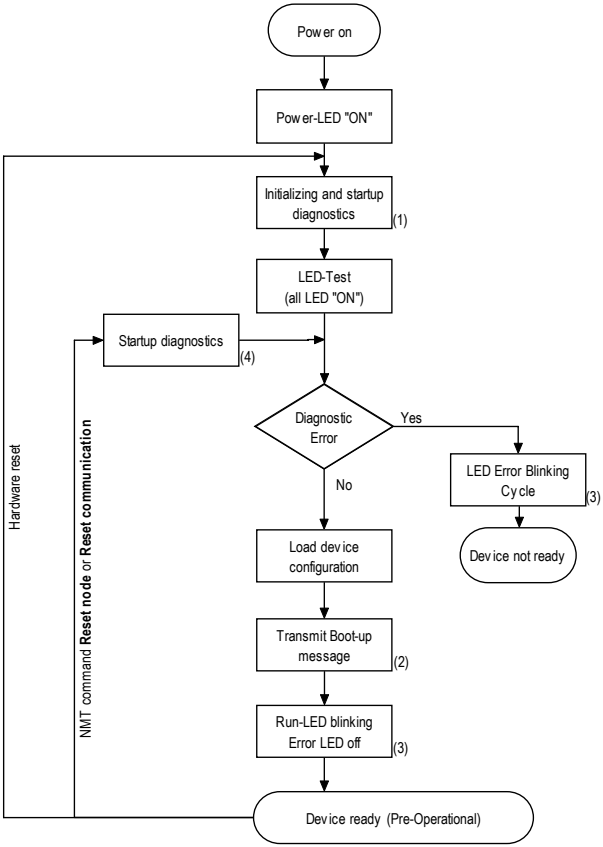


Figure 17: Startup cycle of a sysWORXX I/O device

(1) After power-on or hardware reset the device will perform some internal diagnostic covering:

Flash, *RAM*, *EEPROM*, *LEDs*, *Device configuration*

This checks will take 2-3 seconds depending on module type.

(2) The boot-up message is a single CAN-message with CAN-identifier *700H+node-ID* with 1 byte data containing the value 0.

It is used to signal that a CANopen slave device has entered the NMT state Pre-operational after the NMT state INITIALIZATION (*see Section 9.5*).

(3) Please refer to *Section 9.1* for detailed information about the LED blinking cycles

(4) After NMT command **Reset_Node** and **Reset_Communication** the device will perform some internal diagnostic covering: *Flash*, *RAM*, *EEPROM*

This checks will take about 2 seconds to complete.

Autonomous startup of CANopen network/devices

The sysWORXX I/O devices support the CANopen Minimum Boot Up. Following reset and internal initialization, the board is in state PRE-OPERATIONAL (*refer to Section 9.5*). Upon receipt of the NMT command message *Start_Remote_Node*) the device switches to state OPERATIONAL (*refer to Section 9.5*).

In some applications the use of a full NMT master may not be necessary. However, CANopen nodes need the *Start_Remote_Node* message to enter the OPERATIONAL state. Therefore, all sysWORXX I/O modules feature a manufacturer specific extension, which enables them to act as a simple NMT boot-up master. Please refer to *Section 8.4* for detailed information.

This side was left empty intentionally.

7 Maintenance and service

7.1 Removing and inserting I/O modules

Introduction

This section describes how to insert and remove I/O modules of a distributed system, which was already configured and put into operation. This might become necessary in case of defect or if the system configuration changes partly.

The sysWORXX I/O modules support insertion and removal of devices without effecting other devices (e.g. power-off).

Furthermore, the sysWORXX I/O modules support mechanisms for device monitoring (*see Section 9.5*) that enable the application master (e.g. a PLC) to detect missing devices by loss of communication, e.g. when powered-off. If this happens, the application master is responsible to perform appropriate actions (e.g. securing the machine, stop movement ect.).

After inserting a new device, the correct bit rate and node-ID must be configured (*see Section 5.2*) before it is connected to the bus.

Warning

An incorrect bit rate or node-ID might lead to severe communication problems and malfunction of the attached application.

After power-on of the device will perform some self diagnostics (*see Section 8.3*), which might take several seconds to complete. When the device is ready for operation, it will send out an boot-up message to notify the application master about its appearance. The application master is responsible to perform appropriate actions (e.g. configure the device).

In case an internal error was detected during the self diagnostics process, the device will not appear on the bus. Measures must be taken locally.

After successfully commissioning (*see Section 6*) the device, it needs to get configured, using a CANopen configuration tool. Some application masters provide the functionality of automatic device configuration, e.g. if a device was replaced.

Required tools

CAN-bus monitoring tool with CANopen protocol analyzer (e.g. CAN-Report with CANopen extension)

CANopen configuration tool providing SDO access (e.g. CANopen Device Monitor)

PC/CAN interface

Slotted screwdriver with 4 mm blade

Replacing the module

We assume that all tools are already installed on the Service-PC and the PC/CAN interface is connected and ready for operation.

(1) Power-off the device and remove all plugs

(2) Unmount the device and replace it with a new one. Because the sysWORXX I/O modules have removable terminal blocks, the wiring does not need to be touched.

(3) Configure bit rate and node-ID of the new device according to your network configuration.

(4) Connect the device to the bus.

(5) Connect the device to the plant (I/O).

(6) Connect the device to power supply.

(7) When the device is powered on, it starts internal diagnostic tests that take about 2 … 3 seconds to finish.

(8) Check the bus for the appearance of the boot-up message.

(9) Configure the device (e.g. PDO connections, I/O type ect.) using a CANopen configuration tool and DCF, if the application master does not provide automatic device configuration.

(10) Set the device to state OPERATIONAL to start PDO communication, if the application master does not manage this device.

See also

Section 9, Error behavior and system messages

Section 5, Configuring

Section 3, Mounting

8 Functions

8.1 The Object Dictionary of the sysWORXX I/O modules

Introduction

This section describes the communication specific part of the Object Dictionary (OD). The device specific part of the OD (6000H – 9FFFh) is described with the modules. The manufacturer specific part of the OD (2000H – 5FFFH) is described in *Section 8.4*.

Object Dictionary overview

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
1000H	Var	Device type number	Unsigned32	-	-	-
1001H	Var	Error Register	Unsigned8	-	-	-
1003H	Array	Pre-defined Error Field	Unsigned32	-	AUTO	AUTO
1005H	Var	COB-ID SYNC-Message	Unsigned32	-	X	X
1007H	Var	SYNC window length	Unsigned32	-	X	X
1008H	Var	Manufacturer Device name	String	-	-	-
1009H	Var	Manufacturer Hardware Version	String	-	-	-
100AH	Var	Manufacturer Software Version	String	-	-	-
100CH	Var	Guard Time	Unsigned16	-	X	X
100DH	Var	Life Time Factor	Unsigned8	-	X	X
1010H	Array	Store Parameter	Unsigned32	-	-	-
1011H	Array	Restore Default Parameters	Unsigned32	-	-	-
1014H	Var	COB-ID Emergency Message	Unsigned32	-	X	X

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
1016H	Array	Consumer Heartbeat Time [8]	Unsigned32	-	X	X
1017H	Var	Producer Heartbeat Time	Unsigned16	-	X	X
1018H	Record	Identity Object	Identity	-	-	-
1029H	Array	Error Behavior	Unsigned8	-	X	X
1200H	Record	1st Server SDO Parameter	SDO Parameter	-	-	-
1400H	**Record**	**RPDO1 [9] Communication parameter**	**PDOComPar**	**-**	**X**	**X**
	00H	Largest Subindex supported	Unsigned8			
	01H	COB-ID used by PDO	Unsigned32			
	02H	Transmission Type	Unsigned8			
	03H	Inhibit Time	Unsigned16			
	05H	Event timer	Unsigned16			
1401H	Record	RPDO2 Communication parameter	PDOComPar	-	X	X
1600H	**Record**	**RPDO1 Mapping parameter**	**PDOMapPar**	**-**	**X**	**X**
	00H	Number of Mapped Objects	Unsigned8			
	01H	PDO Mapping 1. App. Object	Unsigned32			
		…				
	08H	PDO Mapping 8.	Unsigned32			

[8] Object only available on IO-X1, IO-X3, IO-X6

[9] Up to 4 RPDOs are available on the sysWORXX I/O modules depending on module type.

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
		App. Object				
1601H	Record	RPDO2 Mapping parameter	PDOMapPar	-	X	X
1800H	**Record**	**TPDO1[10] Communication parameter**	**PDOComPar**	**-**	**X**	**X**
	00H	Largest Subindex supported	Unsigned8			
	01H	COB-ID used by PDO	Unsigned32			
	02H	Transmission Type	Unsigned8			
	03H	Inhibit Time	Unsigned16			
	05H	Event timer	Unsigned16			
1801H	Record	TPDO2 Communication parameter	PDOComPar	-	X	X
1A00H	**Record**	**TPDO1 Mapping parameter**	**PDOMapPar**	**-**	**X**	**X**
	00H	Number of Mapped Objects	Unsigned8			
	01H	PDO Mapping 1. App. Object	Unsigned32			
		…				
	08H	PDO Mapping 8. App. Object	Unsigned32			
1A01H	Record	TPDO2 Mapping parameter	PDOMapPar	-	X	X
1F51H	VAR	ProgramControl (from firmware version 1.30)	Unsigned8			

Table 18: Object Dictionary (Communication Profile)

[10] Up to 4 TPDOs are available on the sysWORXX I/O modules depending on module type.

Light-grey shaded objects are not available on all modules. The device specific PDO mapping is given with the device description.

References

CiA 301 V4.02

See also

Section 11, Digital I/O modules

Section 12, Analog I/O modules

8.2 CANopen Communication Services

Introduction

This section provides generic information about the CANopen communication services implemented on the sysWORXX I/O devices. Two services are available for data communication:

(1) **Process Data Objects** for fast transmission of process data without protocol overhead

(2) **Service Data Objects** for accessing the OD and transmission of service data (e.g. configuration download)

Process Data Objects (PDO)

The Process Data Object (PDO) implements an optimized method for placing multiple process data variables from the Object Dictionary into a single CAN message of up to 8 bytes.

Because CAN supports the multi-master communication concept (any node can send a message at any time and collisions are resolved by message priority), this direct communication method allows for more efficient, higher-priority access to process data.

The process data transferred via PDOs are divided into segments with maximum of 8 bytes (maximum data-length of a CAN message). The PDOs each correspond to a CAN message. PDOs are distinguished into Receive-PDOs (RPDOs) and Transmit-PDOs (TPDOs). A RPDO contains “output data”, received from the network. TPDO contain “input data”, that are to be sent out to the network.

There are two parameter sets to configure PDOs:

(1) **Communication Parameter Set** contains communication specific configuration (COB-ID assignment, transmission type, ect.).

(2) **Mapping Parameter Set** contains the assignment of application objects (process data) within the PDO.

The process of configuring PDOs includes the so-called **PDO linking** (communication parameterization) and **PDO mapping** (process data assignment/placement). If a CANopen network consists of more than two nodes the use of a CANopen configuration tool makes sense to avoid configuration errors and having the PDO linking and mapping done automatically by the tool.

PDO linking

In its default configuration (Pre-defined Connection Set), the PDO identifiers of a sysWORXX I/O device (here: slaves) are setup for communication with one central station (the master). For this kind of

communication structure the PDO communication parameters do not need to be changed.

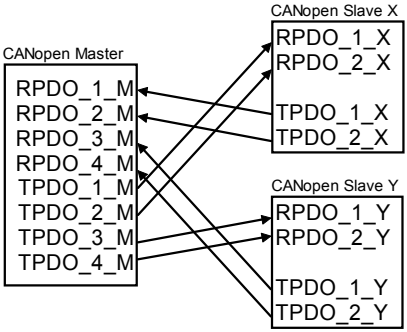


Figure 18: PDO linking for master/slave communication structure

If PDOs are used for direct data exchange between nodes (without a master involved), the identifier allocation of the devices must be adapted, so that the TPDO identifier of the producer matched with the RPDO identifier of the consumer.

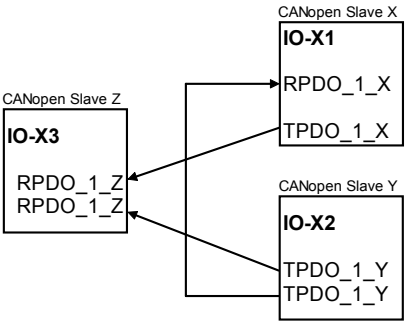


Figure 19: PDO linking for peer-to-peer communication structure

PDO Transmission Types

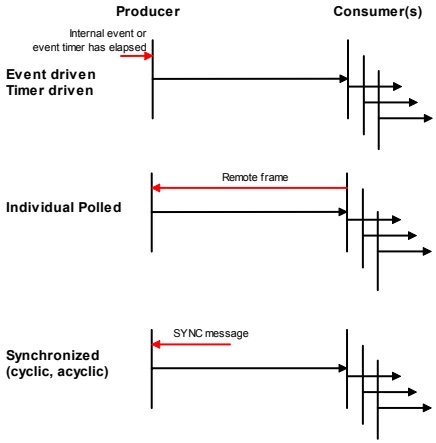


Figure 20: PDO transmission types

Transmit Trigger Options	Description
Event driven (asynchronous)	The event driven or change-of-state transmission method simply transmits a PDO if the process data mapped to it changes. The exact meaning of “event” thereby is specified in the corresponding device profile and is partly configurable by the user. It could be *any* change to the data as well as a *specific* change (e.g. positive edge or reaching a minimum difference). Event driven transmission does not depend on a master that polls for the data. The so-called **Inhibit Time** is a configurable timeout in multiplies of 100 microseconds to limit the frequency of a TPDO transmission. After starting the transmission of a TPDO the Inhibit Timer must expire before the TPDO may be transmitted again. **Note** One problem of event driven communication is the lack of determinism. It is very hard to predict the worst-case scenarios of how often messages will get transmitted. By using the Inhibit Time the worst-case becomes predictable as it can be directly determined by the Inhibit Time. Further device-specific communication control mechanisms are described with the modules.

Transmit Trigger Options	Description
Timer driven (cyclic, asynchronous)	In time driven communication method a PDO is transmitted at a fixed time basis, the Event Timer. The Event Timer is a local timer running on each node and specified in milliseconds. If the Event Timer is specified with 50ms, for example, the PDO is transmitted every 50ms. Per default the Event Timers of multiple nodes are not synchronized. **Note** On the one hand use of time driven transmission simplifies performance and latency calculations. On the other hand, it produces more overhead than pure event-driven communication since data will get transmitted even if it did not change at all.
Individual polled (remote requested)	Although it is possible to use individual polling in CANopen, it is not recommended that this communication method is used. Individual polling uses a CAN feature called “Remote-Request” (aka RTR frame) to trigger the transmission of a TPDO remotely. When using RTR frames the device behavior is usually not transparent to the user. Furthermore there are CAN controllers still in use that do not support remote frames at all. **Note** All sysWORXX I/O devices use CAN controllers following the FullCAN principle and make sure the requested data are up-to-date.
Synchronized	The synchronized communication method uses a SYNC signal. This SYNC signal is a specific message without any data only used for synchronization purpose. Because the SYNC signal is typically produced on a fixed time basis, this triggering mode can also be regarded as using a global timer for triggering instead of using the event timer local on each node. Please refer to Section 8.6 for mode detailed information on how to use SYNC with the sysWORXX I/O devices.

Table 19: TPDO transmit trigger options

Transmission type parameter	Cyclical	Acyclical	Synchronous	Asynchronous	RTR only
0		X	X		
1-240	X		X		
241-251	- reserved -				
252			X		X
253				X	X
254, 255				X	

Table 20: Transmission type parameter overview

Transmission type	Description
Acyclic synchronous	TPDOs with **transmission type 0** will get transmitted synchronously, but not cyclically. A corresponding RPDO is only evaluated after the next SYNC message has been received. This allows, for example, to give a new target position to axis groups one by one, but these positions only become valid with reception of the next SYNC signal. For TPDOs with transmission type 0, its input data are acquired with the reception of the SYNC message and then transmitted if the data state in it has changed. Transmission type 0 thus combines event-driven and time driven transmission (and, as far as possible, sampling) with synchronized processing given by the reception of a SYNC signal.
Cyclic synchronous	A TPDO configured with **transmission types 1…240** is transmitted cyclically after every ”n-time" (n = 1...240) reception of the SYNC message. Since transmission types for several TPDOs can be combined on a device as well as in the network, it is possible, for example, to assign a fast cycle for digital inputs (n = 1), whereas analog input values are transmitted in a slower cycle (e.g. n = 10). RPDOs do not generally distinguish between the transmission types 0...240. A received RPDO with a transmission type of 0 … 240 is set valid with the reception of the next SYNC message. The synchronous cycle time (SYNC rate) is stored in Object 1006H and thereby known to the consumer. If the SYNC fails the device reacts in accordance with the definition in the device profile

Transmission type	Description
	and switches, for example, its outputs into the fault state (See Object 1029H in Section 9).
RTR only	TPDO with **transmission types 252 or 253** are transmitted exclusively on request by reception of a Remote Request (RTR frame). Transmission type 252 is for synchronous transmission. Upon reception of the SYNC message the process data of the corresponding TPDO are acquired and it gets transmitted. Transmission type 253 is for asynchronous transmission. The process data of the corresponding PDO are acquired continuously, and transmitted upon reception of the RTR frame. **Note** This type of transmission is not generally recommended, because fetching input data from some CAN controllers is only partially supported. Furthermore, some CAN controllers sometimes replies to remote frames automatically (without requesting up-to-date input data from the application). Thus, under some circumstances the polled data might not be up-to-date.
Asynchronous	PDOs with **transmission types 254 and 255** are asynchronous, but may also be event-driven. For transmission type 254 the event is manufacturer-specific, whereas for type 255 the events are defined in the device profile. In the simplest way, the event is the change of an input value that is transmitted with every change in the value or state. These transmission types can be coupled with the Event Timer and Inhibit Time in order to control the transmission behavior.

Table 21: Transmission type description

PDO mapping

PDO mapping describes the mapping of the application objects (process data) from the Object Directory to the PDO.

All sysWORXX I/O modules support dynamic PDO mapping, which allows for changes on the mapping, even if the node is in state OPERATIONAL.

The CANopen device profile provides a default mapping for every device type, which is applicable for most applications. The default mapping for digital I/O, for example, simply represents the inputs and outputs in their physical sequence in the RPDO and TPDO respectively.

The PDO mapping is located in the Object Directory at index 1600H and following for the RPDOs and at 1A00H and following for TPDOs.

Figure 21 shows a example for mapping of three objects to the first TPDO of a node.

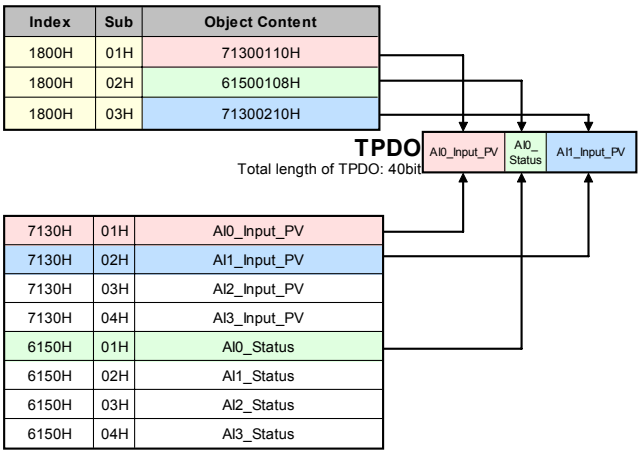


Figure 21: PDO mapping example

Usually CANopen configuration tools or configuration managers are used for changing the PDO mapping. However, under some circumstances it might become necessary to change the PDO mapping manually. Therefore the following procedure is necessary:

(1) Disable the PDO by setting its COB-ID to 80000xxxH
(xxx -> node-ID of the device to be changed)

(2) In the Mapping Parameter Set of the PDO set the number of mapped objects to 0
e.g. Object 1800H Subindex 00H for the first RPDO

(3) Change the mapping entries of the PDO

(4) Set the number of mapped back to a valid value according to the new PDO mapping.

(5) Set the COB-ID of the PDO back to its original value.

Dummy Mapping

A further feature of CANopen is the mapping of placeholders, or so-called “dummy entries”. The data type entries stored in the object directory, which do not themselves have data, are used as placeholders. If such entries are contained in the mapping table, the corresponding data from the device is not evaluated. In this way, for instance, a number of devices could be supplied with new set values

using a single CAN telegram, or outputs on a number of nodes can be set simultaneously, even in event-driven mode.

Service Data Objects (SDO)

The Service Data Object implements a direct communication channel for accessing the Object Dictionary. Service Data Objects (SDO) implement a basic client/server communication method, as point-to-point communication mode that allows for the issuing of read or write requests to the node's Object Dictionary. SDO messages contain requests or answers to/from the Object Dictionary. Because of its protocol overhead and master-driven communication principle, it is not well suited for process data communication.

A SDO connection is usually initiated by the application master, which acts as SDO client and owns all SDO communication channels. The sysWORXX I/O devices provide SDO servers, which means that at the request of a client (e.g. of the IPC or the PLC) they make data available (upload), or they receive data from the client (download).

References

CiA 301 V4.02

8.3 Internal diagnostics and monitoring functions

Introduction

The sysWORXX I/O modules feature two types of internal diagnostics and monitoring functionality:

(1) Device self testing at startup (Startup diagnostics)

(2) Device monitoring during runtime

Device diagnostics at startup

After **power-on** or **hardware reset** the device will perform the following tests:

- Flash
- RAM
- EEPROM
- LED's,
- Device configuration

On NMT command **Reset_Node** or **Reset_Communication** the following components are tested:

- Flash
- RAM
- EEPROM

If one of the above mentioned tests fails for any reason, the device will go to error state and indicate this by a special blinking cycle of the Run- and Error-LED (*see Figure 22 below)*. If this happens please contact our support team for further instructions.

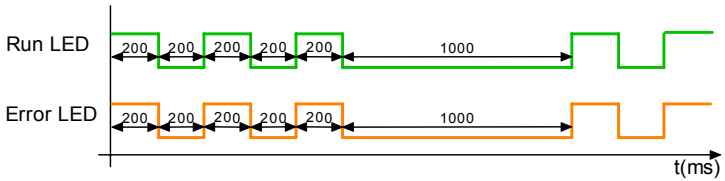


Figure 22: Error state blinking cycle

Device monitoring during runtime

Table 22 shows the monitoring functions that are performed during runtime. These functions are accessible through the manufacturer-specific section (Object 2001H).

If any abnormal condition is detected, the device will send out an Emergency message to report the error that occurred. Please refer to

Section 9.3 for further information on how to read/evaluate Emergency messages.

Function	Description
I/O circuitry	This function monitors the I/O lines for abnormal conditions. Depending on the type of module, several error conditions are monitored (e.g. cable break, short-circuit, measurement value out of range)
Device temperature	This function monitors the temperature in the enclosure. The device temperature can be read from the Object Dictionary (Object 2001H Subindex 01H) and is given with a resolution of 0.1 degrees centigrade. $Temp(°C) = \frac{OD\,value}{10}$
Device main voltage	This monitors the main voltage supplied to the device. The main voltage can be read from the Object Dictionary (Object 2001H Subindex 02H) and is given with a resolution of 0.1 Volts. $U_{main}(V) = \frac{OD\,value}{10}$
Runtime behavior	The sysWORXX I/O modules feature an internal watchdog to prevent undiscovered dead-locks. If a watchdog reset occurred, an Emergency message is sent out via CAN (*see Section 9.3 on page 88*).

Table 22: Internal runtime diagnostics and monitoring functions

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restore via 1011H
2001H	Array	Device Features	Integer16	-	-	-
	00H	Number of Entries	Unsigned8			
	01H	Device temperature	Integer16			
	02H	Device main voltage	Integer16			

Table 23: Object Dictionary entries for diagnostic and monitoring functions

References

CiA 301 V4.02

8.4 Manufacturer specific extensions

Introduction

This section describes the manufacturer specific functions implemented in the sysWORXX I/O modules.

In addition to the corresponding device profile, the following extensions are available:

- Minimal NMT boot up master (Object 2000H)
- I/O filtering (Object 2010H)
- PowerFail configuration

Minimal NMT boot up master

The minimal NMT boot up master function enables operation of sysWORXX I/O devices without NMT master present on the network. This function handles the transmission of a NMT boot up message after a given delay time has expired.

Two object entries (Object 2000H Subindex 01H) exist to control this function. Refer to *Table 25* for detailed information. Any change of settings for this function is stored to non-volatile memory immediately after write access, independent of the common load/save mechanism provided with Object 1010H and Object 1011H. To activate the new settings, a reboot (by reset or power on) is necessary.

Powerfail configuration

The monitoring of main voltage by power fail can be deactivated. If it is deactivated no emergency message and no reset is generated if main voltage drops under power fail level (see 9).

I/O filtering

This function implements a bit-wise applied filter for digital inputs on the sysWORXX I/O modules. It allows for selective Enable/Disable of digital inputs.

Object Dictionary entries

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
2000H	**Var**	**NMT Boot Configuration**	**Unsigned8**	**-**	**Auto access** [11]	**Auto access**
	00H	Number of Entries	Unsigned8			
	01H	NMT Boot enable	Unsigned8			
	02H	NMT Start Time	Unsigned16			
2001H	Array	Device Features [12]	Integer16	-	-	-
2002H	**Var**	**Power Fail Configuration** (from firmware version 1.30)	**Unsigned8**	**-**	**X**	**X**
	00H	Number of Entries	Unsigned8			
	01H	PowerFail Interrupt enable	Unsigned8			
2010H	**Array**	**Disable digital input 8-Bit**	**Unsigned8**	**-**	**X**	**X**
	00H	Number of Entries	Unsigned8			
	01H	DI0_DI7_Disable	Unsigned8			
	02H	DI8_DI15_Disable	Unsigned8			
	03H	DI16_DI23_Disable	Unsigned8			

Table 24: Object Dictionary entries for manufacturer specific extensions

The light-grey shaded objects are only available on module type IO-X1 and IO-X2.

Parameter description

Parameter	Description
NMT Boot enable	Enable or disable the NMT boot function 0 = disable

[11] Value is stored to non-volatile memory immediately after write access to the Object.

[12] Object 2001H is described in Section 8.4, Table 23 on page 70.

Parameter	Description
	1 = enable Default value: 0
NMT Start Time	This index contains the delay time for the boot function. The time base is milliseconds. Default value: 500ms.
Power Fail Interrupt Enable (from firmware version 1.30)	Enable or disable Power Fail monitoring 0 = disable 1 = enable (default)
DIx_DIx_Disable	Byte-value, which is applied bit-by-bit to a digital input block on the device. 0 = disable 1 = enable Default value: 00H (all disabled)

Table 25: Parameter description for manufacturer specific extensions

References

CiA 301 V4.02

8.5 Device identification data

Introduction

The Identity Object provides identifying information about the node. It stores basic information about the manufacturer, the product, revision and serial number and therefore is unique for each CANopen device. This Object is mainly used for remote configuring via LSS.

Object Dictionary entries

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
1018H	**Record**	**Identity Object**	**Identity**	**-**	**-**	**-**
	00H	Number of Entries	Unsigned8			
	01H	Vendor ID	Unsigned32			
	02H	Product Code	Unsigned32			
	03H	Revision Number	Unsigned32			
	04H	Serial Number	Unsigned32			

Table 26: Object Dictionary entry for the Identity Object

Parameter description

Parameter	Description
Vendor ID	This Subindex contains the identification code of the manufacturer of the device. This value is assigned uniquely to each vendor of CANopen devices by the CiA[13] Users and Manufacturers Association.
Product Code	This Subindex contains the unique value assigned by the vendor, specifying the device type. For the sysWORXX I/O modules, this Subindex contains the order number of the device.

[13] CAN in Automation (http://www.can-cia.org)

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Revision Number</td><td>This Subindex stores the revision number of the device firmware, assigned by the vendor. The table below shows the structure of this value.
MSB LSB
31 24 23 16 15 8 7 0
<table><tr><td colspan="2">Firmware</td><td colspan="2">CANopen Stack</td></tr><tr><td>Major revision</td><td>Minor revision</td><td>Major revision</td><td>Minor revision</td></tr></table>e.g. The value 01030528H is to be read:
Firmware version: 1.03
CANopen stack version: 5.28</td></tr>
<tr><td>Serial Number</td><td>This Subindex contains the serial number of the device.
The serial number can also be find on a sticker (number + barcode) placed on the enclosure.</td></tr>
</table>

Table 27: Parameter description Identity Object

References

CiA 301 V4.02

8.6 Synchronized operations

Introduction

This section describes the configuration of SYNC settings for the sysWORXX I/O modules and provides a brief overview on how synchronization works in CANopen networks.

In CANopen, the synchronized communication method is implemented using a SYNC signal, which is a specific message (SYNC message) without any data and high priority. SYNC is based on the Producer/Consumer principle. Typically, the SYNC producer transmits SYNC messages on a fixed time basis. The number of SYNC producers in a CANopen network is not limited. This enables setup of different groups of synchronized operating devices.

SYNC principle in CANopen

Synchronized communication for inputs (sensors)

The sensors constantly read their input data and keep a current copy in the message transmit buffer. Upon reception of the SYNC message, all sensors stop updating the message transmit buffer and start transmitting the data. Although all messages are transmitted serially via CAN, the data received by the main controller are from the same moment of time (i.e. the moment the SYNC signal was received by the sensors).

Synchronized communication for outputs (actuators)

Once the processing unit has new values for the outputs it transmits the data serially via CAN. The actuators receiving the messages keep the received data in their receive buffers without applying the data to their outputs. Upon the reception of the next SYNC signal the data are applied to the outputs in parallel.

Note

The complete communication cycle, including transmission of input data, processing and transmission of output data, should be finished within the communication cycle period! Refer to *Figure 23* below.

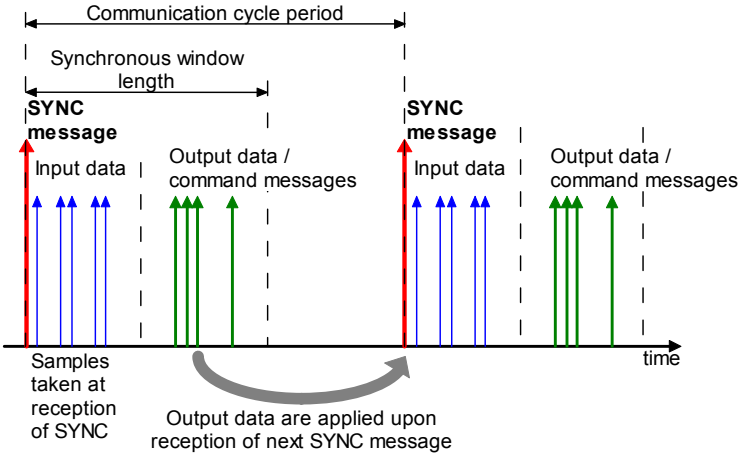


Figure 23: Synchronized communication principle in CANopen

Object Dictionary entries

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
1005H	**Var**	**COB-ID SYNC**	**Unsigned32**	**-**	**X**	**X**
	00H	COB-ID SYNC	Unsigned32			
1007H	**Var**	**Synchronous Window Length**	**Unsigned32**	**-**	**X**	**X**
	00H	Synchronous Window Length	Unsigned32			

Figure 24: Object dictionary entries for SYNC

Parameter description

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>COB-ID SYNC</td><td>Contains the COB-ID used by the SYNC Object along with a flag to indicate if the node generates the SYNC Object or not.

For 11-bit CAN identifier (COB-ID) the value is constructed as follows:

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0-10</td><td>COB-ID for SYNC Object</td></tr>
<tr><td>11-28</td><td>Set to 0 (reserved for 29-bit COB-ID)</td></tr>
<tr><td>29</td><td>Set to 0 to select 11-bit COB-ID</td></tr>
<tr><td>30</td><td>Set to 0 as the sysWORXX I/O devices do not support generation of SYNC messages.</td></tr>
<tr><td>31</td><td>Do not care (set to 0)</td></tr>
</table>
Note
The sysWORXX I/O modules only support 11-bit COB-ID and cannot operate as SYNC producer.

Default value: 80H/128</td></tr>
<tr><td>Synchronous Window Length</td><td>This entry defines the period of time in microseconds after a SYNC Object has been transmitted on the bus in which synchronous PDOs must be transmitted.

This period must be smaller than the Communication Cycle Period (see Figure 23). Each node using the same SYNC COB-ID must have the same Synchronous Window Length.

If the node fails to transmit the PDO within the Synchronous Window Length (e.g. because higher prior messages were transmitted on the bus), this PDO is not transmitted again for this cycle.

Note
Synchronous PDO are never transmitted outside the Synchronous Window Length. This requires a careful assignment of message priorities during application planning in order to make sure all data can be transmitted in time.

Default value: 00H (SYNC not used)</td></tr>
</table>

Figure 25: Parameter description for synchronous operation

References

CiA 301 V4.02

This side was left empty intentionally.

9 Error behavior and system messages

9.1 Device status LEDs

Introduction

This section describes the meaning and blinking cycles of the Run- and Error-LED on the sysWORXX I/O devices. The I/O status LEDs are described with the devices.

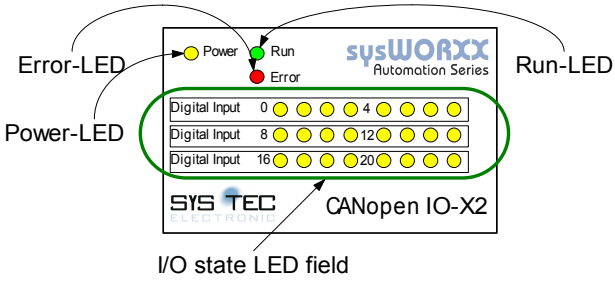
In addition to the module state, some hardware errors are displayed, too. The reason of the hardware error is displayed in the I/O state LED field (1st line). These error states are manufacturer-specific and highlighted with a light-grey background in *Table* 28 and *Table* 29.

Status LEDs

The **Run-LED** (green) indicates the current NMT state of the sysWORXX I/O module.

The **Error-LED** (red) indicates errors that occurred (e.g. CAN-bus, configuration error).

An overview about the LED display is given below.



Description of LED states

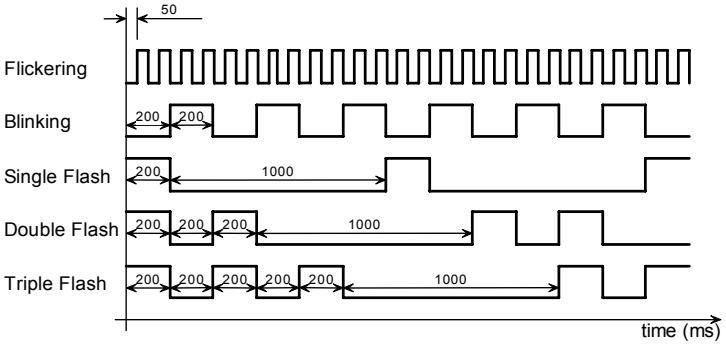


Figure 26: LED blinking cycles of the sysWORXX I/O modules

Run-LED state	NMT state	Description
Always On	OPERATIONAL	The device is in NMT state OPERATIONAL
Blinking	PRE-OPERATIONAL	The device is in NMT state PRE-OPERATIONAL
Single Flash	STOPPED	The device is in NMT state STOPPED
Flickering	OPERATIONAL or PRE-OPERATIONAL	LSS service in progress (alternate flickering with Error-LED)
Synchronous blinking with Error-LED	INITIALIZING	**Configuration error** A wrong configuration selected at hardware switches.
Synchronous Triple Flash with Error-LED	INITIALIZING	**Hardware error** Hardware error detected during internal diagnostics.

Table 28: Description of Run-LED states

Error-LED states	NMT-state	Description
Off	No error	The device is operating under normal conditions.
Flickering	OPERATIONAL or PRE-OPERATIONAL	LSS service in progress (alternate flickering with Run-

Error-LED states	NMT-state	Description
		LED)
Single Flash	OPERATIONAL or PRE-OPERATIONAL	**Warning limit reached** At least one of the error counters of the CAN controller has reached or exceeded the warning limit
Double Flash	OPERATIONAL or PRE-OPERATIONAL	**Error control event** A node guarding event or heartbeat event has occurred (*see Section 9.5*).
On	OPERATIONAL or PRE-OPERATIONAL	**Bus off**[14] The CAN controller is in state bus-off (too many error frames on the bus).
Synchronous blinking with RUN LED	INITIALIZING	**Configuration error** A wrong configuration is selected with the rotary switches. see Table 30
Synchronous Triple Flash with RUN LED	INITIALIZING	**Hardware error** The internal diagnostic functions detect an hardware error during power on and the NMT command "Reset Node". see Table 30

Table 29: Description of Error-LED states

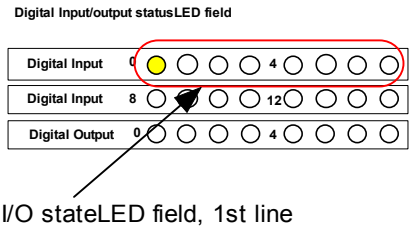


Figure 27: Signaling configuration or hardware errors, example for baudrate error, see Table 30

[14] After the Bus off error state has disappeared, the module sends out an Emergency message (see Section 9.3) and continues with normal operation. The NMT state (see Section 9.5) remains unchanged.

I/O stateLED 1St line	Description
LED0	configuration error: Baudrate switch out of range (value >8)
LED1	configuration error: Node-ID switch out of range (0 or >7FH)
LED2	hardware error: serial number invalid
LED3	hardware error: CRC error, nonvolatile memory
LED4	hardware error: product code invalid
LED5	hardware error: calibration data invalid

Table 30: Description of configuration and hardware error signaling

Error	Action
Warning limit reached	Please refer to *Section* 13.2 for a detailed test procedure.
Bus off	Please refer to *Section 13.2* for a detailed test procedure.
Configuration error	Check for correct settings on the hardware switches and reset. If this doesn't help reset to factory defaults (*see Section 5.6*). If the error still persists contact the support for further assistance
Hardware error	Please contact the support for further assistance!

Table 31: User action required for error events

If Program Control (CANopen Bootloader) is active the RUN- and ERROR-LED gets a special function for monitoring program download (from firmware version V1.30):

Run-LED state	ERROR-LED state	Description
Always Triple Flash	Off	program download is running
	Single Flash	reason for bootloader is "application-signature is not set"
	Double Flash	reason for bootloader is "application-CRC is wrong"

Table 32: Description of Run- and Error-LED at Program Control

Reference

CiA 303-3 V1.0

9.2 Reading diagnostic data

Introduction

In addition to the state LEDs the sysWORXX I/O devices feature several standardized Object Dictionary entries providing detailed information about the device state and an error history.

On some device types extended status information for I/Os are provided in the device profile section of the Object dictionary. These Objects are described with the respective device.

This section describes the diagnostic data readable via OD access during runtime. Internal diagnostics at startup and monitoring features are described in *Section 8.3* on *page 69*. Emergency messages are described in *Section 9.3* on *page 88*.

Object Dictionary entries

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
1001H	**Var**	**Error Register**	**Unsigned8**	**-**	**-**	**-**
	00H	Error Register	Unsigned8			
1003H	**Array**	**Pre-Defined Error Field**	**Unsigned32**	-	-	-
	00H	Number of Entries / Error Counter	Unsigned8			
	01H to 0AH	Standard Error Field	Unsigned32			

Table 33: Object Dictionary entries for error data on the sysWORXX I/O devices

Parameter description

Parameter	Description
Error Register	The error register value indicates if various types of errors have occurred. It is a part of the Emergency object, which is transmitted with the Emergency message. The following error values are implemented: 00H = no error, respectively error reset 01H = generic error 11H = CAN communication error 81H = manufacturer specific error

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td></td><td>85H = manufacturer specific error, voltage</td></tr>
<tr><td>Error Count</td><td>Contains the number of errors stored in Object 1003H. Writing the value 00H to this entry results in resetting the stored values in the Standard Error Fields, i.e. the error history (see below).</td></tr>
<tr><td>Standard Error Field</td><td>This Object provides an error history containing the 10 most recent errors that occurred on the node and result in the transmission of the Emergency message. Subindex 01H always contains the most recent error. If a new error occurs, it will be stored to Subindex 01H and the older values are shuffled down. Subindex 00H contains the number of errors stored in the error history.
Note
The complete error history (Object 1003H is stored to non-volatile memory and restored after power cycle. The error history can be erased by writing the value 0 to Subindex 0 of Object 1003H.
This entry has the following structure:
MSB LSB
31 16 15 0
Manufacturer specific additional information | Error code
A list of error codes is given in Section 9.3.
In the area of the additional information are stored, for example, the channel number of an input were the error occurred.</td></tr>
</table>

Table 34: Parameter description for error data

Reference

CiA 301 V4.02

See also

Section 11, Digital I/O modules

Section 12, Analog I/O modules

9.3 Evaluation of diagnostic messages (CANopen Emergency messages)

Introduction

Each sysWORXX CANopen I/O module features an Emergency Object (aka EMCY) to report errors via CAN (Emergency messages). This enables a remote device with Emergency Consumer Service, typically the application master (e.g. a PLC), to listen to this messages and thus react on specific errors.

Any malfunctions of the following components of a device are covered by Emergency messages:

- Digital outputs
- Analog input and outputs
- Integrated power supply and diagnostics

In general, Emergency messages are only reported once, as the reported error is considered to be existing (“still be there”) until the node uses another Emergency message to clear/reset that specific error.

Error conditions for digital outputs

Refers to modules:

CANopen IO-X1 and CANopen IO-X3

Error condition	Scope
short-circuit	Channel

Table 35: Error conditions for digital outputs

Error conditions for analog inputs

Refers to modules:

CANopen IO-X4, CANopen IO-X5 and CANopen IO-X7

Error condition	Scope
Line-break/short-circuit	Channel
Process value exceeded lower-limit of measurement range	Channel
Process value exceeded upper-limit of measurement range	Channel
Configuration error (invalid value range)	Channel

Table 36: Error conditions for analog inputs

Error conditions for analog outputs

Refers to module CANopen IO-X6.

Error condition	Scope
Line-break/short circuit	Channel

Table 37: Error conditions for analog outputs

Error conditions for integrated power supply and diagnostics

Refers to all modules.

Error condition	Scope
Power-fail	Device
Overheat	Device
Configuration error	Device
Hardware error	Device

Table 38: Error conditions for power supply and diagnostics

Emergency message structure

An Emergency message always contains 8 data bytes. The first two bytes hold the CANopen Error Code (*see Table 40*). The third byte contains a copy of the error register (see Object 1001H in *Section 9.2*) and the remaining 5 bytes contain the manufacturer specific error code.

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
Error Code		Error Register	Manufacturer specific error code				

Table 39: Structure of an Emergency message

Emergency error codes

Emergency error code	Description
0000H	No error / error reset
2310H	Current at the digital output to high (overload)
3120H	Mains voltage too low, power-fail
4201H	Internal device temperature above 60°C (overheat)
5001H	Hardware reset caused by watchdog or reset button
5002H	CRC error on FLASH memory **WARNING** Device not ready for operation and must be replaced! Please contact our support for further instructions.

Emergency error code	Description
5003H	CRC error on RAM  **WARNING** Device not ready for operation and must be replaced! Please contact our support for further instructions.
5004H	CRC error on EEPROM  **WARNING** Device not ready for operation and must be replaced! Please contact our support for further instructions.
6101H	Unexpected software reset  **WARNING** Device not ready for operation and must be replaced! Please contact our support for further instructions.
6102H	Stack overflow  **WARNING** Please contact our support for further instructions.
6103H	Unused software interrupt  **WARNING** Please contact our support for further instructions.
6110H (from firmware version V1.30)	reason for starting bootloader is “application-signature is not set” This information is not saved in Predefined Errorfield.
6111H (from firmware version V1.30)	reason for starting bootloader is “application-CRC is wrong” This information is not saved in Predefined Errorfield.
8110H	CAN overrun error. CAN message could not be transmitted.
8120H	CAN controller in error passive mode
8130H	Lifeguarding or heartbeat error
8140H	CAN controller recovered from bus off

Emergency error code	Description	
8210H	PDO not processed due to length error	
Device specific error codes		**Scope**
FF03H	Sensor fraction on input	Channel; IO-X4, IO-X5, IO-X7
FF04H	Sensor overload on Input	
FF05H	Short-circuit at input	
FF06H	Chosen value range too low for configured sensor type	
FF07H	Chosen value range too high for configured sensor type	
FF08H	If channel configured as voltage input (U-mode): short circuit	Channel; IO-X6
	If channel configured as current input (I-mode): open output, no load connected, cable-break	

Table 40: Supported emergency error codes

Object Dictionary entries

The following table describes Object 1014H, used to configure the Emergency message COB-ID of an device.

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
1014H	**Var**	**COB-ID Emergency message**	**Unsigned32**	**-**	**X**	**X**
	00H	COB-ID Emergency message	Unsigned32			

Table 41:Object Dictionary entries for the Emergency COB-ID

Parameter description

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>COB-ID Emergency message</td><td>Defines the COB-ID used for the Emergency message transmitted by the node and specifies if the Emergency Object is used or not.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0-10</td><td>COB-ID for Emergency message</td></tr>
<tr><td>11-28</td><td>Set to 0 (reserved for 29-bit COB-ID)</td></tr>
<tr><td>29</td><td>Set to 0 to select 11-bit COB-ID</td></tr>
<tr><td>30</td><td>Reserved, set to 0.</td></tr>
<tr><td>31</td><td>Set to 0 if the node does use the Emergency Object.
Set to 1 if the node does not use the Emergency Object.</td></tr>
</table>
Note
The sysWORXX I/O devices only support 11-bit identifiers (COB-IDs)</td></tr>
</table>

Table 42: Parameter description for the Emergency COB-ID

Reference

CiA 301 V4.02

9.4 Error behavior

Introduction

This section describes the behavior (state-change) of a sysWORXX I/O device in case errors. This behavior is configurable by the user via Object 1029H and several device specific objects.

Emergency messages are covered by *Section 9.3* and therefore not included in this section.

Supported error conditions

The sysWORXX I/O modules perform a state-change upon the following communication errors:

- Bus off on the CAN controller
- Life guarding event occurred
- Heartbeat event occurred

Supported state changes

The following state-changes may be performed:

- NMT state-change
- Output state change (device specific)

Object Dictionary entries

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
1029H	**Array**	**Error Behavior**	**Unsigned8**	**-**	**X**	**X**
	00H	Number of Error Classes	Unsigned8			
	01H	Communication Errors	Unsigned8			
6206H[15]	Array	Error Mode Output 8-bit	Unsigned8	-	X	X
6207H[1]	Array	Error Value Output 8-bit	Unsigned8	-	X	X
6340H[16]	Array	AO Fault Mode	Unsigned8	-	X	X
7341H	Array	AO Fault Value	Integer16	-	X	X

Table 43: Object Dictionary entries for configuring the error behavior

The light-grey shaded objects are not available on all module types. Please refer to the corresponding device description for detailed information.

[15] Objects 6206H and 6207H are only available on modules with digital outputs (IO-X1, IO-X3)

[16] Objects 6340H and 7341H are only available on modules with analog outputs (IO-X6)

Parameter description

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Communication Error</td><td>Defines the behavior of the node when a communication error is encountered.
Valid values are:<table><tr><th>Value</th><th>Description</th></tr><tr><td>00h</td><td>Switch to NMT state PRE-OPERATIONAL</td></tr><tr><td>01h</td><td>No change of NMT state</td></tr><tr><td>02h</td><td>Switch to NMT state STOPPED</td></tr></table>Default value: 00H</td></tr>
<tr><td>Error Mode Output 8-bit</td><td>Device specific parameter. Described in Section 11.1</td></tr>
<tr><td>Error Value Output 8-bit</td><td>Device specific parameter. Described in Section 11.1</td></tr>
<tr><td>AO Fault Mode</td><td>Device specific parameter. Described in Section 12.3</td></tr>
<tr><td>AO_Fault_Value</td><td>Device specific parameter. Described in Section 12.3</td></tr>
</table>

Table 44: Parameter description for configuring the error behavior

Reference

CiA 301 V4.02

CiA 401 V2.1

CiA 404 V1.2

9.5 Module/Network status and device guarding

Introduction

This section provides information about the network management capabilities of the sysWORXX I/O modules when deployed in a CANopen network.

Each sysWORXX I/O module implements a CANopen NMT slave device. This enables a Network Management Master (e.g. a PLC) to watch over all nodes to see if they are operating within their parameters. Upon failure of a node or reception of a certain alarm/emergency message it can initiate the appropriate recovery or shutdown procedures. Therefore a so-called “NMT state machine” is implemented on the device and several options for device guarding are supported:

Module state machine

The NMT state machine implemented sysWORXX I/O modules allows the device to be in different operating states. A NMT master can initiate state transitions by sending so-called “node control messages” to either a single node or all nodes on the network. The NMT state machine of the sysWORXX I/O devices is shown below.

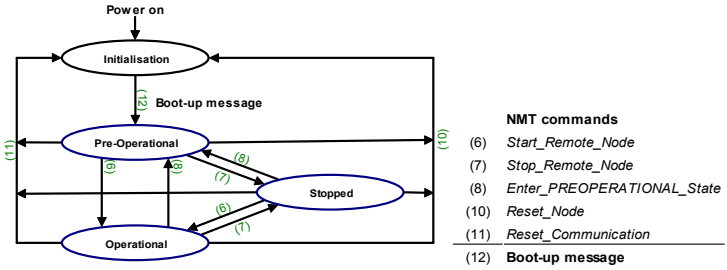


Figure 28: The NMT state machine

In state **INITIALIZATION** , the CANopen data structures (e.g. Object Dictionary) of a node is initialized by the application. This startup procedure is described in *Section 6.2* on page 52.

After INITIALIZATION has been completed the node automatically switches into state **PRE-OPERATIONAL**(12) and transmits the *Boot-up* message to inform the NMT master about this state change. In this state PDO communication is disabled. However, device access via SDO, NMT services and device guarding are available in this state.

After the device configuration has been completed (typically done by the application or the NMT master), the NMT command *Start_Remote_Node* (6) can be used to switch the node from state

PRE-OPERATIONAL into state **OPERATIONAL**. This state change results in the initial transmission of all active TPDOs to make the current process I/O state known to the network.

NMT command *Reset_Node* (10) is used to reset node remotely. The power on values or values stored in non-volatile memory (if previously stored) are used for reset values.

In state **STOPPED** any communication except NMT, Heartbeat and Nodeguarding is disabled.

All sysWORXX I/O devices also support the NMT commands *Stop_Remote_Node* (7), *Enter_PRE-OPERATIONAL_State* (8), *Reset_Node* (10), *Reset_Communication* (11) to control state transitions (see *Figure* 28 and *Table* 46).

	INITIALIZING	PRE-OPERATIONAL	OPERATIONAL	STOPPED
Boot-up	**X**			
SDO		**X**	**X**	
EMCY		**X**	**X**	
SYNC		**X**	**X**	
Heartbeat/ Node guarding		**X**	**X**	**X**
PDO			**X**	

Table 45: NMT state dependent communication

NMT command messages

The first data byte of a NMT command message always contains the NMT command. The second byte contains the node-ID of the device to be started. The value 00H for node-ID addresses all nodes in the network (Broadcast). The COB-ID is always set to 000H.

Command	Description
Start_Remote_Node	This command is used to set one or all nodes to state OPERATIONAL. COB-ID: 000H; 2 byte data: 01H, node-ID
Stop_Remote_Node	This command is used to set one or all nodes to state STOPPED. COB-ID: 000H; 2 byte data: 02H, node-ID

Command	Description
Enter_PREOPERATIONAL_State	This command is used to set one or all nodes to state PRE-OPERATIONAL. COB-ID: 000H; 2 byte data: 80H, node-ID
Reset_Node	This command is used to reset one or all nodes on the network. COB-ID: 000H; 2 byte data: 81H, node-ID
Reset_Communication	This command is used to reset the communication parameters of one or all nodes on the network. COB-ID: 000H; 2 byte data: 82H, node-ID

Table 46: NMT commands

Options for device guarding

Node guarding / Life guarding

With *node guarding* the NMT master polls all slaves for their current NMT state information. If a node does not respond within a specified time the NMT master assumes that this slave was lost and can take appropriate actions.

Life guarding uses the same principle as node guarding. However, with life guarding the NMT slave monitors the cyclical node guarding request of the NMT master. If the NMT slave has not been polled during its lifetime[17], a remote node error is indicated through the NMT service life guarding event (*see Section 9.4 on page 92*).

Life guarding can be disabled on the NMT slave node by setting the Guard Time (Object 100CH) or the Life Time Factor (Object 100DH) to 0.

COB-ID	DLC	Data byte 0
700H + node-ID	1	state

Table 47: Response to a node/life guarding remote frame

State	NMT state
00H	BOOT UP
04H	STOPPED

[17] See Table 51 on page 100

State	NMT state
05H	OPERATIONAL
7FH	PRE-OPERATIONAL

Table 48: Node state of a CANopen device

Bit 7 of the status byte always starts with a 0 and changes its value after each transmission. The application is responsible for actively toggling this bit. This ensures that the NMT master gets the confirmation that the application on the slave is still running.

Note

As node/life guarding uses CAN remote frames it is not recommended to use these supervision methods. Instead, the use of Heartbeat (see below) is recommended as it offers more flexibility at reduced busload.

Heartbeat

With the Heartbeat method, each node by itself transmits a dedicated Heartbeat message (Heartbeat Producer) with 1-byte data containing the NMT state of the node. No NMT master is required for using Heartbeat.

Heartbeat Producer

The Heartbeat producer cyclically sends its Heartbeat message. The *Producer Heartbeat Time* is configurable via Object 1017H (16-bit value in ms) and specifies the time between two subsequent Heartbeat messages. To disable the Heartbeat producer set Object 1017H to 0.

COB-ID	DLC	Data byte 0
700H + node-ID	1	state

Table 49: Heartbeat message

The content of the status byte corresponds to that of the Node Guarding message (*see Table 48*). Contrary to the node and/or life Guarding, bit 7 of the status byte does not toggle with each transmission. It is always set to 0.

Heartbeat Consumer

The Heartbeat Consumer receives the Heartbeat messages sent from the producer. Therefore, the supervised nodes need to get registered with its node-ID and corresponding Heartbeat time.

This information is stored in the Object Dictionary at Object 1016H containing a Subindex for each Heartbeat Consumer. Up to 5 Heartbeat Consumers are available on sysWORXX I/O modules with digital or analog outputs. Devices with inputs only do not need Heartbeat Consumers, as there is no output to set into “Fault State” in case of error. The Heartbeat Consumer is activated with the first

Heartbeat message, that has been received, and a corresponding entry is registered in the OD. If the Heartbeat time configured for a producer expires without reception of the corresponding Heartbeat message, the consumer reports a Heartbeat error event resulting in the error behavior described in *Section 9.4*.

The Heartbeat consumer is disabled when the consumer Heartbeat time is set to 0.

Object Dictionary entries

Object Index	Object type / Subindex	Object name	Data type	Object mapable	Object stored via 1010H	Object restored via 1011H
100CH	**Var**	**Guard Time**	**Unsigned16**	**-**	**X**	**X**
	00H	Guard Time	Unsigned16			
100DH	**Var**	**Life Time Factor**	**Unsigned8**	**-**	**X**	**X**
	00H	Life Time Factor	Unsigned8			
1016H	**Array**	**Consumer Heartbeat Time**	**Unsigned32**	**-**	**X**	**X**
	00H	Number of Entries	Unsigned32			
	01H-05H	Consumer Heartbeat Time	Unsigned32			
1017H	**Var**	**Producer Heartbeat Time**	**Unsigned16**	**-**	**X**	**X**
	00H	Producer Heartbeat Time	Unsigned16			

Table 50:Object Dictionary entries for device guarding

Parameter description

Parameter	Description
Guard Time	Specifies the period between the node guarding requests sent to the node in milliseconds. Default value: 00H
Life Time Factor	Specifies the number of multiplies of the Guard Time to wait for a response from the supervised node. The *Node Life Time* is the Guard Time multiplied by the Life Time Factor. If the node does not respond within the Node Life Time, then a node/life guarding error occurs (*see Section 9.4*). Default value: 00H

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Consumer Heartbeat Time</td><td>Specifies the maximum time to wait for a Heartbeat message (in milliseconds) before generating a Heartbeat error event.
The value is constructed as follows:<table><tr><th>Bit</th><th>Value</th></tr><tr><td>24..31</td><td>00H</td></tr><tr><td>16..23</td><td>Node-ID</td></tr><tr><td>0..15</td><td>Consumer Heartbeat Time</td></tr></table>Default value: 00H (disabled)
Note
The Heartbeat Consumer Time must be greater than the Heartbeat Consumer time, as there might are delays in transmission of the Heartbeat message.</td></tr>
<tr><td>Producer Heartbeat Time</td><td>Specifies the time between transmission of two Heartbeat messages in milliseconds.
Default value: 00H (disabled)</td></tr>
</table>

Table 51: Parameter description for device guarding configuration

Reference

CiA 301 V4.02

10 General technical data

10.1 Standards and certifications

Introduction

This section specifies the standards, test values and test criteria applicable to the sysWORXX CANopen I/O devices.

CE label

CE

The sysWORXX I/O devices meet the requirements and protective objectives of the following EC directives, which were published in the official pamphlets of the European Community:

73/23/EEC"	Electrical Equipment Designed for Use within Certain Voltage Limits" (low voltage directive)
89/336/EEC	"Electromagnetic Compatibility" (EMC Directive)

The EC Declaration of Conformity is available to the relevant authorities at:

SYS TEC electronic GmbH
Quality Management Dept.
August-Bebel Str. 29
D-07973 Greiz
GERMANY

CAN and CANopen standards

The sysWORXX CANopen I/O devices comply with the following standards and specifications:

CiA DR 303-1 V1.11	Cabling and Connector Pin Assignment
CiA DR 303-2 V1.1	Representation of SI Units and Prefixes
CiA DR 303-3 V1.0	Indicator Specification
CiA DS 301 V4.02	Application Layer and Communication Profile
CiA DSP 305 V1.1	Layer Setting Services and Protocol
CiA DS 401 V2.1	Device Profile for Generic I/O Modules
CiA DS 404 V1.2	Device Profile Measuring Devices and Closed Loop Controllers
ISO 11898-2	Road vehicles - Controller area network (CAN) - Part 2: High-speed medium access unit

10.2 Electromagnetic compatibility

Definition

Electromagnetic compatibility refers to the capability of electrical equipment in reliably performing its dedicated function in an electromagnetic environment, without causing interference in the same environment.

The sysWORXX CANopen I/O devices meet all requirements of EMC legislation for the European market, under the condition that the electrical configuration of the devices has been carried out in compliance with the specifications and directives respectively.

10.3 Shipping and storage conditions

Shipping and storage conditions

The specifications below apply to modules, which are shipped and stored in their original packaging.

Type of condition	Permissible range
Free fall	≤ 1m
Temperature	from –20 °C to +90 °C
Temperature fluctuation	<20 K/h
Barometric pressure	-1080 hPa to 660 hPa (corresponds with altitudes from -1000m to 3500m)
Relative humidity	<95 %, without condensation

Table 55: Shipping and storage conditions

10.4 Mechanical and climatic ambient conditions

Climatic ambient conditions

Applicable climatic ambient conditions (only indoor use):

Ambient conditions	Fields of application	Remarks
Temperature	-20 °C to 70 °C -20 °C to 50 °C (IO-X7 only)	All mounting positions
Temperature fluctuation	<10 K/h	-
Relative humidity	<95 %	without condensation

Ambient conditions	Fields of application	Remarks
Air pressure	from -1080 hPa to 795 hPa	corresponds with an altitude of -1000m to 2000m

Table 56: Climatic ambient conditions

Modules for operation in the range from -20°C to 50°C

The table below shows all modules suitable for operation in the range from -20°C to 50°C (only indoor use):

Designation	Order no.
CANopen IO-X7	3001006

Table 57: Modules suitable for commercial temperature range

Modules for operation in the range from -20°C to 70°C

The table below shows all modules suitable for operation in the range from -20°C to 70°C (only indoor use):

Designation	Order no.
CANopen IO-X1	3001000
CANopen IO-X2	3001001
CANopen IO-X3	3001002
CANopen IO-X4	3001003
CANopen IO-X5	3001004
CANopen IO-X6	3001005

Table 58: Modules suitable for extended temperature range

This side was left empty intentionally

11 Digital I/O modules

11.1 CANopen IO-X1, digital input and output module 16DI + 8DO DC 24V

Order No. and options

3001000	CANopen IO-X1 Galvanic isolated CAN
3001010	CANopen IO-X1 Galvanic isolated CAN, with pulsed output

Properties

- 16 digital inputs 24VDC, galvanic isolated in groups of 4 inputs
- 8 digital outputs 24VDC/500mA, transistor, high side switch, short-circuit protected
- 8 digital pulsed output (version 3001010 only)
- CANopen device according to CiA 401 V2.1
- 24 LEDs for I/O state indication
- Galvanic isolated CAN-bus interface
- Non-volatile storage of configuration data
- Watchdog
- CAN bus termination (120Ω resistor) via Jumper
- Separated power supply pin for supply of digital outputs

Module pinout

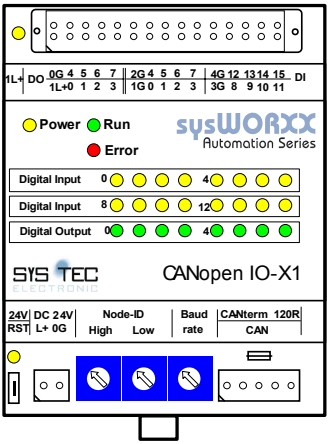


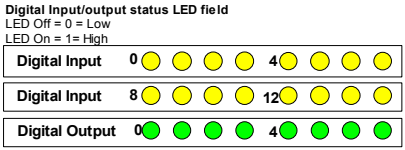
Figure 29: CANopen IO-X1 device schema

Pin	Label	Description
Power supply connector		
1*	L+	+24VDC ±20%
2	0G	Ground 0 for device power supply
CAN-bus interface connector		
1*		CAN_GND
2		CAN_L
3		n.c.
4		CAN_H
5		CAN_V+ (connected to L+ on modules without galvanic isolation, not used on modules with galvanic isolated CAN)
I/O connector		
1*	1L+	+24VDC (connected to L+)
2	0G	Ground 0 for digital outputs 0 to 7
3	0	digital output 0 24V/500mA
4	4	digital output 4 24V/500mA
5	1	digital output 1 24V/500mA
6	5	digital output 5 24V/500mA
7	2	digital output 2 24V/500mA
8	6	digital output 6 24V/500mA

Pin	Label	Description
9	3	digital output 3 24V/500mA
10	7	digital output 7 24V/500mA
11	1G	Ground 1 for digital inputs 0 to 3
13	0	digital input 0 24V to 1G
15	1	digital input 1 24V to 1G
17	2	digital input 2 24V to 1G
19	3	digital input 3 24V to 1G
12	2G	Ground 2 for digital inputs 4 to 7
14	4	digital input 4 24V to 2G
16	5	digital input 5 24V to 2G
18	6	digital input 6 24V to 2G
20	7	digital input 7 24V to 2G
21	3G	Ground 3 for digital inputs 8 to 11
23	8	digital input 8 24V to 3G
25	9	digital input 9 24V to 3G
27	10	digital input 10 24V to 3G
29	11	digital input 11 24V to 3G
22	4G	Ground 4 for digital inputs 12 to 15
24	12	digital input 12 24V to 4G
26	13	digital input 13 24V to 4G
28	14	digital input 14 24V to 4G
30	15	digital input 15 24V to 4G

Table 61: CANopen IO-X1 device pinout

LED display



Block diagram

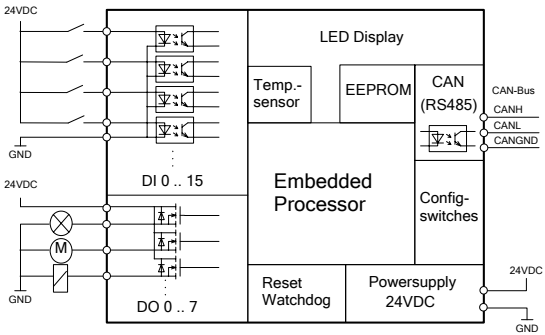


Figure 30: CANopen IO-X1 block diagram

Technical data

Common		Typical	Maximum
Power supply	V_{CPU}	24V DC	±20%
	V_{IO}	24V DC	±20%
Current consumption (I/Os inactive)	I_{CPU}	0,05A	
	I_{IO}	0,01A	
Temperature range	Storage		-20° … +90°C
	Operation		-20° … +70°C
Protection class	Enclosure	IP20	
Module weight		130g	
Dimensions	Width	71 mm	
	High	58 mm	
	Length	95 mm	
Connection scheme	Removable spring-type clamp connectors		

Table 62: CANopen IO-X1 technical data part common

Communication		Minimum	Maximum
CAN 2.0B (passive) compliant to CiA 120 and ISO 11898-2	bit rate	10kBit/s	1MBit/s
	number of nodes supported on same CAN-bus segment		110
	Isolation voltage		1kV
	CAN_H and CAN_L, short-circuit proof towards 24V DC		
	High-speed CAN-bus transceiver compliant to ISO 11898		

Table 63: CANopen IO-X1 technical data part communication

I/O		Minimum	Maximum
Digital outputs DO0..8			
24V DC output, high-side switch, transistor	U_{OH} at I_{OH} = 500mA	V_{IO}-0,16V < U_{OH} <V_{IO}	
	U_{OL} at I_{OL} = 0mA		0.5V
	Current limitation I_{OH_max}	2,8A	
	Maximum current (Polyswitch protected)		4A (at 20°C)
	I_{OL}(off)		10µA
	t_{off} at I_{OH} = 500 mA	115µs	190µs
	t_{on} at I_{OH} = 500 mA	75µs	125µs
Digital inputs DI0 .. 15			
24V DC inputs	U_{IH}	15V	30V
	U_{IL}	-3V	5V
	I_{IH} = (U_{IH}-5,6)/2700	3,5mA	9mA

Table 64: CANopen IO-X1 technical data part I/O

Manufacturer specific functions

The CANopen IO-X1 supports the following device specific manufacturer extension:

- Disable digital input 8-Bit (Object 2010H)
- Enable pulsed digital output (Object 2011H)
- Enable retrigger pulsed digital output (Object 2012H)
- Enable active off pulsed digital output (Object 2013H)
- **Pulslength pulsed digital output** (Object 2014H)

The generic manufacturer specific extensions are described in *Section 8.4*.

Error behavior

In addition to the error behavior described with *Section 9.4* the CANopen IO-X1 features a device specific error behavior for its digital outputs with the following parameters:

- Error Mode Output 8-Bit
- Error Value Output 8-Bit
- Filter Constant Output 8-Bit

Object dictionary

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restored
2010H	**Array**	**Disable digital input 8-Bit (*see Section 8.4 on page 71*)**	**Unsigned8**	**-**	**x**	**x**
2011H	**Array**	**Enable pulsed digital output**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7	Unsigned8			
2012H	**Array**	**Enable retrigger pulsed digital output**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7	Unsigned8			
2013H	**Array**	**Enable active off pulsed digital output**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7	Unsigned8			
2014H	**Array**	**Pulslength pulsed digital output**	**Unsigned8**	-	x	x
	00H	Number of Input 8-Bit	Unsigned16			
	01H	DO0_Pulselength	Unsigned16			
	02H	DO1_Pulselength	Unsigned16			
	…					
	08H	DO7_Pulselength	Unsigned16			
6000H	**Array**	**Read Digital Input 8-Bit**	**Unsigned8**	x	-	-
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7	Unsigned8			
	02H	DI8_DI15	Unsigned8			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restored
6003H	**Array**	**Filter Constant Input 8-Bit**	**Unsigned8**	-	X	X
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7_FilterConstant	Unsigned8			
	02H	DI8_DI15_FilterConstant	Unsigned8			
6005H	**Var**	**Global Interrupt Enable 8-Bit**	**Boolean**	**-**		
6006H	**Array**	**Interrupt Mask Any Change 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7_InterruptAny Change	Unsigned8			
	02H	DI8_DI15_InterruptAny Change	Unsigned8			
6007H	**Array**	**Interrupt Mask Low to High 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7_InterruptLowToHigh	Unsigned8			
	02H	DI8_DI15_InterruptLowToHigh	Unsigned8			
6008H	**Array**	**Interrupt Mask High to Low 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7_InterruptHighToLow	Unsigned8			
	02H	DI8_DI15_InterruptHighToLow	Unsigned8			
6200H	**Array**	**Write Output 8-Bit**	**Unsigned8**	**x**	**-**	**-**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7	Unsigned8			
6206H	**Array**	**Error Mode Output 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7_ErrorMode	Unsigned8			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restored
6207H	**Array**	**Error Value Output 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7_ErrorValue	Unsigned8			
6208H	**Array**	**Filter Constant Output 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7_FilterConstant	Unsigned8			

Table 65: CANopen IO-X1 Object Dictionary (Device specific part)

Parameter description

Parameter	Description
Disable digital input 8-Bit	Specifies a manufacturer specific filter for the digital inputs. The filter disables or enables specific input lines. 0 = disable 1 = enable Default value: 00H
Filter constant of digital inputs 8-Bit	Specifies whether the manufacturer specific filter is used for an input. 0 = Filter for input disable 1 = Filter for input enable Default value: 00H
Global interrupt enable 8-Bit	This parameter enables/disables the interrupt of the inputs (generating of events) globally, without changing the interrupt masks in Object 6006H, 6007H and 6008H. 1 = enable 0 = disable Default value: 00H
Interrupt mask any change 8-Bit	Specifies the input lines that generate an event upon positive and/or negative edge detection. 0 = interrupt disable 1 = interrupt enable Default value: FFH

Parameter	Description
Interrupt mask low to high 8-Bit	Specifies the input lines that generate an event upon positive edge detection. 0 = interrupt disable 1 = interrupt enable Default value: 00H
Interrupt mask high to low 8-Bit	Specifies the input lines that generate an event upon negative edge detection. 0 = interrupt disable 1 = interrupt enable Default value: 00H
Error mode output 8-Bit	Specifies whether an output is set to its pre-defined error value (*see Object 6207H*) in case of an error event (*see Section 9.4*). 0 = output value not changed 1 = output value switch to the state specified in Object 6207H Default value: 00H
Error value output 8-Bit	This parameter specifies the error value for a digital output. 0 = output shall be set to '0' 1 = output shall be set to '1' Default value: 00H
Filter Constant output 8-Bit	This parameter specifies an output filter mask for a group of 8 outputs. 0 Updating of outputs disabled. The current value is kept, even on reception of a new output value. 1 Updating of outputs enabled upon reception of new output data. Default value: FFH
Enable pulsed digital output	This parameter specifies the possibility to configure a digital output as a pulsed output. 0 = pulsed output disable 1 = pulsed output enable Default value: 00H
Enable retrigger pulsed digital output	This parameter specifies the possibility to retrigger a digital pulsed output before the digital output is switch off automaticaly. The time of the pulse is new started. 0 = retrigger disable 1 = retrigger possible Default value: 00H

Parameter	Description
Enable active off pulsed digital output	This parameter specifies the possibility to switch off a digital pulsed output before the time, specified in 2014H, is over. 0 = active switch off disable 1 = switch off possible Default value: 00H
Pulslength pulsed digital output	This parameter specifies the pulslength of a digital pulsed output. The unit ist millisecond [ms] e.g.: 100 means 100ms switch on time Default value: 00H

Table 66: CANopen IO-X1 parameter description

Default mapping of I/O

PDO	TPDO1	RPDO1
COB-ID	180H+node-ID	200H+node-ID
Mapped objects	2	1
Mapped obj 1 (data byte 0)	DI0_7 6000H/01H/08H[18]	DO0_7 6200H/01H/08H
Mapped obj.2 (data byte 1)	DI8_15 6000H/02H/08H	

Table 67: CANopen IO-X1 default mapping

[18] A mapping entry consists of: Object/Subindex/Datasize of mapped data

Device specific commissioning

The following steps list the device specific configuration, which are necessary to put the device into operation. Communication specific configuration (e.g. PDO Mapping and Linking, device guarding, ect.) is not considered here. Furthermore it is assumed that the basic commissioning (*see Section 6.1*) of the device has been finished.

When using digital inputs

(1) Configure the digital input PDO transmission triggers (Object 6006H to 6008H).
Note: Only one trigger type per channel is permitted.

(2) Enable global interrupt generation for digital inputs (Object 6005H)

(3) If required by the application, set the manufacturer specific filtering (Object 2010H) and enable these filters for the corresponding channels (Object 6003H)

When using digital outputs

(1) Enable updating of used channels (Object 6208H)

If the application requires pre-defined error values for the outputs:

(2) Configure the error value of the outputs (Object 6207H)

(3) Enable the error mode of the channels (Object 6206H)

*When using digital pulsed outputs (*Order number 3001010 only*)*

(4) Enable pulsed output of used channels (Object 2011H), e.g. 01H for channel AO0

(5) Set the **pulslength of the pulsed digital output** of used channels (Object 2014H)
e.g. set subindex 1 to 100_{dec} for AO0, pulslength 100ms

If the application requires retrigger functionality:

(6) Configure the retrigger value of the outputs (Object 2012H) e.g. 01H for AO0

If the application requires active switch off functionality:

(7) Configure the active switch off value of the outputs (Object 2013H)
e.g. 01H for AO0

Accessory

Order number	Part
171024	2 pole plug for the power supply
171023	5 pole plug with adapter cable to 9-pin D-Sub connector for CAN bus
171034	30-pin I/O connector plug
180134	Jumper for the CAN bus termination

Table 68: Accessory for CANopen IO-X1

References

CiA 303-1 V1.3

CiA 303-3 V1.2

CiA 301 V4.02

CiA 401 V2.1

11.2 CANopen IO-X2, digital input module 24DI DC 24V

Order No. and options

3001001	CANopen IO-X2 galvanic isolated CAN

Properties

- 24 digital inputs 24VDC, galvanic isolated in groups of 4 inputs
- CANopen device according to CiA 401 V2.1
- 24 LEDs for I/O state indication
- Galvanic isolated CAN-bus interface
- Non-volatile storage of configuration data
- Watchdog
- CAN bus termination (120Ω resistor) via Jumper

Module pinout

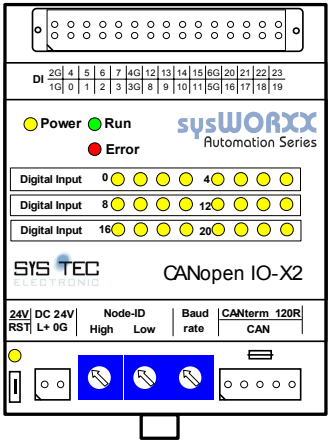


Figure 31: CANopen IO-X2 device schema

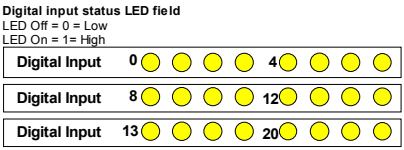
Pin	Name	Description
Power supply connector		
1*	L+	+24VDC ±20%
2	0G	Ground 0 for device power supply

Pin	Name	Description
CAN-bus interface connector		
1*		CAN_GND
2		CAN_L
3		n.c.
4		CAN_H
5		CAN_V+ (connected to L+ on modules without galvanic isolation, not used on modules with galvanic isolated CAN)
I/O connector		
1*	1G	Ground 1 for digital inputs 0 to 3
2	2G	Ground 2 for digital inputs 4 to 7
3	0	digital input 0 24V to 1G
4	4	digital input 4 24V to 2G
5	1	digital input 1 24V to 1G
6	5	digital input 5 24V to 2G
7	2	digital input 2 24V to 1G
8	6	digital input 7 24V to 2G
9	3	digital input 3 24V to 1G
10	7	digital input 8 24V to 2G
11	3G	Ground 3 for digital inputs 8 to 11
13	4G	Ground 4 for digital inputs 12 to 15
15	8	digital input 8 24V to 3G
17	12	digital input 12 24V to 4G
19	9	digital input 9 24V to 3G
12	13	digital input 13 24V to 4G
14	10	digital input 10 24V to 3G
16	14	digital input 14 24V to 4G
18	11	digital input 11 24V to 3G
20	15	digital input 15 24V to 4G
21	5G	Ground 5 for digital inputs 16 to 19
23	6G	Ground 6 for digital inputs 20 to 23
25	16	digital input 16 24V to 5G
27	20	digital input 20 24V to 6G
29	17	digital input 17 24V to 5G
22	21	digital input 21 24V to 6G

Pin	Name	Description
24	18	digital input 18 24V to 5G
26	22	digital input 22 24V to 6G
28	19	digital input 19 24V to 5G
30	23	digital input 23 24V to 6G

Table 69: CANopen IO-X2 device pinout

LED display



Block diagram

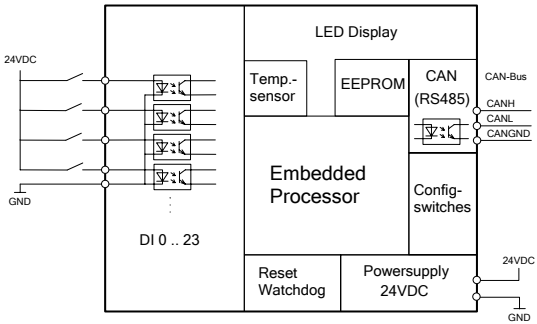


Figure 32: CANopen IO-X2 block diagram

Technical data

Common		Typical	Maximum
Power supply	V_{CPU}	24V DC	±20%
Current consumption (I/Os inactive)	I_{CPU}	0,05A	
Temperature range	Storage		-20° … +90°C
	Operation		-20° … +70°C
Protection class	Enclosure	IP20	
Module weight		130g	

Common		Typical	Maximum
Dimensions	Width	71 mm	
	High	58 mm	
	Length	95 mm	
Connection scheme	Removable spring-type clamp connectors		

Table 70: CANopen IO-X2 technical data part common

Communication		Minimum	Maximum
CAN 2.0B (passive) compliant to CiA 120 and ISO 11898-2	bit rate	10kBit/s	1MBit/s
	number of nodes supported on same CAN-bus segment		110
	Isolation voltage		1kV
	CAN_H and CAN_L, short-circuit proof towards 24V DC		
	High-speed CAN-bus transceiver compliant to ISO 11898		

Table 71: CANopen IO-X2 technical data part communication

I/O		Minimum	Maximum
Digital inputs DI0 .. 23			
24V DC-inputs	U_{InHigh}	15V	30V
	U_{InLow}	-3V	5V
	$I_{IH} = (U_{IH}-5,6)/2700$	3,5mA	9mA
	Isolation voltage for galvanic isolation		1kV

Table 72: CANopen IO-X2 technical data part I/O

Manufacturer specific functions

The CANopen IO-X2 supports the following device specific manufacturer extension:

- Disable digital input 8-Bit (Object 2010H)

The generic manufacturer specific extensions are described in *Section 8.4*.

Error behavior

The CANopen IO-X2 has no device specific error behavior. Please refer to *Section 9.4* for configuration of error behavior on communication errors.

Object dictionary

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restored
2010H	**Array**	**Disable digital input 8-Bit (*see Section 8.4 on page 71*)**	**Unsigned8**	**-**	**x**	**x**
6000H	**Array**	**Read Digital Input 8-Bit**	**Unsigned8**	x	-	-
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7	Unsigned8			
	02H	DI8_DI15	Unsigned8			
6003H	**Array**	**Filter Constant Input 8-Bit**	**Unsigned8**	-	X	X
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7_FilterConstant	Unsigned8			
	02H	DI8_DI15_FilterConstant	Unsigned8			
6005H	**Var**	**Global Interrupt Enable 8-Bit**	**Boolean**	**-**		
6006H	**Array**	**Interrupt Mask Any Change 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7_InterruptAnyChange	Unsigned8			
	02H	DI8_DI15_InterruptAnyChange	Unsigned8			
6007H	**Array**	**Interrupt Mask Low to High 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7_InterruptLowToHigh	Unsigned8			
	02H	DI8_DI15_InterruptLowToHigh	Unsigned8			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restored
6008H	**Array**	**Interrupt Mask High to Low 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Input 8-Bit	Unsigned8			
	01H	DI0_DI7_InterruptHighToLow	Unsigned8			
	02H	DI8_DI15_InterruptHighToLow	Unsigned8			

Table 73: CANopen IO-X2 Object Dictionary (Device specific part)

Parameter description

Parameter	Description
Disable digital input 8-Bit	This parameter specifies a manufacturer specific filter for the digital inputs. The filter disables or enables specific input lines. 0 = disable 1 = enable Default value: 00H
Filter constant of digital inputs 8-Bit	Specifies whether the manufacturer specific filter is used for an input block. 0 = Filter for input disable 1 = Filter for input enable Default value: 00H
Global interrupt enable 8-Bit	This parameter enables / disables the interrupt of the inputs (generating of events) globally without changing the interrupt masks in Object 6006H, 6007H and 6008H. 1 = enable 0 = disable Default value: 00H
Interrupt mask any change 8-Bit	Specifies the input lines that generate an event upon positive and/or negative edge detection. 0 = interrupt disable 1 = interrupt enable Default value: FFH
Interrupt mask low to high 8-Bit	Specifies the input lines that generate an event upon positive edge detection. 0 = interrupt disable 1 = interrupt enable Default value: 00H

Parameter	Description
Interrupt mask high to low 8-Bit	Specifies the input lines that generate an event upon negative edge detection. 0 = interrupt disable 1 = interrupt enable Default value: 00H

Table 74: CANopen IO-X2 parameter description

Default mapping of I/O

PDO	TPDO1
COB-ID	180H+node-ID
Mapped objects	3
Mapped obj 1 (data byte 0)	DI0_7 6000H/01H/08H[19]
Mapped obj.2 (data byte 1)	DI8_15 6000H/02H/08H
Mapped obj.3 (data byte 2)	DI16_23 6000H/03H/08H

Table 75: CANopen IO-X2 default mapping

Device specific commissioning

The following steps list the device specific configuration, which are necessary to put the device into operation. Communication specific configuration (e.g. PDO Mapping and Linking, device guarding, ect.) is not considered here. Furthermore it is assumed that the basic commissioning (*see Section 6.1*) of the device has been finished.

(1) Configure the digital input PDO transmission triggers (Object 6006H to 6008H).

Note

Only one trigger type per channel is permitted.

(2) Enable Global interrupt for digital inputs (Object 6005H)

If the manufacturer-specific filters are required:

(3) Set the manufacturer specific filtering (Object 2010H)

(4) Enable these filters for the corresponding channels (Object 6003H)

[19] A mapping entry consists of: Object/Subindex/Datasize of mapped data

Accessory

Order number	Part
171024	2-pin plug for the power supply
171023	5-pin plug with adapter cable to 9-pin D-Sub connector for CAN bus
171034	30-pin I/O connector plug
180134	1 jumper for the CAN bus termination

Table 76: Accessory for CANopen IO-X2

References

CiA 303-1 V1.3

CiA 303-3 V1.2

CiA 301 V4.02

CiA 401 V2.1

11.3 CANopen IO-X3, digital output module 24DO DC 24V

Order No. and options

3001002	CANopen IO-X3 galvanic isolated CAN

Properties

- 24 digital outputs 24VDC/500mA, transistor, high side switch, short circuit protected
- CANopen device according to CiA 401 V2.1
- 24 LEDs for I/O state indication
- Galvanic isolated CAN-bus interface
- Non-volatile storage of configuration data
- Watchdog
- CAN bus termination (120Ω resistor) via Jumper
- Separated power supply pin for supply of digital output groups (*see Section 2.3*)

Module pinout

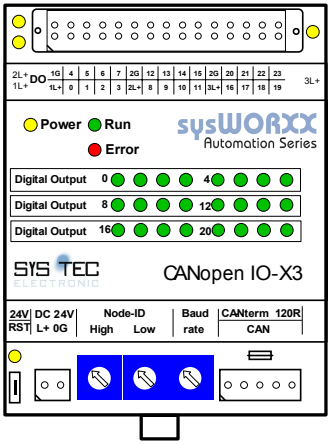


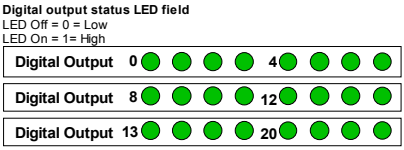
Figure 33: CANopen IO-X3 device schema

Pin	Name	Description
Power supply connector		
1*	L+	+24VDC ±20%
2	0G	Ground 0 for device power supply
CAN-bus interface connector		
1*		CAN_GND
2		CAN_L
3		n.c.
4		CAN_H
5		CAN_V+ (connected to L+ on modules without galvanic isolation, not used on modules with galvanic isolated CAN)
I/O connector		
1*	1L+	+24VDC for digital output 00 to 07 (connected to L+)
2	1G	Ground 1 for digital output 00 … 07
3	0	digital output 0
4	4	digital output 4
5	1	digital output 1
6	5	digital output 5
7	2	digital output 2
8	6	digital output 6
9	3	digital output 3
10	7	digital output 7
11	2L+	+24VDC for digital output 8 to 15 (connected to L+)
12	2G	Ground 2 for digital output 8 to 15
13	8	digital output 8
14	12	digital output 12
15	9	digital output 9
16	13	digital output 13
17	10	digital output 10
18	14	digital output 14
19	11	digital output 11
20	15	digital output 15
21	3L+	+24VDC for digital output 16 to 23 (connected to L+)

Pin	Name	Description
22	3G	Ground 3 for digital output 16 to 23
23	16	digital output 16
24	20	digital output 20
25	17	digital output 17
26	21	digital output 21
27	18	digital output 18
28	22	digital output 22
28	19	digital output 19
30	23	digital output 23

Table 77: CANopen IO-X3 device pinout

LED display



Block diagram

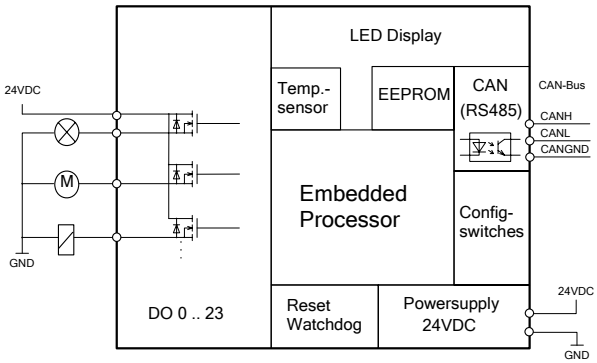


Figure 34: CANopen IO-X3 block diagram

Technical data

Common		Typical	Maximum
Power supply	V_{CPU}	24V DC	±20%
	V_{IO}	24V DC	±20%
Current consumption (I/Os inactive)	I_{CPU}	0,05A	
	I_{IO}	0,01A	
Temperature range	Storage		-20° … +90°C
	Operation		-20° … +70°C
Protection class	Enclosure	IP20	
Module weight		130g	
Dimensions	Width High Length	71 mm 58 mm 95 mm	
Connection scheme	Removable spring-type clamp connectors		

Table 78: CANopen IO-X3 technical data part common

Communication		Minimum	Maximum
CAN 2.0B (passive) compliant to CiA 120 and ISO 11898-2	bit rate	10kBit/s	1MBit/s
	number of nodes supported on same CAN-bus segment		110
	Isolation voltage		1kV
	CAN_H and CAN_L, short-circuit proof towards 24V DC		
	High-speed CAN-bus transceiver compliant to ISO 11898		

Table 79: CANopen IO-X3 technical data part communication

I/O		Minimum	Maximum
Digital outputs DO0..23			
24V DC output, high-side switch	U_{OH} at I_{OH} = 500mA	V_{IO}-0,16V < U_{OH} <V_{IO}	
	U_{OL} at I_{OL} = 0mA		0.5V
	Current limitation I_{OH_max}	2,8A	
	Maximum current per group (Polyswitch protected)		4A (at 20°C)
	I_{OL}(off)		10µA
	t_{off} at I_{OH} = 500 mA	115µs	190µs
	t_{on} at I_{OH} = 500 mA	75µs	125µs

Table 80: CANopen IO-X3 technical data part I/O

Manufacturer specific functions

The CANopen IO-X3 has no device specific manufacturer extensions. The generic manufacturer specific extensions are described in *Section 8.4*.

Error behavior

In addition to the error behavior described with *Section 9.4* the CANopen IO-X3 features a device specific error behavior for its digital outputs with the following parameters:

- Error Mode Output 8-Bit
- Error Value Output 8-Bit
- Filter Constant Output 8-Bit

Object dictionary

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restored
6200H	**Array**	**Write Output 8-Bit**	**Unsigned8**	**x**	**-**	**-**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7	Unsigned8			
6206H	**Array**	**Error Mode Output 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7_ErrorMode	Unsigned8			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restored
6207H	**Array**	**Error Value Output 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7_ErrorValue	Unsigned8			
6208H	**Array**	**Filter Constant Output 8-Bit**	**Unsigned8**	**-**	**x**	**x**
	00H	Number of Output 8-Bit	Unsigned8			
	01H	DO0_DO7_FilterConstant	Unsigned8			

Table 81: CANopen IO-X3 Object Dictionary (Device specific part)

Parameter description

Parameter	Description
Error mode output 8-Bit	Specifies whether an output is set to its pre-defined error value (*see Object 6207H*) in case of an error event(*see Section 9.4*). 0 = output value not changed 1 = output value switch to the state specified in Object 6207H Default value: 00H
Error value output 8-Bit	Specifies the error value for a group of 8 outputs. 0 = output shall be set to '0' in case of error event 1 = output shall be set to '1' in case of error event Default value: 00H
Filter Constant output 8-Bit	Specifies an output filter mask for a group of 8 outputs. 0 Updating of outputs disabled. The current value is not updated on reception of new output data. 1 Updating of outputs enabled upon reception of new output data. Default value: FFH

Table 82: CANopen IO-X3 parameter description

Default mapping of I/O

PDO	RPDO1
COB-ID	200H+node-ID
Mapped objects	3
Mapped obj 1 (data byte 0)	DO0_7 6200H/01H/08H[20]
Mapped obj.2 (data byte 1)	DO8_15 6200H/02H/08H
Mapped obj.3 (data byte 2)	DO16_23 6200H/03H/08H

Table 83: CANopen IO-X3 default mapping

Device specific commissioning

The following steps list the device specific configuration, which are necessary to put the device into operation. Communication specific configuration (e.g. PDO Mapping and Linking, device guarding, ect.) is not considered here. Furthermore it is assumed that the basic commissioning (*see Section 6.1*) of the device has been finished.

(1) Enable updating of used channels (Object 6208H)

If the application requires pre-defined error values for the outputs:

(2) Configure the error value of the outputs (Object 6207H)

(3) Enable the error mode of the channels (Object 6206H)

Accessory

Order number	Part
171024	2 pole plug for the power supply
171023	5 pole plug with adapter cable to 9-pin D-Sub connector for CAN bus
171034	30-pin I/O connector plug
180134	1 jumper for the CAN bus termination

Table 84: Accessory for CANopen IO-X3

[20] A mapping entry consists of: Object/Subindex/Datasize of mapped data

References

CiA 303-1 V1.3

CiA 303-3 V1.2

CiA 301 V4.02

CiA 401 V2.1

This side was left empty intentionally.

12 Analog I/O modules

12.1 CANopen IO-X4, analog input module 8AI U/I

Order No. and options

3001003	CANopen IO-X4 galvanic isolated CAN, 12-bit ADC

Properties

- 8 analog input separately configurable for voltage or current measurement, differential measurement
- CANopen device according to CiA 404 V1.2
- LED for I/O state indication
- Galvanic isolated CAN-bus interface
- Non-volatile storage of configuration data
- Watchdog
- CAN bus termination (120Ω resistor) via Jumper

Module pinout

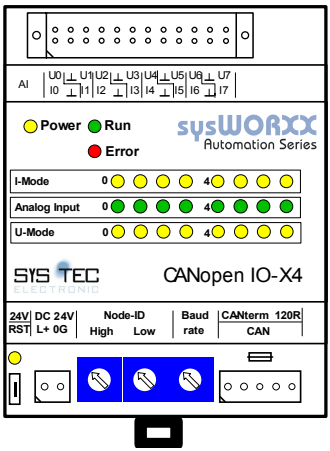


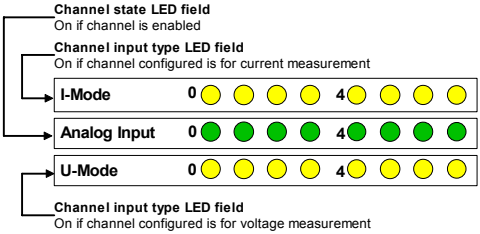
Figure 35: CANopen IO-X4 device schema

Pin	Name	Description
Power supply connector		
1*	L+	+24VDC ±20%
2	0G	Ground 0 for device power supply

Pin	Name	Description
CAN-bus interface connector		
1*		CAN_GND
2		CAN_L
3		n.c.
4		CAN_H
5		CAN_V+ (connected to L+ on modules without galvanic isolation, not used on modules with galvanic isolated CAN)
I/O connector		
1*	I0	Current input 0
2	U0	Voltage input 0
3	GND	GND channel 0
4	GND	GND channel 1
5	I1	Current input 1
6	U1	Voltage input 1
7	I2	Current input 2
8	U2	Voltage input 2
9	GND	GND channel 2
10	GND	GND channel 3
11	I3	Current input 3
12	U3	Voltage input 3
13	I4	Current input 4
14	U4	Voltage input 4
15	GND	GND channel 4
16	GND	GND channel 5
17	I5	Current input 5
18	U5	Voltage input 5
19	I6	Current input 6
20	U6	Voltage input 6
21	GND	GND channel 6
22	GND	GND channel 7
23	I7	Current input 7
24	U7	Voltage input 7

Table 85: CANopen IO-X4 device pinout

LED display



Block diagram

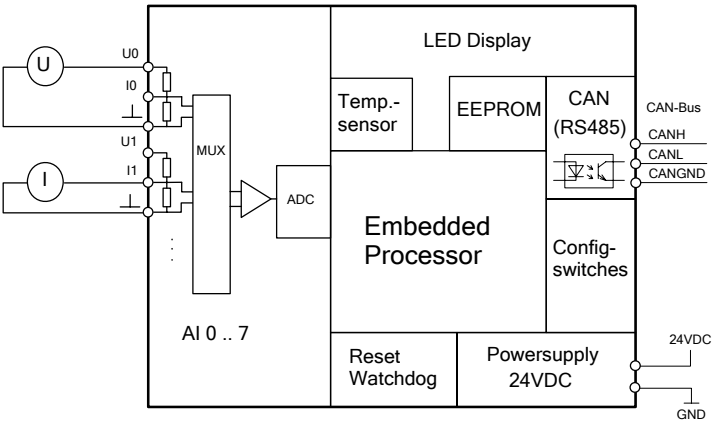


Figure 36: CANopen IO-X4 block diagram

Technical data

Common		Typical	Maximum
Power supply	V_{CPU}	24V DC	±20%
Current consumption (I/Os inactive)	I_{CPU}	0,05A	
Temperature range	Storage		-20° … +90°C
	Operation		-20° … +70°C
Protection class	Enclosure	IP20	
Module weight		130g	
Dimensions	Width High Length	71 mm 58 mm 95 mm	

Common		Typical	Maximum
Connection scheme	Removable spring-type clamp connectors		

Table 86: CANopen IO-X4 technical data part common

Communication		Minimum	Maximum
CAN 2.0B (passive) compliant to CiA 120 and ISO 11898-2	bit rate	10kBit/s	1MBit/s
	number of nodes supported on same CAN-bus segment		110
	Isolation voltage		1kV
	CAN_H and CAN_L, short-circuit proof towards 24V DC		
	High-speed CAN-bus transceiver compliant to ISO 11898		

Table 87: CANopen IO-X4 technical data part communication

I/O		Minimum	Maximum
Analog inputs AI0..7			
U-Mode (voltage input)	Input range	-10V	+10V
	Input-Resistance		22kΩ
	Offset error		3%
I-Mode (current input)	Input range	0mA	20mA
	Input resistance		195Ω
	Offset error		2%
Common	Accuracy		0,5% PE (at 12-bit)
	ADC solution	12-bit, 14-bit optional	
	Sampling rate[21]	12,5Hz (8ch)	100Hz (1ch)

Table 88: CANopen IO-X4 technical data part I/O

[21] The sampling rate decreases with the number of inputs enabled.

Manufacturer specific functions

The CANopen IO-X4 supports the following device specific manufacturer extension:

- **for Production only** (Object 2500H)

The generic manufacturer specific extensions are described in *Section 8.4*.

Error behavior

The CANopen IO-X4 has no device specific error behavior. Please refer to *Section 9.4* for configuration of error behavior on communication errors.

Object dictionary

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
2500H	**Array**	**for Production only**		**-**		
	00H	Number Of Entries	Unsigned8			
	01H	reserved	Unsigned32			
	02H	manufacture date	Unsigned32			
	03H	calibration data	Unsigned32			
	04H	pAI_0_U_Gain	Real32			
	05H	pAI_0_U_Offset	Real32			
	06H	pAI_1_U_Gain	Real32			
	07H	pAI_1_U_Offset	Real32			
	…					
	12H	pAI_7_U_Gain	Real32			
	13H	pAI_7_U_Offset	Real32			
	14H	pAI_0_I_Gain	Real32			
	15H	pAI_0_I_Offset	Real32			
	…					
	22H	pAI_7_I_Gain	Real32			
	23H	pAI_7_I_Offset	Real32			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
	24H	reserved	Unsigned8			
	25H	reserved	Unsigned8			
	26H	reserved	Unsigned8			
6110H	**Array**	**AI Sensor Type**	**Unsigned16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Sensor_Type	Unsigned16			
	…					
	07H	AI7_Sensor_Type	Unsigned16			
6112H	**Array**	**AI Operation mode**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Operation_Mode	Unsigned8			
	…					
	07H	AI7_Operation_Mode	Unsigned8			
6126H	**Array**	**AI Scaling Factor**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI Scaling Factor 0	Real32			
	…					
	07H	AI Scaling Factor 7	Real32			
6127H	**Array**	**AI Scaling Offset**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI Scaling Offset 0	Real32			
	…					
	07H	AI Scaling Offset 7	Real32			
6131H	**Array**	**AI Physical Unit PV**	**Unsigned32**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Physical_Unit_PV	Unsigned32			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
	…					
	07H	AI7_Physical_Unit_PV	Unsigned32			
6132H	**Array**	**AI Decimal Digits PV**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Decimal_Digits_PV	Unsigned8			
	…					
	07H	AI7_Decimal_Digits_PV	Unsigned8			
6150H	**Array**	**AI Status**	**Unsigned8**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Status	Unsigned8			
	…					
	07H	AI7_Status	Unsigned8			
7100H	**Array**	**AI Input FV**	**Integer16**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Input_FV	Integer16			
	…					
	07H	AI7_Input_FV	Integer16			
7130H	**Array**	**AI Input PV**	**Integer16**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Input_PV	Integer16			
	…					
	07H	AI7_Input_PV	Integer16			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
7133H	**Array**	**AI Interrupt delta Input PV**	**Integer16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Interrupt_Delta_Input_PV	Integer16			
	…					
	07H	AI7_Interrupt_Delta_Input_PV	Integer16			

Table 89: CANopen IO-X4 Object Dictionary

Parameter description

Parameter	Description
AI Sensor Type	This parameter specifies the input type/range of the channel. 41_{dec} = input type ±10V (U-mode) 42_{dec} = input type 0..10V (U-mode) 51_{dec} = input type 4..20mA (I-mode) 52_{dec} = input type 0..20mA (I-mode) Default value: 41_{dec} **Note** Each channel has separated I/O points for connection of voltage input and current inputs.
AI Operation mode	Enables/disables an input channel. 0 = Channel disabled 1 = Channel enabled (operating) Default value: 00H **Note** Each operating channel (order not important) will reduce the maximum sampling rate by apx.1/8.
AI Physical Unit PV	This parameter assigns SI units and prefixes for the process values of each channel. The coding of the physical unit and prefixes is done according to the CiA 303-2. This value just provides additional information and has no influence on process value calculation. Possible values: 00260000H = V

Parameter	Description
	FD040000H = mA Default value: 00260000H
AI Decimal Digits PV	This parameter specifies the number of decimal digits following the decimal point for interpretation of data type Integer16. Example : A process value of 1.234 V will be coded as 123 in Interger16 format if the number of decimal digits is set to 2. 0 = no decimal digits 1 = one decimal digits 2 = two decimal digits 3 = three decimal digits Default value: 02H
AI Status	This read only parameter holds the status of the analog input channel. 0 = no error 1 = measurement range underflow 2 = measurement range exceeded
AI Interrupt delta input PV	Specifies a “delta” value for triggering PDO transmission for an analog input channel. If the process value has changed for “delta” or more since the last transmission of the PDO, then the PDO is transmitted again. To disable this function set delta to 0. Default value: 00H (disabled) **Note** The entered value must have the same physical unit and number of digits as configured for the respective channel.
AI Input FV	This object contains the field value (before scaling and calibration).
AI Input PV	This object contains the process value (after scaling).
manufacture date	This object contains the manufacture date. The object is “read only” 01112007H means 1st November 2007
calibration date	This object contains the date of the last calibration. The object is “read only”. 12112007H means 12th November 2007

Parameter	Description
AI Scaling Factor	The Value “Factor” is multiply with the Processvalue.
AI Scaling Offset	The Value “Offset” is add to the Processvalue. see below

Table 90: CANopen IO-X4 parameter description

Default mapping of I/O

PDO	TPDO1	TPDO2	TPDO3	TPDO4
COB-ID	180H+ node-ID	280H+ node-ID	380H+ node-ID	480H+ node-ID
Mapped objects	4	4	4	4
Mapped object 1 (data byte 0+1)	AI0 7130H /01H/10H [22]	AI2 7130H /03H/10H	AI4 7130H /05H/10H	AI6 7130H /07H/10H
Mapped object 2 (data byte 2)	AI0 State 6150H /01H/08H	AI2 State 6150H /03H/08H	AI4 State 6150H /05H/08H	AI6 State 6150H /07H/08H
Mapped object 3 (data byte 3+4)	AI1 7130H /02H/10H	AI3 7130H /04H/10H	AI5 7130H /06H/10H	AI7 7130H /08H/10H
Mapped object 4 (data byte 5)	AI1 State 6150H /02H/08H	AI3 State 6150H /04H/08H	AI5 State 6150H /06H/08H	AI7 State 6150H /08H/08H

Table 91: CANopen IO-X4 default mapping

Relation between Fieldvalue (FV), Processvalue (PV) and Calibration

U-mode:

$PV_{bc} = FV*6{,}947*10^{-4}$

PV = (PV_{bc} *pAI_x_U_Gain+pAI_x_U_Offset)*AI Scaling Factor_x
+ AI Scaling Offset_x

I-mode:

$PV_{bc} = FV*7{,}825*10^{-7}$

PV = (PV_{bc} *pAI_x_I_Gain+pAI_x_I_Offset)*AI Scaling Factor_x
+ AI Scaling Offset_x

[22] A mapping entry consists of: Object/Subindex/Datasize of mapped data

"x" means number of AI channel

Device specific commissioning

The following steps list the device specific configuration, which are necessary to put the device into operation. Communication specific configuration (e.g. PDO Mapping and Linking, device guarding, etc.) is not considered here. Furthermore it is assumed that the basic commissioning (*see Section 6.1*) of the device has been finished.

(1) Configure the input type of each cannel (Object 6110H).

(2) Configure the number of digits used for calculation and presentation of the process value (Object 6132H) in Integer16.

(3) Set the physical unit of each channel (Object 6131H).

(4) If delta-triggered transmission of process values is needed configure the "delta" value of each channel (Object 7133).

(5) Enable the channels in use (Object 6112H).

Accessory

Order number	Part
171024	2 pole plug for the power supply
171023	5 pole plug with adapter cable to 9-pin D-Sub connector for CAN bus
171038	24-pin I/O connector plug
180134	Jumper for CAN-bus termination

Table 92: Accessory for CANopen IO-X4

References

CiA 303-1 V1.3

CiA 303-3 V1.2

CiA 301 V4.02

CiA 404 V1.2

12.2 CANopen IO-X5, analog input module 8RTD

Order No. and options

3001004	CANopen IO-X5 galvanic isolated CAN, 12-bit ADC

Properties

- 8 analog input suitable for resistor temperature devices (RTD) (e.g. PT100 or PT1000) in 2- or 3-wire connection scheme
- CANopen device according to CiA 404 V1.2
- LED for I/O state indication
- Galvanic isolated CAN-bus interface
- Non-volatile storage of configuration data
- Watchdog
- CAN bus termination (120Ω resistor) via Jumper

Module pinout

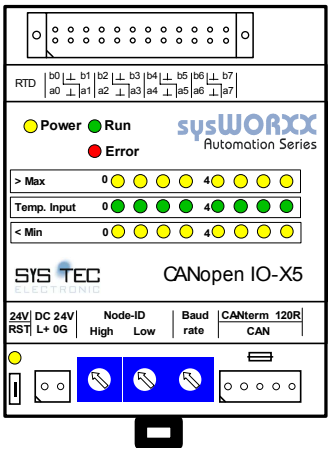


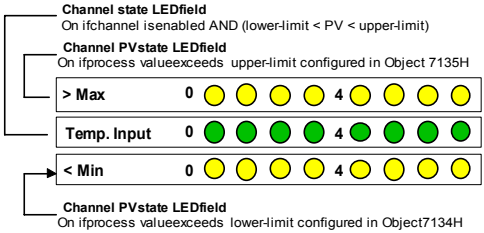
Figure 37: CANopen IO-X5 device schema

Pin	Name	Description
Power supply connector		
1*	L+	+24VDC ±20%
2	0G	Ground 0 for device power supply

Pin	Name	Description
CAN-bus interface connector		
1*		CAN_GND
2		CAN_L
3		n.c.
4		CAN_H
5		CAN_V+ (connected to L+ on modules without galvanic isolation, not used on modules with galvanic isolated CAN)
I/O connector		
1*	a0	RTD input a0
2	b0	RTD input b0
3	GND	GND RTD input 0
4	GND	GND RTD input 1
5	a1	RTD input a1
6	b1	RTD input b1
7	a2	RTD input a2
8	b2	RTD input b2
9	GND	GND RTD input 2
10	GND	GND RTD input 3
11	a3	RTD input a3
12	b3	RTD input b3
13	a4	RTD input a4
14	b4	RTD input b4
15	GND	GND RTD input 4
16	GND	GND RTD input 5
17	a5	RTD input a5
18	b5	RTD input b5
19	a6	RTD input a6
20	b6	RTD input b6
21	GND	GND RTD input 6
22	GND	GND RTD input 7
23	a7	RTD input a7
24	b7	RTD input b7

Table 93: CANopen IO-X5 device pinout

LED display



Condition	LED states	EMCY trigger
PV > Upper Limit PV	Yellow "*>MAX"* state LED (upper) on	no
Short Circuit (PV < Sensor Range Limit)	Yellow "*>MAX"* state LED (upper) blinking	yes
Channel enabled	Green *Temp. Input* LED on	no
PV < Lower Limit PV	Yellow "*<MIN"* state LED (lower) on	no
Sensor Fraction (PV > Sensor Range Limit)	Yellow *"<MIN"* state LED (lower) blinking	yes

Table 94: Device specific LED states for CANopen IO-X5

Block diagram

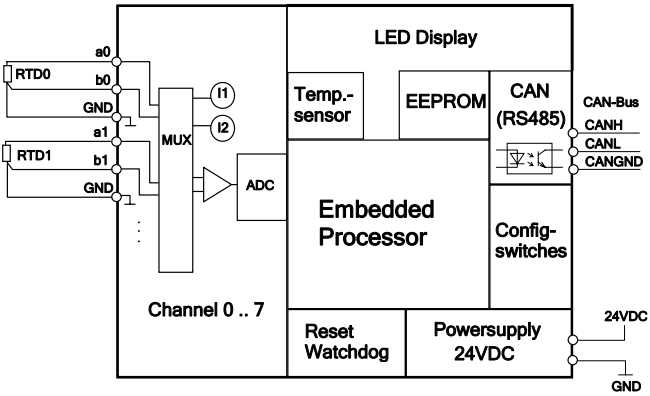


Figure 38: CANopen IO-X5 block diagram (3-wire connection)

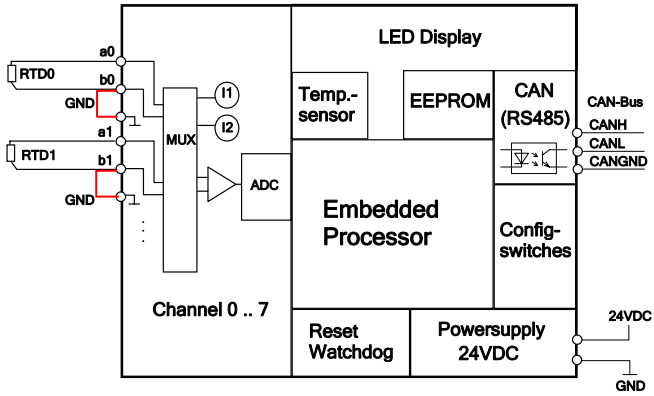


Figure 39: CANopen IO-X5 block diagram (2-wire connection)

Technical data

Common		Typical	Maximum
Power supply	V_{CPU}	24V DC	±20%
Current consumption (I/Os inactive)	I_{CPU}	0,05A	
Temperature range	Storage		-20° … +90°C
	Operation		-20° … +70°C
Protection class	Enclosure	IP20	
Module weight		130g	
Dimensions	Width High Length	71 mm 58 mm 95 mm	
Connection scheme	Removable spring-type clamp connectors		

Table 95: CANopen IO-X5 technical data part common

Communication		Minimum	Maximum
CAN 2.0B (passive) compliant to CiA 120 and ISO 11898-2	bit rate	10kBit/s	1MBit/s
	number of nodes supported on same CAN-bus segment		110
	Isolation voltage		1kV
	CAN_H and CAN_L, short-circuit proof towards 24V DC		
	High-speed CAN-bus transceiver compliant to ISO 11898		

Table 96: CANopen IO-X5 technical data part communication

I/O	Minimum	Maximum
RTD input Ch0..Ch7		
Temperature range (sensor independent)	-200°C	+600°C
	73,2 K	873,2 K
	-328,0°F	1112,0°F
ADC solution	12-bit (optional 14-bit)	
Accuracy	0,5% PE (at 12-bit)	
Resolution PV		0,1K
Sampling rate[23]	12,5Hz (8ch)	100Hz (1ch)

Table 97: CANopen IO-X5 technical data part I/O

Manufacturer specific functions

The CANopen IO-X5 supports the following device specific manufacturer extension:

- **for Production only** (Object 2500H)

The generic manufacturer specific extensions are described in *Section 8.4*.

Error behavior

The CANopen IO-X5 has no device specific error behavior. Please refer to *Section 9.4* for configuration of error behavior on communication errors.

If an input channel is switched on without a sensor connected, the module will indicate this by setting the channel's AI Status in Object 6150H to value 01H.

[23] The sampling rate decreases with the number of inputs enabled.

Object dictionary

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
2500H	**Array**	**for Production only**		**-**		
	00H	Number Of Entries	Unsigned8			
	01H	reserved	Unsigned32			
	02H	manufacture date	Unsigned32			
	03H	calibration data	Unsigned32			
	04H	pGain_PT100_0	Real32			
	05H	pOffset_PT100_0	Real32			
	06H	pGain_PT100_1	Real32			
	07H	pOffset_PT100_1	Real32			
	…					
	12H	pGain_PT100_7	Real32			
	13H	pOffset_PT100_7	Real32			
	14H	pGain_PT1000_0	Real32			
	15H	pOffset_PT1000_0	Real32			
	…					
	22H	pGain_PT1000_7	Real32			
	23H	pOffset_PT1000_7	Real32			
	24H	reserved	Unsigned8			
	25H	reserved	Unsigned8			
	26H	reserved	Unsigned8			
6110H	**Array**	**AI Sensor Type**	**Unsigned16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Sensor_Type	Unsigned16			
	…					
	07H	AI7_Sensor_Type	Unsigned16			
6110H	**Array**	**AI Sensor Type**	**Unsigned16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
	01H	AI0_Sensor_Type	Unsigned16			
	…					
	07H	AI7_Sensor_Type	Unsigned16			
6112H	**Array**	**AI Operation mode**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Operation_Mode	Unsigned8			
	…					
	07H	AI7_Operation_Mode	Unsigned8			
6126H	**Array**	**AI Scaling Factor**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI Scaling Factor 0	Real32			
	…					
	07H	AI Scaling Factor 1	Real32			
6127H	**Array**	**AI Scaling Offset**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI Scaling Offset 0	Real32			
	…					
	07H	AI Scaling Offset 1	Real32			
6131H	**Array**	**AI Physical Unit PV**	**Unsigned32**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Physical_Unit_PV	Unsigned32			
	…					
	07H	AI7_Physical_Unit_PV	Unsigned32			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
6132H	**Array**	**AI Decimal Digits PV**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Decimal_Digits_PV	Unsigned8			
	…					
	07H	AI7_Decimal_Digits_PV	Unsigned8			
6150H	**Array**	**AI Status**	**Unsigned8**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Status	Unsigned8			
	…					
	07H	AI7_Status	Unsigned8			
7100H	**Array**	**AI Input FV**	**Integer16**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Input_FV	Integer16			
	…					
	07H	AI7_Input_FV	Integer16			
7130H	**Array**	**AI Input PV**	**Integer16**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Input_PV	Integer16			
	…					
	07H	AI7_Input_PV	Integer16			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
7133H	**Array**	**AI Interrupt delta Input PV**	**Integer16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Interrupt_Delta_Input_PV	Integer16			
	…					
	07H	AI7_Interrupt_Delta_Input_PV	Integer16			
7134H	**Array**	**AI Interrupt lower limit Input PV**	**Integer16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Interrupt_Lower_Limit_Input_PV	Integer16			
	…					
	07H	AI7_Interrupt_Lower_Limit_Input_PV	Integer16			
7135H	**Array**	**AI Interrupt upper limit Input PV**	**Integer16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Interrupt_Upper_Limit_Input_PV	Integer16			
	…					
	07H	AI7_Interrupt_Upper_Limit_Input_PV	Integer16			

Table 98: CANopen IO-X5 Object Dictionary

Parameter description

Parameter	Description
AI Sensor Type	This parameter specifies the type of sensor, which is connected to the analog input. 30_{dec} = PT100 33_{dec} = PT1000 Default value: 30_{dec}

Parameter	Description
AI Operation mode	Enables/disables an input channel 0 = Channel disabled 1 = Channel enabled (operating) Default value: 0H **Note** Each active channel (the order is not important) will reduce the maximum sampling rate by apx.1/8.
AI Physical Unit PV	This parameter assigns SI units and prefixes for the process values of each channel. The coding of the physical unit and prefixes is done according to the CiA 303-2. This value just provides additional information and has no influence on process value calculation. Possible values: 00050000H = K 002D0000H = °C 00AC0000H = °F Default value: 002D0000H (°C)
AI Decimal Digits PV	This parameter specifies the number of decimal digits following the decimal point for interpretation of data type Integer16. Example : A process value of 98.2°C will be coded as 982_{dec} in Interger16 format if the number of decimal digits is set to 1 and 98_{dec} if number of decimal digits is set to 0. 0 = no decimal digits 1 = one decimal digits Default value: 1
AI Status	This read only parameter holds the status of the analog input channel. 0 = no error 1 = input not valid (e.g. sensor break, short circuit, underflow)

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>AI Interrupt delta input PV</td><td>Specifies a “delta” value for triggering PDO transmission for an analog input channel.

If the process value has changed for “delta” or more since the last transmission of the PDO, then the PDO is transmitted again.

To disable this function set delta to 0.

Default value: 10_{dec}
(corresponds to 1,0°C under default settings)

Note

The entered value must have the same physical unit and number of digits as configured for the respective channel.</td></tr>
<tr><td>AI interrupt lower limit input PV</td><td>This parameter sets the lower limit for triggering PDO transmission of an analog input channel. If the PV goes below this value, the corresponding LED on the LED display (< MIN) is switched on. Is the process value between the minimal and maximal value, no PDO is transmitted.

The Temperature range is defined as followed:

<table><tr><th>Minimum</th><th>Maximum</th></tr><tr><td>-200,0°C</td><td>+600,0°C</td></tr><tr><td>73,2 K</td><td>873,2 K</td></tr><tr><td>-328,0°F</td><td>1112,0°F</td></tr></table>
Example :
A value of 50,5°C will be coded as 505 in Integer16 format if the number of decimal digits is set to 1.

Default value: -200°C</td></tr>
<tr><td>AI interrupt upper limit input PV</td><td>This parameter sets the upper limit for triggering PDO transmission of an analog input channel. If the PV exceeds this value, the corresponding LED on the LED display (> MAX) is switched on. Is the process value between the minimal and maximal value, no PDO is transmitted.

The Temperature range is defined as followed:

<table><tr><th>Minimum</th><th>Maximum</th></tr><tr><td>-200,0°C</td><td>+600,0°C</td></tr><tr><td>73,2 K</td><td>873,2 K</td></tr><tr><td>-328,0°F</td><td>1112,0°F</td></tr></table>
Example :
A value of 328,5°C will be coded as 3285 in Integer16 format if the number of decimal digits is</td></tr>
</table>

Parameter	Description
	set to 1. Default value: -200°C
AI Input FV	This object contains the field value (before scaling and calibration).
AI Input PV	This object contains the process value (after scaling).
manufacture date	This object contains the manufacture date. The object is “read only” e.g.: 01112007H means 1st November 2007
calibration date	This object contains the date of the last calibration. The object is “read only”. e.g.: 12112007H means 12th November 2007
AI Scaling Factor AI Scaling Offset	The Value “Factor” is multiply with the Processvalue. The Value “Offset” is add to the Processvalue. see below

Table 99: CANopen IO-X5 parameter description

Default mapping of I/O

PDO	TPDO1	TPDO2	TPDO3	TPDO4
COB-ID	180H+ node-ID	280H+ node-ID	380H+ node-ID	480H+ node-ID
Mapped objects	4	4	4	4
Mapped object 1 (data byte 0+1)	AI0 7130H /01H/10H [24]	AI2 7130H /03H/10H	AI4 7130H /05H/10H	AI6 7130H /07H/10H
Mapped object 2 (data byte 2)	AI0 State 6150H /01H/08H	AI2 State 6150H /03H/08H	AI4 State 6150H /05H/08H	AI6 State 6150H /07H/08H
Mapped object 3 (data byte 3+4)	AI1 7130H /02H/10H	AI3 7130H /04H/10H	AI5 7130H /06H/10H	AI7 7130H /08H/10H
Mapped object 4 (data byte 5)	AI1 State 6150H /02H/08H	AI3 State 6150H /04H/08H	AI5 State 6150H /06H/08H	AI7 State 6150H /08H/08H

Table 100: CANopen IO-X5 default mapping

[24] A mapping entry consists of: Object/Subindex/Datasize of mapped data

Relation between Fieldvalue (FV), Processvalue (PV) and Calibration

PT100:

R_{Tbc} = FV * 1,5108*10^{-2}

R = (R_{Tbc} * pGain_PT1000_x + pOffset_PT1000_x)
*AI Scaling Factor_x + AI Scaling Offset_x

PT1000:

R_{Tbc} = FV * 1,3872*10^{-1}

R = (R_{Tbc} * pGain_PT100_x + pOffset_PT100_x)
*AI Scaling Factor_x + AI Scaling Offset_x

"x" means number of AI channel

The calculation of PV is according to DIN IEC 60751.

Device specific commissioning

The following steps list the device specific configuration, which are necessary to put the device into operation. Communication specific configuration (e.g. PDO Mapping and Linking, device guarding, ect.) is not considered here. Furthermore it is assumed that the basic commissioning (*see Section 6.1*) of the device has been finished.

(1) Configure the input/sensor type of each cannel (Object 6110H).

(2) Configure the number of digits used for the process value (Object 6132H).

(3) Set the physical unit of each channel (Object 6131H).

(4) Configure the "delta" value of each channel (Object 7133).

(5) Configure the upper and lower limit of each channel (Object 7134H and 7135H)

(6) Enable the channels in use (Object 6112H).

Accessory

Order number	Part
171024	2 pole plug for the power supply
171023	5 pole plug with adapter cable to 9-pin D-Sub connector for CAN bus
171038	24-pin I/O connector plug
180134	Jumper for the CAN bus termination

Table 101: Accessory for CANopen IO-X5

References

CiA 303-1 V1.3

CiA 303-3 V1.2

CiA 301 V4.02

CiA 404 V1.2

12.3 CANopen IO-X6, analog output module 8AO U/I

Order No. and options

3001006	CANopen IO-X6 galvanic isolated CAN, 10-bit DAC

Properties

- 8 analog output, each configurable as current or voltage output
- CANopen device according to CiA 404 V1.2
- LED for I/O state indication
- Galvanic isolated CAN-bus interface
- Non-volatile storage of configuration data
- Watchdog
- CAN bus termination (120Ω resistor) via Jumper

Module pinout

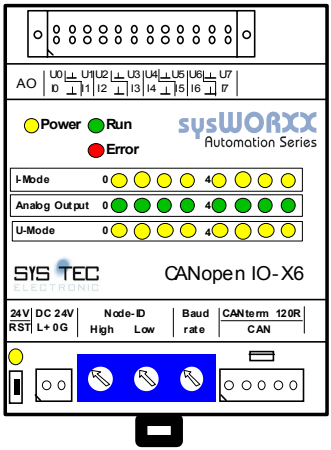


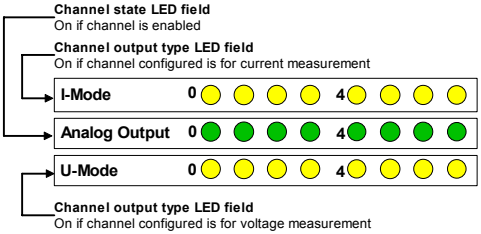
Figure 40: CANopen IO-X6 device schema

Pin	Name	Description
Power supply connector		
1*	L+	+24VDC ±20%
2	0G	Ground 0 for device power supply

Pin	Name	Description
CAN-bus interface connector		
1*		CAN_GND
2		CAN_L
3		n.c.
4		CAN_H
5		CAN_V+ (connected to L+ on modules without galvanic isolation, not used on modules with galvanic isolated CAN)
I/O connector		
1*	I0	Current output 0
2	U0	Voltage output 0
3	GND	GND channel 0
4	GND	GND channel 1
5	I1	Current output 1
6	U1	Voltage output 1
7	I2	Current output 2
8	U2	Voltage output 2
9	GND	GND channel 2
10	GND	GND channel 3
11	I3	Current output 3
12	U3	Voltage output 3
13	I4	Current output 4
14	U4	Voltage output 4
15	GND	GND channel 4
16	GND	GND channel 5
17	I5	Current output 5
18	U5	Voltage output 5
19	I6	Current output 6
20	U6	Voltage output 6
21	GND	GND channel 6
22	GND	GND channel 7
23	I7	Current output 7
24	U7	Voltage output 7

Table 102: CANopen IO-X6 device pinout

LED display



Block diagram

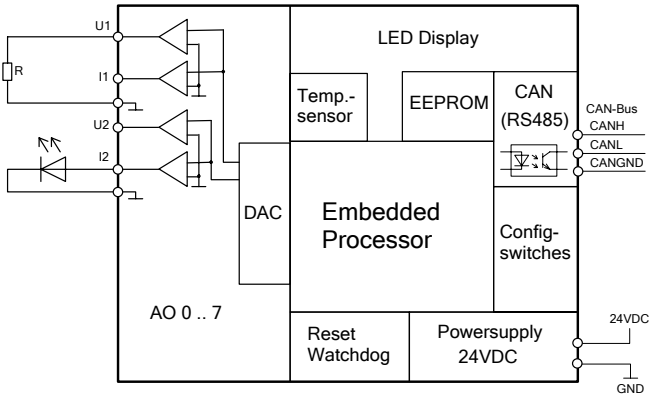


Figure 41: CANopen IO-X6 block diagram

Technical data

Common		Typical	Maximum
Power supply	V_{CPU}	24V DC	±20%
Current consumption (I/Os inactive)	I_{CPU}	0,09A	
Temperature range	Storage		-20° … +90°C
	Operation		-20° … +70°C
Protection class	Enclosure	IP20	
Module weight		130g	
Dimensions	Width High Length	71 mm 58 mm 95 mm	

Common		Typical	Maximum
Connection scheme	Removable spring-type clamp connectors		

Table 103: CANopen IO-X6 technical data part common

Communication		Minimum	Maximum
CAN 2.0B (passive) compliant to CiA 120 and ISO 11898-2	bit rate	10kBit/s	1MBit/s
	number of nodes supported on same CAN-bus segment		110
	Isolation voltage		1kV
	CAN_H and CAN_L, short-circuit proof towards 24V DC		
	High-speed CAN-bus transceiver compliant to ISO 11898		

Table 104: CANopen IO-X6 technical data part communication

I/O		Minimum	Maximum
Analog outputs AO0 .. AO7			
Voltage	Output range	0V	+10V
	Load Resistance	500Ω	
	DAC solution	10-bit (optional 12-bit)	
	Accuracy	0,5% (at 10-bit)	
	Zero-scale offset		120mV
	Settling time per channel		3µs
Current	Output range	0mA	+20mA
	Load Resistance	0Ω	500Ω
	DAC solution	10-bit (optional 12-bit)	
	Accuracy	0,5% (at 10-bit)	
	Zero-scale offset		0,3mA
	Settling time per channel		3µs

Table 105: CANopen IO-X6 technical data part I/O

Manufacturer specific functions

The CANopen IO-X6 supports the following device specific manufacturer extension:

- **Channel Calibration** (Object 2400H)
- **for Production only** (Object 2500H)

The generic manufacturer specific extensions are described in *Section 8.4*.

Error behavior

In addition to the error behavior described with *Section 9.4* the CANopen IO-X6 features a device specific error behavior for its analog outputs with the following parameters:

- AO Fault mode
- AO Fault FV

Object dictionary

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
2400H	**Array**	**Channel Calibration**		**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AO0_Gain	Real32			
	02H	AO0_Offset	Real32			
	03H	AO1_Gain	Real32			
	04H	AO1_Offset	Real32			
	…					
	0FH	AO7_Gain	Real32			
	10H	AO7_Offset	Real32			
2500H	**Array**	**for Production only**		**-**		
	00H	Number Of Entries	Unsigned8			
	01H	reserved	Unsigned32			
	02H	manufacture date	Unsigned32			
	03H	calibration data	Unsigned32			
	04H	pAO_0_U_Gain	Real32			
	05H	pAO_0_U_Offset	Real32			
	06H	pAO_1_U_Gain	Real32			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
	07H	pAO_1_U_Offset	Real32			
	…					
	12H	pAO_7_U_Gain	Real32			
	13H	pAO_7_U_Offset	Real32			
	14H	pAO_0_I_Gain	Real32			
	15H	pAO_0_I_Offset	Real32			
	…					
	22H	pAO_7_I_Gain	Real32			
	23H	pAO_7_I_Offset	Real32			
	24H	reserved	Unsigned8			
	25H	reserved	Unsigned8			
	26H	reserved	Unsigned8			
6301H	Array	AO Physical unit PV	Unsigned32	-	x	x
	00H	Number Of Entries	Unsigned8			
	01H	AO0_Physical_ Unit_PV	Unsigned32			
	…					
	07H	AO7_Physical_ Unit_PV	Unsigned32			
6302H	Array	AO Decimal digits PV	Unsigned8	-	x	x
	00H	Number Of Entries	Unsigned8			
	01H	AO0_Decimal_ Digits_PV	Unsigned8			
	…					
	07H	AO7_Decimal_ Digits_PV	Unsigned8			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
6310H	Array	AO Output Type	Unsigned16	-	x	x
	00H	Number Of Entries	Unsigned8			
	01H	AO0_Output_Type	Unsigned16			
	…					
	07H	AO7_Output_Type	Unsigned16			
6340H	Array	AO Fault mode	Unsigned8	-	x	x
	00H	Number Of Entries	Unsigned8			
	01H	AO0_Fault_Mode	Unsigned8			
	…					
	07H	AO7_Fault_Mode	Unsigned8			
7300H	Array	AO Output PV	Integer16	x	-	-
	00H	Number Of Entries	Unsigned8			
	01H	AO0_Output_PV	Integer16			
	…					
	07H	AO7_Output_PV	Integer16			
7341H	Array	AO Fault FV	Integer16	-	x	x
	00H	Number Of Entries	Unsigned8			
	01H	AO0_Fault_Value	Integer16			
	…					
	07H	AO7_Fault_Value	Integer16			

Table 106: CANopen IO-X6 Object Dictionary

Parameter description

Parameter	Description
AO Physical Unit PV	This parameter assigns SI units and prefixes for the process values of each channel. The coding of the physical unit and prefixes is done according to the CiA 303-2. This value just provides additional information and has no influence on process value calculation. Possible values: 00260000H = V FD040000H = mA Default value: 00260000H
AO Decimal Digits PV	Specifies the number of decimal digits following the decimal point for interpretation of data type Integer16. 0 = no decimal digits 1 = one decimal digits 2 = two decimal digits 3 = three decimal digits (12-bit resolution only) Default value: 02H Example : A process value of 1.234 V will be coded as 123 in Interger16 format if the number of decimal digits is set to 2.
AO Output type	Specifies the analog output type. 00_{dec} = disabled 12_{dec} = 0..10V 21_{dec} = 4..20mA 23_{dec} = 0..20mA Default value: 00_{dec}
AO Fault mode	Specifies whether an output is set to its pre-defined error value (see Object 6207H) in case of an error event (*see Section 9.4*). 0 = output value reset 1 = output value shall take the pre-defined error value specified in Object 7341H
AO Fault FV	Specifies the value that an output channel shall be set to in case of an error event (*see Section 9.4*). You have to set as Fieldvalue: e.g. U-Mode, Faultvalue should set to 2VDC. FV = 2V * 3048,09 = 6096 I-Mode, Faultvalue should set to 4mA. FV = 4mA * 1310,68 = 5243
AO Output PV	This object holds the current process values.
manufacture	This object contains the manufacture date. The object is

Parameter	Description
date	"read only" e.g.: 01112007H means 1st November 2007
calibration date	This object contains the date of the last calibration. The object is "read only". e.g.: 12112007H means 12th November 2007
Channel Calibration	Output Value (U_{out} or I_{out}) is the result of the following: The Value "Gain" is multiply with the Processvalue. The Value "Offset" is add to the Processvalue. see below

Table 107: CANopen IO-X6 parameter description

Default mapping of I/O

PDO	RPDO1	RPDO1
COB-ID	200H+node-ID	300H+node-ID
Mapped objects	4	4
Mapped object 1 (data byte 0+1)	AO0_PV 7300H/01H/10H	AO4_PV 7300H/05H/10H
Mapped object 2 (data byte 2+3)	AO1_PV 7300H/02H/10H	AO5_PV 7300H/06H/10H
Mapped object 3 (data byte 4+5)	AO2_PV 7300H/03H/10H	AO6_PV 7300H/07H/10H
Mapped object 4 (data byte 6+7)	AO3_PV 7300H/04H/10H	AO7_PV 7300H/08H/10H

Table 108: CANopen IO-X6 default mapping

Relation between Fieldvalue (FV), Processvalue (PV) and Calibration

U-mode:

$FV_U = PV*3048,09$

$U_{out}=(PV*pAO\_x\_U\_Gain+pAO\_x\_U\_Offset)*AOx\_Gain+AOx\_Offset$

I-mode:

$FV_I = PV*1310,68$

$I_{out} = (PV*pAO\_x\_I\_Gain+pAO\_x\_I\_Offset)*AOx\_Gain+AOx\_Offset$

"x" means number of AO channel

Device specific commissioning

The following steps list the device specific configuration, which are necessary to put the device into operation. Communication specific configuration (e.g. PDO Mapping and Linking, device guarding, ect.) is not considered here. Furthermore it is assumed that the basic commissioning (*see Section 6.1*) of the device has been finished.

(1) Configure the output sensor type of each cannel (Object 6310H).

(2) Configure the number of digits used for the process value (Object 6302H).

(3) Set the physical unit of each channel (Object 6301H).

If the application requires pre-defined error-values:

(4) Configure the error value for each channel (Object 7341H).

(5) Enable the fault mode for each channel that has an error value (Object 6340H)

Accessory

Order number	Part
171024	2 pole plug for the power supply
171023	5 pole plug with adapter cable to 9-pin D-Sub connector for CAN bus
171038	24-pin I/O connector plug
180134	Jumper for the CAN bus termination

Table 109: Accessory for CANopen IO-X6

References

CiA 303-1 V1.3

CiA 303-3 V1.2

CiA 301 V4.02

CiA 404 V1.2

12.4 CANopen IO-X7, analog input module 8TC

Order No. and options

3001006	CANopen IO-X7 galvanic isolated CAN, 12-bit ADC
3001008	CANopen IO-X7 galvanic isolated CAN, 14-bit ADC

Properties

- 8 analog input suitable for various types of thermocouple elements
- CANopen device according to CiA 404 V1.2
- LED for I/O state indication
- Galvanic isolated CAN-bus interface
- Non-volatile storage of configuration data
- Watchdog
- CAN bus termination (120Ω resistor) via Jumper

Module pinout

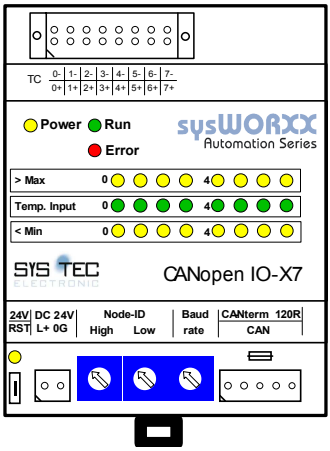


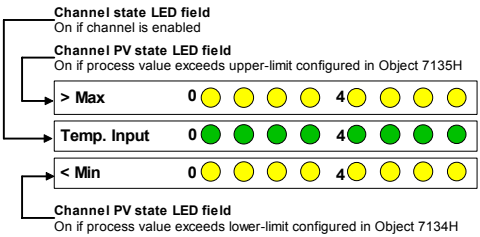
Figure 42: CANopen IO-X7 device schema

Pin	Name	Description
Power supply connector		
1*	L+	+24VDC ±20%

Pin	Name	Description
2	0G	Ground 0 for device power supply
CAN-bus interface connector		
1*		CAN_GND
2		CAN_L
3		n.c.
4		CAN_H
5		CAN_V+ (connected to L+ on modules without galvanic isolation, not used on modules with galvanic isolated CAN)
I/O connector		
1*	0+	thermocouple input 0+
2	0-	thermocouple input 0-
3	1+	thermocouple input 1+
4	1-	thermocouple input 1-
5	2+	thermocouple input 2+
6	2-	thermocouple input 2-
7	3+	thermocouple input 3+
8	3-	thermocouple input 3-
9	4+	thermocouple input 4+
10	4-	thermocouple input 4-
11	5+	thermocouple input 5+
12	5-	thermocouple input 5-
13	6+	thermocouple input 6+
14	6-	thermocouple input 6-
15	7+	thermocouple input 7+
16	7-	thermocouple input 7-

Table 110: CANopen IO-X7 device pinout

LED display



Condition	LED states	EMCY trigger
PV > Upper Limit PV	Yellow "*>MAX"* state LED (upper) on	no
PV > Sensor Range Limit	Yellow "*>MAX"* state LED (upper) blinking	yes
Channel enabled	Green *"Temp. Input"* LED on	no
PV < Lower Limit PV	Yellow *"<MIN"* state LED (lower) on	no
Sensor Fraction (PV < own temperature)	Yellow *"<MIN"* state LED (lower) blinking	yes

Table 111: Device specific LED states for CANopen IO-X7

Block diagram

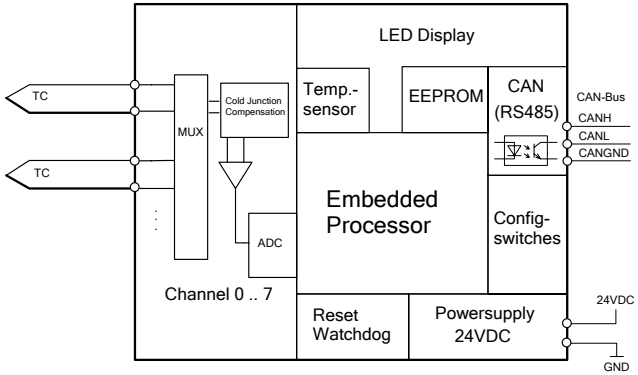


Figure 43: CANopen IO-X7 block diagram

Technical data

Common		Typical	Maximum
Power supply	V_{CPU}	24V DC	±20%
Current consumption (I/Os inactive)	I_{CPU}	0,07A	
Temperature range	Storage		-20° … +90°C
	Operation		0° … +70°C
Protection class	Enclosure	IP20	
Module weight		130g	
Dimensions	Width High Length	71 mm 58 mm 95 mm	
Connection scheme	Removable spring-type clamp connectors		

Table 112: CANopen IO-X7 technical data part common

Communication		Minimum	Maximum
CAN 2.0B (passive) compliant to CiA 120 and ISO 11898-2	bit rate	10kBit/s	1MBit/s
	number of nodes supported on same CAN-bus segment		110
	Isolation voltage[25]		1kV
	CAN_H and CAN_L, short-circuit proof towards 24V DC		
	High-speed CAN-bus transceiver compliant to ISO 11898		

Table 113: CANopen IO-X7 technical data part communication

[25] Only on modules with galvanic isolated CAN-bus interface.

I/O		Minimum	Maximum
Input Ch0 .. Ch7			
Supported sensor types		J, K, L, R, S, T, E	
Input range for type E,J,K,L,T		0V	75mV
Input range for type R,S		0V	25mV
Temperature range for sensor type:	J	50°C	1200°C
	K	50°C	1372°C
	L	50°C	900°C
	R	50°C	1768°C
	S	50°C	1768°C
	T	50°C	400°C
	E	50°C	1000°C
Over voltage protection			5V
Sampling rate per channel		12,5Hz (8 channels)	100Hz (1 channel)
ADC solution		12-bit (optional 14-bit)	
Gain factor for type E,J,K,L,T		34,33	
Gain factor for type R,S		101	
Accuracy		≤ 0,5% (at 12-bit)	
Resolution PV			0,1K (at 12-bit)

Table 114: CANopen IO-X7 technical data part I/O

Manufacturer specific functions

The CANopen IO-X7 supports the following device specific manufacturer extension:

- **for Production only** (Object 2500H)

The CANopen IO-X7 has no device specific manufacturer extensions. The generic manufacturer specific extensions are described in *Section 8.4*.

Error behavior

The CANopen IO-X7 has no device specific error behavior. Please refer to *Section 9.4* for configuration of error behavior on communication errors.

If an input channel is switched on without a sensor connected (or sensor break), the module will indicate this by setting the corresponding Subindex of Object 6150H to value 01H.

Object dictionary

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
2500H	**Array**	**for production only**		**-**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
	01H	password	Unsigned32			
	02H	manufacture date	Unsigned32			
	03H	calibration date	Unsigned32			
	04H	AI_0_EJKLT_Gain	Real32			
	05H	AI_0_EJKLT_Offset	Real32			
	06H	AI_1_EJKLT_Gain	Real32			
	07H	AI_1_EJKLT_Offset	Real32			
	…					
	12H	AI_7_EJKLT_Gain	Real32			
	13H	AI_7_EJKLT_Offset	Real32			
	14H	AI_0_RS_Gain	Real32			
	15H	AI_0_RS_Offset	Real32			
	…					
	22H	AI_7_RS_Gain	Real32			
	23H	AI_7_RS_Offset	Real32			
	24H	reserved	Unsigned8			
	25H	reserved	Unsigned8			
	26H	reserved	Unsigned8			
6110H	**Array**	**AI Sensor Type**	**Unsigned16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Sensor_Type	Unsigned16			
	…					
	07H	AI7_Sensor_Type	Unsigned16			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
6112H	**Array**	**AI Operation mode**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Operation_Mode	Unsigned8			
	…					
	07H	AI7_Operation_Mode	Unsigned8			
6126H	**Array**	**AI Scaling Factor**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI_Scaling_Factor_0	Real32			
	…					
	07H	AI_Scaling_Factor_7	Real32			
6127H	**Array**	**AI Scaling Offset**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI_Scaling_Offset_0	Real32			
	…					
	07H	AI_Scaling_Offset_7	Real32			
6131H	**Array**	**AI Physical Unit PV**	**Unsigned32**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Physical_Unit_PV	Unsigned32			
	…					
	07H	AI7_Physical_Unit_PV	Unsigned32			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
	Array	**AI Decimal Digits PV**	**Unsigned8**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
6132H	01H	AI0_Decimal_Digits_PV	Unsigned8			
	…					
	07H	AI7_Decimal_Digits_PV	Unsigned8			
	Array	**AI Status**	**Unsigned8**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
6150H	01H	AI0_Status	Unsigned8			
	…					
	07H	AI7_Status	Unsigned8			
	Array	**AI Input FV**	**Integer16**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
7100H	01H	AI0_Input_FV	Integer16			
	…					
	07H	AI7_Input_FV	Integer16			
	Array	**AI Input PV**	**Integer16**	**X**	**-**	**-**
	00H	Number Of Entries	Unsigned8			
7130H	01H	AI0_Input_PV	Integer16			
	…					
	07H	AI7_Input_PV	Integer16			

Object Index	Object type	Object name	Data type	Object mapable	Object stored	Object restore
7133H	**Array**	**AI Interrupt delta Input PV**	**Integer16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Interrupt_Delta_Input_PV	Integer16			
	…					
	07H	AI7_Interrupt_Delta_Input_PV	Integer16			
7134H	**Array**	**AI Interrupt lower limit Input PV**	**Integer16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Interrupt_Lower_Limit_Input_PV	Integer16			
	…					
	07H	AI7_Interrupt_Lower_Limit_Input_PV	Integer16			
7135H	**Array**	**AI Interrupt upper limit Input PV**	**Integer16**	**-**	**X**	**X**
	00H	Number Of Entries	Unsigned8			
	01H	AI0_Interrupt_Upper_Limit_Input_PV	Integer16			
	…					
	07H	AI7_Interrupt_Upper_Limit_Input_PV	Integer16			

Table 115: CANopen IO-X7 Object Dictionary

Parameter description

Parameter	Description
AI Sensor Type	Specifies the type of sensor, which is connected to the input channel. 0_{dec} = no sensor is connected / disabled 1_{dec} = thermocouple sensor type J 2_{dec} = thermocouple sensor type K 3_{dec} = thermocouple sensor type L 5_{dec} = thermocouple sensor type R 6_{dec} = thermocouple sensor type S

Parameter	Description
	7_{dec} = thermocouple sensor type T 8_{dec} = thermocouple sensor type E Default value: 0_{dec}
AI Operation mode	Enables/disables an input channel 0 = Channel disabled 1 = Channel enabled (operating) Default value: 00H
AI Physical Unit PV	This parameter assigns SI units and prefixes for the process values of each channel. The coding of the physical unit and prefixes is done according to the CiA 303-2. This value just provides additional information and has no influence on process value calculation. Possible values: 00050000H = K 002D0000H = °C 00AC0000H = °F Default value: 002D0000H (°C)
AI Decimal Digits PV	Specifies the number of decimal digits following the decimal point for interpretation of data type Integer16. Example : A process value of 98.2°C will be coded as 982_{dec} in Interger16 format if the number of decimal digits is set to 1 and 98_{dec} if number of decimal digits is set to 0. 0 = no decimal digits 1 = one decimal digits Default value: 1
AI Status	This read only parameter holds the status of the analog input channel. 0 = no error 1 = sensor break 2 = measurement range exceeded
AI Interrupt delta input PV	Specifies a “delta” value for triggering PDO transmission for an analog input channel. If the process value has changed for “delta” or more since the last transmission of the PDO, then the PDO is transmitted again. To disable this function set delta to 0. Default value: 10_{dec} (corresponds to 1.0°C under default settings) **Note**

Parameter	Description
	The entered value must have the same physical unit and number of digits as configured for the respective channel.
AI interrupt lower limit input PV	This parameter sets the lower limit for triggering PDO transmission of an analog input channel. If the PV goes below this value, the corresponding LED on the LED display (< MIN) is switched on. Is the process value between the minimal and maximal value, no PDO is transmitted. **Note** The temperature range depends on the sensor type. There is no internal checking whether the configured range exceeds the selected sensor type or not! Example : A value of 50.5°C will be coded as 505_{dec} in Integer16 format if the number of decimal digits is set to 1.
AI interrupt upper limit input PV	This parameter sets the upper limit for triggering PDO transmission of an analog input channel. If the PV exceeds this value, the corresponding LED on the LED display (> MAX) is switched on. Is the process value between the minimal and maximal value, no PDO is transmitted. **Note** The temperature range depends on the sensor type. There is no internal checking whether the configured range exceeds the selected sensor type or not! Example : A value of 528,5°C will be coded as 5285_{dec} in Integer16 format if the number of decimal digits is set to 1.
AI Input FV	This object contains the field value (before scaling).
manufacture date	This object contains the manufacture date. The object is “read only” e.g.: 01112007H means 1st November 2007
calibration date	This object contains the date of the last calibration. The object is “read only”. e.g.: 12112007H means 12th November 2007
Channel Calibration	Output Value (PV) is the result of the following: The Value “Gain” is multiply with the Fieldvalue. The Value “Offset” is add to the Fieldvalue.

Parameter	Description
	see below
AI Scaling Factor AI Scaling Offset	The Value “Factor” is multiply with the Fieldvalue. The Value “Offset” is add to the Fieldvalue. see below
AI Input PV	This object contains the process value (after scaling).

Table 116: CANopen IO-X7 parameter description

Default mapping of I/O

PDO	TPDO1	TPDO2	TPDO3	TPDO4
COB-ID	180H+ node-ID	280H+ node-ID	380H+ node-ID	480H+ node-ID
Mapped objects	4	4	4	4
Mapped object 1 (data byte 0+1)	AI0 7130H /01H/10H	AI2 7130H /03H/10H	AI4 7130H /05H/10H	AI6 7130H /07H/10H
Mapped object 2 (data byte 2)	AI0 State 6150H /01H/08H	AI2 State 6150H /03H/08H	AI4 State 6150H /05H/08H	AI6 State 6150H /07H/08H
Mapped object 3 (data byte 3+4)	AI1 7130H /02H/10H	AI3 7130H /04H/10H	AI5 7130H /06H/10H	AI7 7130H /08H/10H
Mapped object 4 (data byte 5)	AI1 State 6150H /02H/08H	AI3 State 6150H /04H/08H	AI5 State 6150H /06H/08H	AI7 State 6150H /08H/08H

Table 117: CANopen IO-X7 default mapping

Relation between Fieldvalue (FV), Processvalue (PV) and Calibration

Thermocouple sensor type E, J, K, L and T:

$FV1 = (ADC_{value} * AI\_EJKLT\_Gain\_x + AI\_EJKLT\_Offset\_x)$
*AI Scaling Factor_x + AI Scaling Offset_x

FV = FV1 * 2.222154[µV/Digit]

thermocouple sensor type R and S:

$FV1 = (ADC_{value} * AI\_RS\_Gain\_x + AI\_RS\_Offset\_x)$
*AI Scaling Factor_x + AI Scaling Offset_x

FV = FV1 * 0.755386[µV/Digit]

"x" means number of AI channel

The calculation of PV is according to IEC 584-1:1995.

Device specific commissioning

The following steps list the device specific configuration, which are necessary to put the device into operation. Communication specific configuration (e.g. PDO Mapping and Linking, device guarding, ect.) is not considered here. Furthermore it is assumed that the basic commissioning (*see Section 6.1*) of the device has been finished.

(1) Configure the sensor type of each cannel (Object 6110H).

(2) Configure the number of digits used for the process value (Object 6132H).

(3) Set the physical unit of each channel (Object 6131H).

(4) If required, configure the "delta" value of each channel (Object 7133).

(5) If required, configure the upper and lower limit of each channel (Object 7134H and 7135H)

(6) Enable the channels in use (Object 6112H).

Accessory

Order number	Part
171024	2 pole plug for the power supply
171023	5 pole plug with adapter cable to 9-pin D-Sub connector for CAN bus
171036	16-pin I/O connector plug
180134	Jumper for the CAN bus termination

Table 118: Accessory for CANopen IO-X7

References

CiA 303-1 V1.3

CiA 303-3 V1.2

CiA 301 V4.02

CiA 404 V1.2

13 Appendix

13.1 Conversation table of node-IDs

The following table shows a conversion of decimal node-IDs into hexadecimal format.

Node-ID decimal	Node-ID hex	Node-ID decimal	Node-ID hex	Node-ID decimal	Node-ID hex	Node-ID decimal	Node-ID hex
1	1	33	21	65	41	97	61
2	2	34	22	66	42	98	62
3	3	35	23	67	43	99	63
4	4	36	24	68	44	100	64
5	5	37	25	69	45	101	65
6	6	38	26	70	46	102	66
7	7	39	27	71	47	103	67
8	8	40	28	72	48	104	68
9	9	41	29	73	49	105	69
10	A	42	2A	74	4A	106	6A
11	B	43	2B	75	4B	107	6B
12	C	44	2C	76	4C	108	6C
13	D	45	2D	77	4D	109	6D
14	E	46	2E	78	4E	110	6E
15	F	47	2F	79	4F	111	6F
16	10	48	30	80	50	112	70
17	11	49	31	81	51	113	71
18	12	50	32	82	52	114	72
19	13	51	33	83	53	115	73
20	14	52	34	84	54	116	74
21	15	53	35	85	55	117	75
22	16	54	36	86	56	118	76
23	17	55	37	87	57	119	77
24	18	56	38	88	58	120	78
25	19	57	39	89	59	121	79
26	1A	58	3A	90	5A	122	7A
27	1B	59	3B	91	5B	123	7B
28	1C	60	3C	92	5C	124	7C
29	1D	61	3D	93	5D	125	7D
30	1E	62	3E	94	5E	126	7E
31	1F	63	3F	95	5F	127	7F

Node-ID decimal	Node-ID hex	Node-ID decimal	Node-ID hex	Node-ID decimal	Node-ID hex	Node-ID decimal	Node-ID hex
32	20	64	40	96	60		

Table 119: Conversion table from decimal to hexadecimal Node-ID

13.2 Troubleshooting

Warning limit/Bus off indicated on one or more modules

Error Frames

One sign of errors in the CAN wiring, the address assignment or the setting of the bit rate is an increased number of error frames: the diagnostic LED on the module then indicates Warning Limit exceeded or Bus-off state entered (*see Section 9.1*).

Note

Warning limit exceeded or bus-off state are indicated first of all at those nodes that have detected the most errors. These nodes, however, are not necessarily the cause for the occurrence of error frames!

If, for instance, one node causes unusual heavy bus traffic (e.g. analog inputs which trigger event-driven PDOs at a high rate), then the probability of its telegrams being damaged increases. Its error counter witch be the first one, reaching a critical level.

node-ID and bit rate settings

Make sure that node-IDs were not assigned twice: to make sure there is only one producer for each CAN data telegram (when Pre-Defined Connection Set is used.).

Test 1

Check node-IDs. If the CAN communication works partially and all the devices support the boot up message, then the node-ID assignment can also be checked by recording the boot up messages after resetting the modules. However, this will not work on node-IDs that have been swapped.

Check that the same bit rate has been set on each node.

Testing the CAN wiring

Warning

Proceed with the following test steps while the network is active - communication should not take place during the tests.

The following tests should be carried out in the stated sequence, because some of the tests assume that the previous test has been completed successfully. Not all the tests are generally necessary.

Network terminator and signal lines

Test 2
For this test the nodes must be switched off or the CAN cable unplugged. Otherwise the measured results may be distorted by an active CAN transceiver. Proceed with measuring the resistance between CAN_High and CAN_Low at - each device, if necessary. If the measured value is higher than 65Ω, it indicates the absence of a terminating resistor or a break in a signal lead. If the measured value is less than 5Ω, look for a short circuit between the CAN lines, more than the correct number of terminating resistors, or faulty transceivers.

Test 3
Check for a short circuit between the CAN ground and the signal lines, or between the screen (shield) and signal lines.

Test 4
Remove the earth connection from the CAN ground and screen. Check for a short circuit between the CAN ground and screen.

Topology

The possible cable length in CAN networks depends heavily on the selected bit rate. CAN usually tolerates short drop lines. The maximum permitted length of drop lines should not be exceeded (*see Section 4.3*). The length of the cable installed on the field is often subject to estimating errors. The following test is therefore recommended to be performed!

Test 5
Measure the lengths of the drop lines and the total bus lengths (do not just make rough estimates!) and compare them with the topology rules for the relevant bit rate.

Screening and grounding

Test 6
The power supply and the screen should be carefully earthed at the power supply unit, with a single joint only (star-shaped) and with low resistance. At all connecting points, branches and so forth the screen of the CAN cable (and possibly the CAN_GND) must also be connected, as well as the signal lines.

Test 7
Use a DC ampere meter to measure the current between the power supply ground and the screen at the end of the network most remote from the power supply unit. An equalization current should be present. If there is no current, then either the screen is not connected all the way through, or the power supply unit is not properly earthed. If the power supply unit is somewhere in the middle of the network, the measurement should be performed at both ends. If necessary, this test can also be done at the ends of the drop lines.

Test 8
Interrupt the screen at a number of locations and measure the current to these connections. If there is a current present, the screen is earthed at more than one place, creating a ground loop.

Potential differences

The screen must be connected all the way through for this test, and must not have any current flow (*see Test 8*).

Test 9
Measure and record the voltage between the screen and the power supply ground at each node. The maximum potential difference between any two devices should be less than 5V.

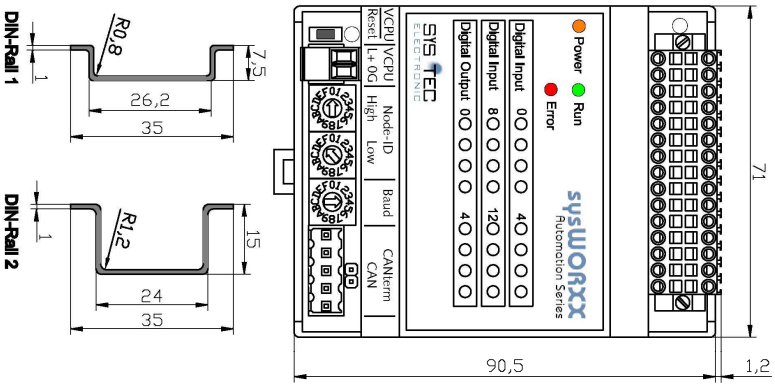
Detect and localize the faults

In a first approach "low-tech method" works best: disconnect parts of the network, and observe if the error disappears.

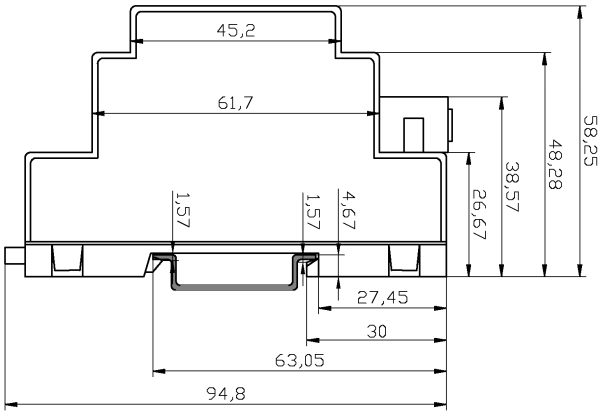
However, this does not work well for problems such as excessive potential differences, ground loops, EMC or signal distortion, since the reduction in the size of the network often solves the problem without the "missing" piece being the cause. The bus load also may change as the network is reduced in size, leading to a more harmonized traffic and therefore making localization of faults more difficult.

Diagnosis with an oscilloscope does not always work out successfully as it is hard do low-level debugging on bit-layer. Especially on heavy traffic and/or disturbances. However, it might be possible to trigger on error frames using a oscilloscope - this type of diagnosis, however, is only recommended for experts.

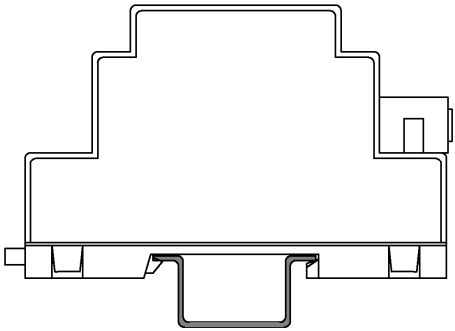
13.3 Module Dimensions



With DIN-Rail 1



With DIN-Rail 2



13.4 Bus cable and termination resistors

The cables, connectors, and termination resistors used in CANopen networks shall meet the requirements defined in ISO 11898. In addition, this section gives some guidelines for selecting cables and connectors.

The table below shows some standard values for DC parameters for CANopen networks with less than 64 nodes:

Bus length [m]	**Bus cable**		**Termination resistance [Ω]**
	Length-related resistance [mΩ/m]	Diameter [mm^2]	
0 … 40	70	0,25 … 0,34	124
40 ... 300	<60	0,34 ... 0,6	150 ... 300
300 ... 600	<40	0,5 ... 0,6	150 ... 300
600 ... 1000	<26	0,75 ... 0,8	150 ... 300

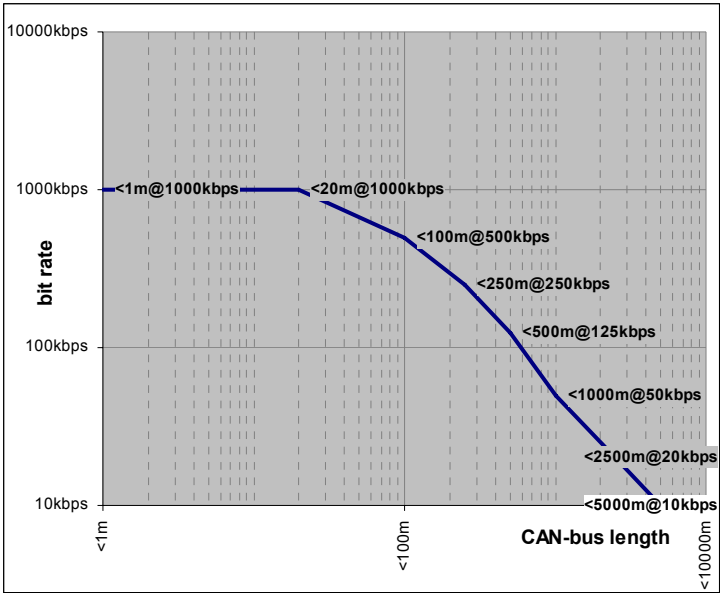


Table 120: CAN-bus length versus bit rate

A figure of 40m at 1 Mbit/s is often found in the CAN literature. This does not, however, apply for networks with optically isolated CAN controllers. The worst case calculation for opto-couplers results in a calculated bus length of 5 m at 1 Mbit/s - in practice, however, 20 m can be reached without difficulty. It may be necessary to use repeaters for bus lengths longer than 1000 m.

For drop cables a wire cross-section of 0.25 to 0.34 mm^2 is recommended.

When calculating the voltage drop, the real connector resistance should be considered too. The contact resistance of one connector can be assumed in a range of 2.5 to 10 mΩ.

With the assumed values for:

Minimum dominant value	$V_{diff.out.min}$ = 1.5 V
Minimum differential input resistance	$R_{diff.min}$ = 20 kΩ
Requested differential input voltage	$V_{th.max}$ = 1.0 V
Minimum termination resistance	$R_{T.min}$ = 118 Ω

The following table shows the maximum length for different bus cables and different number of connected bus nodes.

Wire diameter [mm^2]	**Maximum length [m] safety margin 0,2**			**Maximum length [m] safety margin 0,1**		
	n=32	n=64	n= 100	n=32	n=64	n= 100
0,25	200	170	150	230	200	170
0,5	360	310	270	420	360	320
0,75	550	470	410	640	550	480

Note:

If driving more than 64 nodes and/or more than 250m bus length the accuracy of the V_{cc} supply voltage for the ISO 11898 transceiver is recommended to be 5% or lower. You also have to consider the minimum supply voltage of at least 4.75V when driving 50Ω load, i.e. 64 bus nodes, and at least 4.9V when driving 45Ω load, i.e. 100 bus nodes.

For more information please refer to standard CiA 303-1 and CiA 102.

This page was left empty intentionally.

Glossary

CAN:	**Controller Area Network** is an internationally standardized serial bus system.
COB:	**Communication Object** A unit of transportation in a CAN network. Data must be sent across a CAN Network inside a COB. There are 2048 different COBs in a CAN network. A COB can contain at most 8 bytes of data.
COB-ID:	Each COB is uniquely identified in a CAN network by a number called the **COB Identifier** (COB-ID). The COB-ID determines the priority of that COB for the MAC sub-layer.
Remote COB:	A COB whose transmission can be requested by another device.
CRC:	**Cyclic Redundancy Check**.
CSDO:	**Client SDO**
FV:	Field Value : the converted analog input value (raw value). This value is always left adjusted.
LED:	Light Emitting Diode
MAC:	**Medium Access Control** One of the sub-layers of the Data Link Layer in the CAN Reference Model that controls who gets access to the medium to send a message.
NMT:	**Network Management** One of the CANopen service elements of the application layer in the CAN OSI Reference Model. The NMT serves to configure, initialize, and handle errors in a CANopen network.
Node-ID:	The node-ID is the address of nodes in a CANopen network and therefore has to be assigned uniquely. It also determines the offset of the communication objects (COBs) and the priority of the node. The node-ID 0 is reserved. Possible values: 1 to 127
OSI:	**Open Systems Interconnection**.
PE:	upper range value
PDO:	**Process Data Object**.
PV:	Process Value: The field value is converted to the real physical dimension of the measured quality, and the result is called "process value".
RPDO:	**Receive PDO**.

Pre-defined Connection Set	The pre-defined connection set is a default assignment of CAN message identifiers (COB-IDs) to CANopen objects. This default assignment guarantees that the CAN message identifiers are uniquely assigned in the network, if the node-ID has been assigned uniquely.
SDO:	**Service Data Object**.
SI:	International system of units
SSDO:	**SDO Server**.
SYNC:	**Synchronization Object**.
TPDO:	**Transmit PDO**.
U_{OH}:	Output voltage high
U_{OL}:	Output voltage low
U_{IH}:	Input voltage high
U_{IL}:	Input voltage low
I_{OH}:	Output current high
I_{OL}:	Output current low
I_{IH}:	Input current high
t_{OFF}:	Output turn off time
t_{ON}:	Output turn on time

Index

Revision history

Date	Version	Editor	Comments
2005-Dez-14	L-1070e_01	C.Thomas	Beschreibung der Module und von CANopen Funktionalität eingepflegt
2006-Mar-19	L-1070e_01	A. von Collrepp	Complete revision. Missing sections added. Reformatting.
2006-Apr-03	L-1070e_01	A. von Collrepp	Finalizing. Pre-release.
2006-Jun-17	L-1070e_01	A. von Collrepp	Release version
2007-Nov-22	L-1070e_02	F. Jung-andreas	Completion pulsed DO for X1, Completion Calibartion Entries for X4, X5, X6
2008-Mar-25	L-1070e_03	F. Jung-andreas	Completion Calibartion Entries for X7
2008-Aug-07	L-1070e_04	F. Jung-andreas	Correction Status (0x6150) for X5 and physical unit (0x6131) for X5 and X7
2009-May-18	L-1070e_05	M. Berthel	Correction sampling rate for X7
2010-Mar-24	L-1070e_06	M. Berthel	new OD entries 1F51H and 2002H States of RUN- and ERROR-LED at Program Control Emergency Error Codes at Program Control all changes from firmware version 1.30
2010-Apr-26	L-1070e_07	M. Berthel	Chanche of LED display on modules X5 and X7 all changes from firmware version 1.31

This page was left empty intentionally.

Suggestion for improvements

Document:	**sysWORXX CANopen I/O modules**
Document number:	**L-1070e_07**

How would you improve this manual?

Did you find any mistakes in this manual? page

Submitted by:

Customer number:

Name:

Company:

Address:

Please return your suggests to:

SYS TEC electronic GmbH
August-Bebel-Str. 29
D-07973 Greiz
GERMANY
Fax : +49-3661-6279-99
Email: info@systec-electronic.com

Published by:

Ordering No. L-1070e_07