CANopen

Network configuration

Integration of Bürkert devices in CANopen networks

Operating instructions
Software

bürkert
FLUID CONTROL SYSTEMS

Content

CANopen quick guide

This chapter describes in brief the steps on how to connect a CANopen device and to start it up.

Detailed information can be found in the following full description.

I. Setting the "CANopen" bus mode on the device

To operate a Bürkert device in a CANopen network, the bus mode must be changed from the standard setting "büS" to "CANopen".

There are different options for the setting depending on the device:

A. If devices have a physical switch for switching over the bus mode, move the switch to CANopen. The description can be found in the respective operating instructions.

B. If devices have a display, set the bus mode on the display.
To make the setting with the left navigation button, switch to the configuration area and select "General settings".
Setting in the menu: ➡ "Parameter" ➡ "büS" ➡ "Advanced" ➡ "Bus mode ➡ "CANopen.

C. If devices do not have a physical switch or display, set the bus mode with the PC software Bürkert-Communicator.
To make the setting, select "General settings".
Setting in the menu: ➡ "Parameter" ➡ "büS" ➡ "Advanced" ➡ "Bus mode ➡ "CANopen.

II. Setting baud rate and node ID

After selecting the bus mode "CANopen", the baud rate must be selected for the device and the node ID entered.

Important information:
- Each device in the network has its own unique node ID.
- The baud rate is identical for all devices in the network.

There are different options for the setting depending on the device:

A. If devices have DIP switches, the setting of the baud rate and node ID is described in the respective operating instructions.

B. If devices have a display, set the baud rate and node ID on the display.
To make the setting with the left navigation button, switch to the configuration area and select "General settings".

Setting in the menu: ➡ "Parameter" ➡ "büS" ➡ "Advanced"
➡ "Baud rate" ➡ Select the baud rate.
➡ "büS address" ➡ Enter the node ID

Automatic assignment of the node ID: If the device still does not have a node ID and is started in büS mode, the node ID is assigned automatically. The node ID required for the next steps can be seen in the "büS address" menu.

C. If devices do not have DIP switches or a display, set the baud rate and node ID with the PC software Bürkert-Communicator.
To make the setting in Bürkert-Communicator, select "General settings".
Setting in the menu: ➡ "Parameter" ➡ "büS" ➡ "Advanced"
➡ "Baud rate" ➡ Select the baud rate.
➡ "büS address" ➡ Enter the node ID.

Automatic assignment of the node ID: If the device still does not have a node ID and is started in büS mode, the node ID is assigned automatically. The node ID required for the next steps can be seen in the "büS address" menu.

Note: The setting can also be made with the Layer Setting Service. The description can be found in the following chapter "2.3 Layer Setting Services (LSS)".

III. Setting Heartbeat/Nodeguarding

Bürkert devices transmit Heartbeat messages as standard.
The standard Heartbeat time is 500 ms. If the Heartbeat time is to be changed, it must be entered in ms as SDO in object 0x1017.

If devices support Nodeguarding, this can be used instead of Heartbeat. In the case of Nodeguarding configure the objects 0x100C and 0x100D.

IV. Switching the device to operational

To switch the device to operational mode, transmit the NMT command "Operational". This looks as follows:

Identifier	DLC	Byte 1	Byte 2
0x00	02	01	Node ID

Meaning of node ID: unique device address in the CANopen network

1. Introduction

Bürkert CANopen devices comply with the CANopen standard according to the standards:

- CiA Draft Standard 301; Application Layer and Communication Profile
- CiA Draft Standard Proposal 305; Layer Setting Services (LSS) and

2. Start-up

2.1 Node ID

In a CANopen network each device must have a unique node ID. The node ID is saved in the Device Communication Object (0x2001) in sub-index 0x02 and can also be changed here.

CANopen also offers the option of changing the node ID via the so-called Layer Setting Services (LSS) (see Layer Setting Services (LSS))

2.2 Baud rate

The baud rate describes the transmission rate in a CANopen network.

Bürkert devices support the following baud rates:

Baud rate	Value
1 mbit/s	0
500 kbit/s	2
250 kbit/s	3
125 kbit/s	4
reserved	5
50 kbit/s	6
20 kbit/s	7
10 kbit/s	8

The baud rate for Bürkert devices is 500 kbit/s as standard. It is saved, the same as the node ID, in the Device Communication Object (0x2001) however in sub-index 0x01.

As with the node ID, CANopen also offers the option of changing the baud rate via LSS (see Layer Setting Services (LSS)).

2.3 Layer Setting Services (LSS)

As already mentioned, two things are required to connect CANopen devices to a network:

- All devices must use the same baud rate
- The node ID of a device must be unique.

If the manual setting cannot be made on the device, it can be made via the Layer Setting Services (LSS).

First a connection must be established with the device, then the baud rate and the node ID can be set via a "Dialog".

The device must be activated with the special COB-ID (0x7E5), the device responds via the COB-ID (0x7E4). LSS messages are always 8 bytes long, all unused bytes are reserved and should be initialized with 0.

The responses of the device include either a success message or an error message. The error message consists of an error code, which represents the error, and an error extension which supplies specific information about the error.

Example of the configuration via LSS:

1. Switch devices into configuration mode using the "Switch Mode Global" service; this will switch all devices into the configuration mode:

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x7E5	8	0x04	0x01	reserved					

2. Transfer the new baud rate to the device using the "Configure Bit Timing" service:

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x7E5	8	0x13	Table	Index	reserved				

Meaning of table: Indicates which baud rate table is to be used.

0 is the baud rate table which is defined according to CiA DSP-305:

Value	Baud rate
0	1 mbit/s
2	500 kbit/s
3	250 kbit/s
4	125 kbit/s
5	100 kbit/s
6	50 kbit/s
7	20 kbit/s
8	10 kbit/s

Meaning of value: Indicates the index within the baud rate table

The response of the device to Configure Bit Timing:

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x7E4	8	0x13	Error Code	Error Extension	reserved				

Meaning of error code: Error code:
0   = successfully run
1   = baud rate is not supported
2 – 254   = reserved
255   = special error code, see error extension

Meaning of error extension: manufacturer-specific error code (if error code = 255)

3. If the device responds without an error, the baud rate must now be activated using the "Activate Bit Timing" service:

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x7E5	8	0x15	Delay		reserved				

Meaning of delay: Relative time until the new baud rate switches on in ms

4. Transfer the new node ID to the device using the "Configure node ID" service:

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x7E5	8	0x11	Node ID	reserved					

Meaning of node ID: New node ID for the device (only values between 1 and 127)

The response of the device to "Configure node ID":

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x7E4	8	0x11	Error Code	Error Extension	reserved				

Error code: Error code: 0   = successfully run
                            1   = node ID invalid
                            2 – 254   = reserved
                            255   = special error code, see error extension

Error extension: manufacturer-specific error code (if error code = 255)

5. Switch the devices into operation mode using the "Switch Mode Global" service:

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x7E5	8	0x04	0x00	reserved					

Note:

The baud rate and the node ID can be changed independently of each other or re-assigned.

- Changing baud rate: To do this, steps 1, 2, 3 and 5 must be run.
- Assign new node ID: To do this, steps 1, 4 and 5 must be run.

3. Transmission services

3.1 Service Data Object (SDO)

SDOs (Service Data Objects) are used to parameterize object directory entries. They are read and written with a CANopen telegram. As soon as a CANopen slave is preoperational, they can be processed.

Two CAN messages are required to transfer data. Firstly there is the "SDO request" for an SDO request and secondly the "SDO response" for an SDO response.

The two network nodes, which are involved in this process, are designated as "SDO Client" and as "SDO Server". The server provides the data via its object directory or accepts it. The client requests the data (reads) or transfers it (writes). The initiative for an SDO transfer always comes from the client.

As an SDO transfer is confirmed, each SDO request must be answered, even if the device can provide no relevant data or the request was faulty. This negative response is called "Abort" and includes a long "Abort code" in addition to the 4 bytes. The reason for the abort is explained in more detail in the "Abort Code" and the address of the object directory entry, which refers to the faulty SDO transfer, is indicated.

3.2 Process Data Object (PDO)

PDOs (Process Data Objects) provide a quick option for transporting process and real-time data. PDOs can only be sent when the CANopen slave is operational.

PDOs include the work values of a CANopen slave for the cyclical process operation. Each CANopen Slave can manage several PDOs. However, it is important that each available PDO has its own COB ID.

The data field of a PDO telegram can be between 1 and 8 bytes. Any information of the CANopen slave can be represented within the data field.

The contents of a PDO cannot be directly interpreted. In principle, the transmitter and the receiver can interpret the contents of the PDO. It is therefore adequate to identify a PDO via its COB ID only.

The PDO mapping describes which individual process variables in the data field of a PDOs are sent, how they are arranged and which data type and length they have. For this purpose, each defined PDO describes the contents and the meaning of the data field by means of the PDO mapping within the object directory. This occurs both on the transmitter and on the receiver side.

There are different transmission types for PDOs.

- Asynchronous PDOs: are event-controlled and represent the standard transmission type for PDOs. This means, if at least one of the mapped process variables of the PDO changes, the PDO is transmitted immediately.

- Synchronous PDOs: are only transmitted if they receive a synchronization message beforehand. The PDO is therefore transmitted synchronously into the entire network. It is important that the input of all devices must be scanned according to the synchronization object. This gives a standard snapshot of the process results. When the next synchronization message occurs, the recorded data is transmitted with the synchronous PDOs. This results in a delay according to the cycle time of the synchronization message, as the consumers receive the process variables at the time of the previous synchronization message.

Also, so-called multiplexed PDOs are used for Bürkert devices.

- Multiplexed PDOs: offer the possibility of sending more than 8 bytes of data with an identifier.
  This is then used in Bürkert devices if 4 PDOs are inadequate to send all process values. In this case the remaining process values are mapped onto PDO 4.
  This functions by means of scanner MPDOs. A maximum of one of these scanner PDOs is allowed for each device. The manufacturer uses a so-called Object Scanner List to know which objects he must send.
  Scanner PDO (Source Address Mode (SAM)):

COB-ID	Node ID of producer	Index & sub-index	Data (max. 32 bits)

The sub-index is now changed for each object from the Object Scanner List and the new data is sent with the same MPDO. The advantage of this is that only one identifier is required for the MPDO in comparison with the use of normal PDOs.

3.3 Synchronization Object (SYNC)

The SYNC telegram is a periodical broadcast telegram and is a trigger for CANopen functions. The SYNC telegram can be used to transfer synchronized input data and simultaneously activate output data on a system-wide basis.

3.4 Emergency Object (EMCY)

If a CANopen Slave malfunctions, it sends an emergency message to the fieldbus. Only one emergency message is sent for each error. Neither is it specified how many consumers must receive the emergency message. If the nodes receive an emergency message, they may perform e.g. an emergency stop.

3.5 Nodeguarding

With Nodeguarding, the Master monitors the CANopen Slaves by sending cyclical telegrams to each Slave. Each CANopen Slave must reply to the Nodeguarding telegram with a Status telegram.

Nodeguarding enables the Master to detect the failure of a CANopen Slave.

3.6 Heartbeat

Heartbeat monitoring corresponds to Nodeguarding, in that the CANopen Master does not generate any request telegrams. The Heartbeat telegram is sent independently by the Slave ("Producer Heartbeat") and can be evaluated in the Master ("Consumer Heartbeat").

3.7 Network Management Services (NMT)

Each CANopen device may have different system statuses. These are activated by means of NMT commands.

When the device has been switched on ("Power On"), an internal system initialization ("initialization") is run. Following a successful initialization, a boot-up telegram responds. The device is now operational and is in the "preoperational" status. As a slave can be parameterized in this status, it is therefore possible to read and write SDOs, however the PDOs are not replaced.

The NMT command "Operational" can be used to switch a CANopen device to the operational status. In this status the process data is active, the RPDO and TPDO communication is running.

The NMT command "Reset Application" is used to restart the CANopen device, whereas the NMT command "Reset Communication" is used to reset the CANopen communication of the device only. Following both resets, the device is in the initialization status.
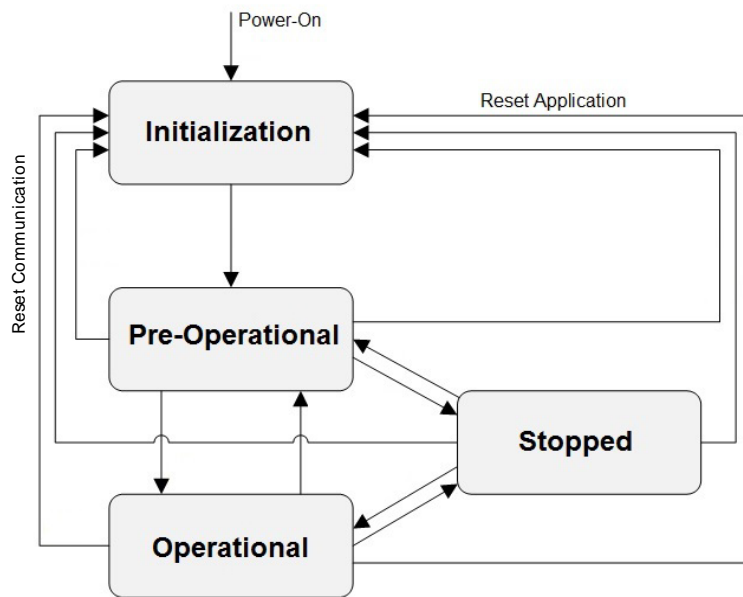
Figure 1: NMT (http://doc.ingeniamc.com/emcl2/command-reference-manual/communications/canopen-protocol/canopen-objects/nmt/nmt-state-machine), amended

3.8 Predefined Connection Set

To streamline the configuration effort with simple network structures, CANopen provides predefined identifiers, the so-called Predefined Connection Set:

Transmission service	COB ID(s)	Device
NMT	0x00	Receive only
SYNC	0x80	Receive only
EMCY	0x80 + node ID	Send
PDO	**0x180 + node ID** **0x200 + node ID** **0x280 + node ID** **0x300 + node ID** **0x380 + node ID** **0x400 + node ID** **0x480 + node ID** **0x500 + node ID**	**1. TPDO** **1. RPDO** **2. TPDO** **2. RPDO** **3. TPDO** **3. RPDO** **4. TPDO** **4. RPDO**
SDO	**0x580 + node ID** **0x600 + node ID**	**Send** **Receive**
Nodeguarding/Heartbeat	**0x700 + node ID**	**Send**
LSS	0x7E4 0x7E5	Send Receive

4. Object overview

The following objects are supported as standard by a Bürkert CANopen device:

Index (hex)	Sub-indices (hex)	Name	Access			PDO-mappable
			read	write	constant	
1000	0	Device Type	x			
1001	0	Error Register	x			
1005	0	COB ID SYNC	x	x		
1006	0	Communication Cycle Period	x	x		
1008	0	Manufacturer Device Name			x	
1009	0	Manufacturer Hardware Version			x	
100A	0	Manufacturer Software Version			x	
100C	0	Guard Time	x	x		
100D	0	Life Time Factor	x	x		
1014	0	COB ID EMCY	x			
1016	0 – 7F	Consumer Heartbeat Time	x	x		
1017	0	Producer Heartbeat Time	x	x		
1018	0 - 4	Identity Object	x			
1200 – 13FF	0 - 3	Server SDO Parameter	x	(x)		
1400 – 15FF	0 - 2	Receive PDO Communication Parameter	x	(x)		
1600 – 17FF	0 - 40	Receive PDO Mapping Parameter	x	x		
1800 – 19FF	0 – 5	Transmit PDO Communication Parameter	x	(x)		
1A00 – 1BFF	0 - 40	Transmit PDO Mapping Parameter	x	x		
2000	0 - 9	Bürkert Device Description Object	x			
2001	0 – A	Device Communication Object	x	(x)		
2002	0 – 4	User Configuration Object	x	x		
2004	0 - F	Device Status Object	x	(x)		(x)
2500 – 253F	0 – 9	Sensor Value	x	(x)		(x)
2540 – 257F	0 – 9	Control Value	x	(x)		(x)

x - the characteristic applies

(x) - the characteristic may apply depending on sub-index

4.1 Detailed description

Object 0x1000   **Device Type**

Describes device type and applied profile
Data type: Unsigned32

Object 0x1001   **Error Register**

Register for device errors; part of the Emergency object
Data type: Usigned8

Object 0x1005   **COB ID SYNC**

Specifies the COB ID for the SYNC object and the generation of SYNC telegrams
Data type: Unsigned32
Default: 0x00000080

Object 0x1006   **Communication Cycle Period**

The interval between successive SYNC signals in µsec
Data type: Unsigned32

Object 0x1008   **Manufacturer Device Name**

Device designation of the manufacturer
Data type: Visible_String

Object 0x1009   **Manufacturer Hardware Version**

Version description of the hardware
Data type: Visible_String

Object 0x100A   **Manufacturer Software Version**

Version description of the software
Data type: Visible_String

Object 0x100C   **Guard Time**

Setting of the Nodeguarding time in msec
Data type: Unsigned16

Object 0x100D   **Life Time Factor**

Setting of a factor according to which the slave must have responded to the Nodeguarding tele-
gram
Data type: Unsigned8

Object 0x1014   **COB ID EMCY**

Specifies the COB ID for the Emergency object
Data type: Unsigned32
Default: 0x80 + node ID

Object 0x1016   **Consumer Heartbeat Time**

Sub-Index	Name	Data type	Default	Access	
				Read	Write
0x00	Number Of Entries	Unsigned8	127	x	
0x01	Consumer Heartbeat Time	Unsigned32	0	x	x
0x02	Consumer Heartbeat Time 2	Unsigned32	0	x	x
…	…	…	…	…	…
0x7F	Consumer Heartbeat Time 7F	Unsigned32	0	x	x

This is where the Heartbeat times are entered which the Master consumes. Up to 127 devices are possible.

Object 0x1017   **Producer Heartbeat Time**

Setting of the Heartbeat time in msec
Data type: Unsigned16
Default: 500

Object 0x1018   **Identity Object**

Sub-Index	Name	Data type	Default	Access	
				Read	Write
0x00	Number Of Entries	Unsigned8	4	x	
0x01	Vendor ID	Unsigned32	0x39	x	
0x02	Product Code	Unsigned32		x	
0x03	Revision Number	Unsigned32		x	
0x04	Serial Number	Unsigned32		x	

Object 0x1200   **Server SDO Parameter**

Sub-Index	Name	Data type	Default	Access	
				Read	Write
0x00	Number Of Entries	Unsigned8	3	x	
0x01	COB ID Client To Server	Unsigned32	0x600 + node ID	x	(x)
0x02	COB ID Server To Client	Unsigned32	0x580 + node ID	x	(x)
0x03	Node ID of the SDO Client	Unsigned8		x	x

(x) – applies conditionally, depending on whether it is the first server SDO parameter or not

There is one object in the object directory for each available server SDO parameter. These objects receive a continuous index, from 0x1200 to maximum 0x13FF.

Object 0x1400 **Receive PDO Communication Parameter**

Sub-Index	Name	Data type	Default	Access	
				Read	Write
0x00	Number Of Entries	Unsigned8	2	x	
0x01	COB ID	Unsigned32	0x200 + node ID	x	x
0x02	Transmission Type	Unsigned8	0xFE	x	x

There is one object in the object directory for each available Receive PDO Communication parameter. These objects receive a continuous index, from 0x1400 to maximum 0x15FF.
The COB ID is increased by 0x100 for each entry.

Object 0x1600 **Receive PDO Mapping Parameter**

Sub-Index	Name	Data type	Default	Access	
				Read	Write
0x00	Number Of Entries	Unsigned8		x	x
0x01	PDO Mapping Entry	Unsigned32		x	x
0x02	PDO Mapping Entry	Unsigned32		x	x
…	…	…	…	…	…

There is one object in the object directory for each available Receive PDO Mapping parameter.
These objects receive a continuous index, from 0x1600 to maximum 0x17FF.
These Mapping parameters may have up to 40 mapping entries, each of which is in a sub-index.

Object 0x1800 **Transmit PDO Communication Parameter**

Sub-Index	Name	Data type	Default	Access	
				Read	Write
0x00	Number Of Entries	Unsigned8	5	x	
0x01	COB ID	Unsigned32	0x180 + node ID	x	x
0x02	Transmission Type	Unsigned8	0xFE	x	x
0x03	Inhibit Time	Unsigned16	5000	x	x
0x04	Compatibility Entry	Unsigned8		x	
0x05	Event Timer	Unsigned16	1000	x	x

There is one object in the object directory for each available Transmit PDO Communication parameter. These objects receive a continuous index, from 0x1800 to maximum 0x19FF.
The COB ID is increased by 0x100 for each entry.

Object 0x1A00   **Transmit PDO Mapping Parameter**

Sub-Index	Name	Data type	Default	Access	
				Read	Write
0x00	Number Of Entries	Unsigned8		x	x
0x01	PDO Mapping Entry	Unsigned32		x	x
0x02	PDO Mapping Entry	Unsigned32		x	x
…	…	…	…	…	…

There is one object in the object directory for each available Transmit PDO Mapping parameter. These objects receive a continuous index, from 0x1A00 to maximum 0x1BFF.

These Mapping parameters may have up to 40 mapping entries, each of which is in a sub-index.

Object 0x2000   **Bürkert Device Description Object**

Sub-Index	Name	Data type	Default	Access	
				Read	Write
0x00	Number Of Entries	Unsigned8	9	x	
0x01	Device Name	Visible_String		x	
0x02	Ident Number	Unsigned32		x	
0x03	Manufacture Date	Visible_String		x	
0x04	Software Ident Number	Unsigned32		x	
0x05	Software version	Unsigned32		x	
0x06	Hardware version	Unsigned32		x	
0x07	Serial Number	Unsigned32		x	
0x08	Product Code	Unsigned32		x	
0x09	Product Group	Unsigned8		x	

Description of the sub-indices:

Device Name:	Unique device name
Ident Number:	Unique ident number of the device
Manufacture Date:	Date of manufacture of the device
Software Ident Number:	Unique ident number of the software
Software version:	Version number of the software
Hardware version:	Version number of the hardware
Serial Number:	Unique serial number
Product Code:	Unique product code
Product Group:	Product group of the device

Object 0x2001   **Device Communication Object**

Sub-Index	Name	Data type	Default	Access Read	Write
0x00	Number Of Entries	Unsigned8	10	x	
0x01	Baud rate	Unsigned8	2	x	x
0x02	Address	Unsigned8		x	x
0x03	büS Mode	Unsigned8	1	x	x
0x04	Reset	Unsigned8	0	x	x
0x05	büS Version	Unsigned32		x	
0x06	Rx error count	Unsigned8		x	
0x07	Rx error count max	Unsigned8		x	x
0x08	Tx error count	Unsigned8		x	
0x09	Tx error count max	Unsigned8		x	x
0x0A	CAN operation status	Unsigned8		x	

Description of the sub-indices:

Baud rate:	Baud rate of the device
Address:	Node ID of the device
büS Mode:	Selection between CANopen and büS mode
Reset:	Possibility of communication or device reset
büS Version:	Version number of the büS
Rx error count:	Counter for Rx error
Rx error count max:	maximum Rx number of errors
Tx error count:	Counter for Tx error
Tx error count max:	maximum Tx number of errors
CAN operation status:	Display of the CAN Operation Status

Object 0x2002   **User Configuration Object**

Sub-Index	Name	Data type	Default	Access Read	Write
0x00	Number Of Entries	Unsigned8	4	x	
0x01	Unique Device Name	Visible_String		x	x
0x02	Location Information	Visible_String		x	x
0x03	User Description	Visible_String		x	x
0x04	Displayed Device Name	Visible_String		x	x

Description of the sub-indices:

Unique Device Name:	Unique device name
Location Information:	Description of the physical device location
User Description:	Description of the device
Displayed Device Name:	Displayed device name

Object 0x2004   **Device Status Object**

Sub-Index	Name	Data type	Default	Access		PDO mappable
				Read	Write	
0x00	Number Of Entries	Unsigned8	8	x		
0x01	Device Status NamurNe107	Unsigned8	0	x		x
0x02	Device Temperature	Unsigned16		x	x	
0x03	Device Supply Voltage	Real32		x	x	
0x04	Operation Time_[s]	Unsigned32		x	x	
0x05	Maximum Device Temperature	Unsigned16		x	x	
0x06	Minimum Device Temperature	Unsigned16		x	x	
0x07	Maximum Device Voltage	Real32		x	x	
0x08	Minimum Device Voltage	Real32		x	x	

Description of the sub-indices:

Device Status NamurNe107:	Namur status of the device
Device Temperature:	Device temperature
Device Supply Voltage:	Supply voltage of the device
Operation Time_[s]:	Time which the device is operating
Maximum Device Temperature:	Maximum temperature of the device
Minimum Device Temperature:	Minimum temperature of the device
Maximum Device Voltage:	Maximum supply voltage of the device
Minimum Device Voltage:	Minimum supply voltage of the device

Device Status NamurNe107:   Bits 0 - 3: Namur State: 0: normal
1: diagnose active
2: maintenance required
3: out of specification
4: warning
5: error

Bits 4 - 5: Namur Mode: 0: auto
1: hand
2: flashing

Bits 6 – 7: Reserved

Object 0x2500   **Sensor Value**

| Sub-Index | Name | Data type | Default | Access | | PDO-mappable |
				Read	Write	
0x00	Number Of Entries	Unsigned8	7	x		
0x01	Value			x	x	x
0x02	Unit	Unsigned32		x		
0x03	Name	Visible_String		x		
0x04	Classification	Unsigned16		x		
0x05	Datatype	Unsigned16		x		
0x06	Precision	Real32		x		
0x07	Feature Group	Unsigned32	0	x		

Description of the sub-indices:

Value:	Value which can be mapped onto a TPDO
Unit:	Unit of the value
Name:	Name of the sensor value
Classification:	Classification of the measured value
Datatype:	Data type of the value
Precision:	Precision of the value
Feature Group:	Grouping for automatic partner search

The sensor value is the process value of a Bürkert device. There is one entry in the object directory for each available sensor value. These objects receive a continuous index, from 0x2500 to maximum 0x253F.
The value objects of a sensor value can be mapped onto the TPDOs and therefore be consumed by other devices in the network.

Object 0x2540   **Control Value**

| Sub-Index | Name | Data type | Default | Access | | PDO-mappable |
				Read	Write	
0x00	Number Of Entries	Unsigned8	7	x		
0x01	Value			x	x	x
0x02	Unit	Unsigned32		x		
0x03	Name	Visible_String		x		
0x04	Classification	Unsigned16		x		
0x05	Datatype	Unsigned16		x		
0x06	Precision	Real32		x		
0x07	Feature Group	Unsigned32	0	x		

Description of the sub indices

Value:	Value which can be mapped onto an RPDO
Unit:	Unit of the value
Name:	Name of the sensor value
Classification:	Classification of the measured value
Datatype:	Data type of the value
Precision:	Precision of the value
Feature Group:	Grouping for automatic partner search

The control value receives a value, e.g. from another device via a PDO or by a user entry. There is a continuous index, from 0x2540 to maximum 0x25BF, for each available control value.
The value objects can be mapped onto the RPDOs and therefore be consumed by other devices in the network.