MicroControl

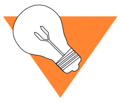
Systemhaus für Automatisierung

µCAN.8.dio-SNAP

Manual Digital I/O-Module
Version 3.04

Explanation of Symbols

To facilitate reading and understanding of the document several symbols will be seen on the left side of the manual.



This symbol marks a paragraph which contains useful information for working with the device or which gives useful hints.

This symbol marks a paragraph which explains possible sources of danger which might cause damage to the system or operating personnel.

Keywords

Important keywords appear in the border column to facilitate navigation through the text.

MicroControl GmbH & Co. KG
Junkersring 23
D-53844 Troisdorf
Fon: +49 / 2241 / 25 65 9 - 0
Fax: +49 / 2241 / 25 65 9 - 11
https://www.microcontrol.net

1. Safety Regulations

Please read the following chapter in any case to ensure safe handling of electrical devices.

1.1 General Safety Regulations

This paragraph contains important information about the usage of µCAN modules. It was written for personnel which is qualified and trained on the use of electrical devices.

Qualified and trained personnel are persons who fulfill at least one of the following conditions:

- You know the safety regulations for automated machines and are familiar with the handling of the machine.
- You are the operator of the machines and you have previously been trained on operation modes. You are familiar with the operation of the devices described in this manual.
- You are responsible for setting devices into operation or servicing and are trained on repairing automated machines. In addition, you are trained in setting electrical devices into operation, to connect the grounding conductor and to label these devices accordingly.

The devices described in this manual shall only be used for the mentioned applications. Other devices used in conjunction have to meet the respective safety regulations and EMI requirements.

To ensure a trouble free and safe operation of the device, please ensure proper transport, appropriate storage, proper assembly as well as careful operation and maintenance.

Please see to it that the local safety regulations are observed during set-up of the devices.

If devices are to be integrated into stationary machines without a mains switch for all phases or fuses, this equipment must be installed first. The stationary machine must be connected to a grounding conductor.

If devices are supplied by mains, please see to it that the selected input voltage fits to the local mains.

1.2 Safety Notice

If the devices are supplied by 24V DC auxiliary supply, please ensure isolation of the low-voltage lines from other voltages.

The cables for power supply, signal lines and sensor lines must be installed in such a way that the functionality of the device is not influenced by EMI.

Devices or machines used in industrial automation must be constructed in such a manner to prevent any unintentional operation.

Safety precautions have to be taken by means of hardware and software in order to avoid undefined operational states of automated machines in case of a cable break.

Where automated machines can cause damage to material or personnel in case of a malfunction, the system designer has to ensure that the safety -precautions are met. Possible safety precautions might be the integration of a limit switch or a mechanical locking device.

2. Operation of µCAN.8.dio-SNAP

2.1 Overview

The µCAN.8.dio-SNAP is the ideally suited for input and output of digital signals via CAN bus.

Fig. 1: Digital I/O module µCAN.8.dio-SNAP

Separated from the main system, the µCAN modules are supposed to acquire data at the source which will reduce wiring and costs.

The development in automation towards decentralized "intelligent" systems makes communication between these components increasingly important.

Modern automated systems require the possibility to integrate components from different manufacturers. The solution to this problem is a common bus system.

All these requirements are fulfilled by the µCAN.8.dio-SNAP module. The µCAN.8.dio-SNAP runs on the standard field bus CAN.

Typical applications of the µCAN.8.dio-SNAP are industrial automation, automotive technology, food industry and environmental technology.

The µCAN.8.dio-SNAP operates with the protocols

CANopen® CANopen® FD

according to CiA-301, CiA-1301 and CiA-401. Other protocols are available on request.

Space saving and compact

The compact and space-saving casing offers the opportunity to apply the module virtually everywhere.

Inexpensive and service friendly

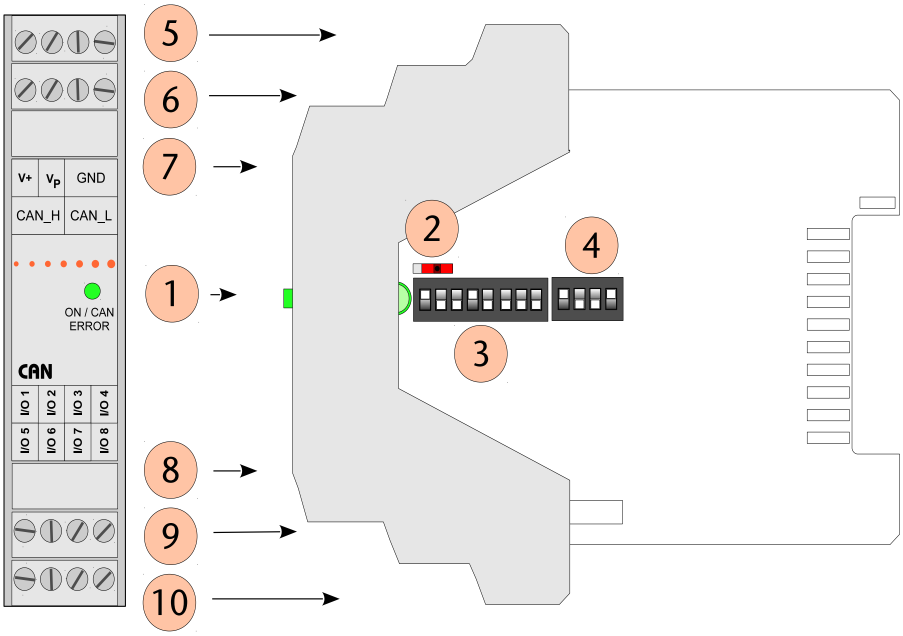
The quick and easy integration of the µCAN.8.dio-SNAP in your application reduces the development effort. Costs for material and personnel- are reduced to a minimum. The easy installation of the module facilitates maintenance and replacement.

3. Project Planning

The chapter Project Planning contains information which is important for systems engineers and users of the µCAN.8.dio-SNAP. This information include case dimensions and optimum conditions of use.

3.1 Functional units of the µCAN module

The following figure shows the different functional groups of the µCAN module. The figure illustrates the structure and position of the setting and application options available.



1: bi-color state LED
2: Termination switch
3: module ID setting
4: set bit rate
5: terminals for power supply
6: terminals for CAN
7: terminal cover
8: terminal cover
9: terminals for digital signals (1..4)
10: terminals for digital signals (1..4)

Fig. 2: Overview of Functional Units

3.2 General Description

The µCAN.8.dio-SNAP is a µCAN module suited for input and output of digital signals via CAN bus. Each µCAN module can manage a maximum of 8 digital I/O signals. The signal terminals are configured (input/output) via software, no jumpers have to be adjusted. The µCAN module can be supplied with a voltage of 9V - 36V.

Connection of the µCAN.8.dio-SNAP to power supply and CAN bus should be realised via four-core wires, thus reducing wiring to a minimum. Adequate CAN cables are also available.

3.3 Maximum System Layout

For an executable system at least one network manager must be connected to the CAN bus. This network manager may be a PLC or PC equipped with an adequate CAN card. Every µCAN module is and active node.

A bus line can logically be controlled from a maximum of 127 modules. Each modules has to be assigned an individual address which is set at the module itself via DIP switch. The CAN bus can be connected through the individual µCAN modules.

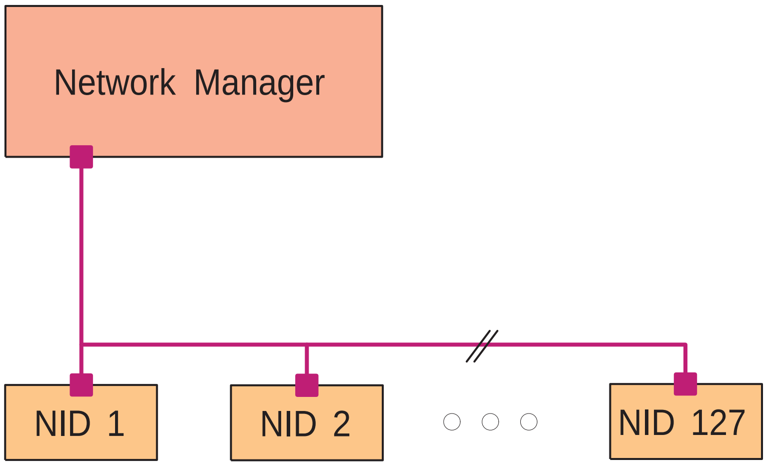


Fig. 3: Maximum System Layout

The maximum cable lengths depend on the selected bit rate and are listed in the table below. The values are recommended by CAN in Automation and can be realized with the µCAN.8.dio-SNAP.

Bit rate	**Cable length**
1000 kBit/s	25 m
800 kBit/s	50 m
500 kBit/s	100 m
250 kBit/s	250 m
125 kBit/s	500 m
100 kBit/s	650 m
50 kBit/s	1000 m

Table 1: Bit Rates in Relation to Cable Length

CAN in Automation recommends not to use the bit rate of 100 kBit/s in new systems.

3.4 Case Dimensions

The case dimensions of the µCAN.8.dio-SNAP are given in the drawing below. Due to protection class IP20 casing, the µCAN module can be installed virtually everywhere. The µCAN modules can either be installed directly on the machine or inside a switching cabinet. Please check the technical data section for detailed information about the maximum environment conditions of the µCAN module.

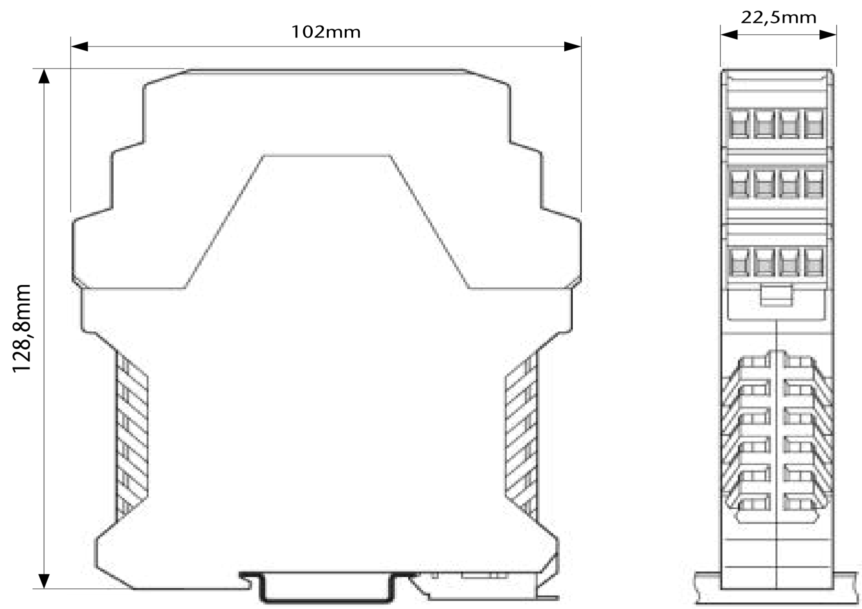


Fig. 4: Case Dimensions

4. Assembly and Disassembly

4.1 General Information

Installation

The modules are designed to be mounted to a standard DIN rail TS35. The µCAN modules are equipped with a snap lock and can be snapped to the rail without using any tools.

Power Supply

Power can be supplied via a two-core wire which is fixed to the respective terminals. However, application of four-core wires is more convenient as the CAN bus can use the same connection.

The PE is supplied by an integrated functional earthing conductor which is located in the back of the casing. Snapping the module onto the rail simultaneously connects the contact to the rail. PE supply within the casing is not allowed due to EMC regulations.

STOP

The PE protective conductor must not lead into the casing or be connected to one of the terminals.

Fig. 5: Supply of the PE protective conductor

STOP

When operating the µCAN.8.dio-SNAP, the casing must be closed.

4.2 Installation

To facilitate identification of the µCAN module during operation the casings should be labeled on the lid after assembly. We recommend using the set µCAN module ID for labeling.

When mounting several µCAN modules on a bus line, please make sure that the last module installed to the bus is terminated with a resistor.

4.3 Dismantling

Please make sure that the device is disconnected from power supply first!

Disconnect all signal wires from the connectors. Then, disconnect the CAN bus and the power supply line from the connector.

4

5. Installation

5.1 Potential Basics

The potential environment of our µCAN.8.dio-SNAP modules is characterized by the following features:

- The CAN bus potential is not isolated from the power supply. (Optionally available: galvanic isolation from CAN bus)
- The individual µCAN.8.dio-SNAP modules are not isolated from the power supply.
- All µCAN modules can be supplied separately.
- The I/O signals are not isolated from each other.

5.2 EMC Considerations

EMC (Electromagnetic Compatibility) is the ability of a device to work in a given electromagnetic environment without influencing this environment in an inadmissible way.

All µCAN modules fit these requirements and are tested with regard to the limit values stipulated by law. The µCAN modules are tested through an officially recognized EMC laboratory. However, an EMC plan for the system should be set up in order to exclude potential noise sources.

In automation and measurement technology noise signals can couple in different ways. Depending on that way (guided wave propagation or non-guided wave propagation) and the distance of the noise source the following kinds of coupling can be distinguished.

5

DC Coupling

If two electronic circuits use the same conductor this is called a DC coupling. Potential noise sources in these cases may be: starting motors, frequency converters (switching devices in general) as well as different potentials of casings of components and the common power supply.

Inductance Coupling

An inductance coupling is given between two current-carrying conductors. The current in one conductor will cause a magnetic field which induces a voltage in the second conductor (transformer principle). Typical noise sources are transformers, parallel power lines and RF signal lines.

Capacitive Coupling

A capacitive coupling is given between two conductors which have a different potential (principle of a capacitor). Potential noise sources in these cases may be: parallel conductors, static discharge and contactors.

RF Coupling

RF coupling is given when electromagnetic fields hit a conductor. This conductor acts as an antenna for the electromagnetic field and induces noise to the system. Typical noise sources are spark plugs and electric motors. Radio sets situated near the system may also cause interference.

To reduce the impact of noise sources, please ensure that the basic EMC rules are observed.

5.2.1 Earthing of inactive Metal Parts

All inactive metal plates must be earthed with low impedance. This ensures that all elements of the system will have the same potential.

The earth potential must not carry any dangerous voltage and must be connected to a protective earthing conductor.

STOP

The µCAN modules are equipped with an integrated functional earthing conductor which is situated in the back of the casing. Snapping the module onto the rail simultaneously connects the contact to the rail. The protective conductor must not lead into the casing of the modules.

5.2.2 Shielding of Cables

Any noise signal which works on a cable shield will be grounded to earth by appropriate conductors. The cable shields have to be connected to the protective conductor with low impedance to avoid interference from the shields as well.

Cable Types

For installation of the µCAN modules, please only use cables with a shield covering at least 80% of the core. Do not use cables with a metallized foil shield as these can be easily damaged on assembly and, therefore, do not guarantee proper shielding.

Cable Layout

In general, the cable shield should be earthed on both ends. The cable shield should only be earthed on one end if an attenuation is necessary in the low frequency range. In addition, earthing on both ends is not possible for certain measurement sensors. In these cases, earthing on one end would be an advantage if:

- an equipotential bonding is not possible,
- analogue signals of several mV or mA are to be transmitted (e.g. through the sensors).

STOP

The shield of the CAN bus cable must not lead inside casing of the µCAN module. Never connect the shield to one of the terminals inside the casing.

For stationary applications the shield of the CAN bus cable should be connected to an earthing conductor by metal terminals.

5.3 General Information on Wiring

All wires used within the system should be grouped in different categories. These categories could be e.g.: signal lines, data lines, high-voltage power lines.

High-voltage power lines and data or signal lines should be arranged in separate cable ducts or groups (ref. inductive coupling).

Data and signal lines should lead along ground planes as near as possible.

Observing the rules of proper wiring layout will avoid or impede interference of parallel lines to a large extend.

5.3.1 Groups of wires

In order to achieve a EMC-compliant wiring layout the wires should be categorized as follows:

Group 1: shielded bus and data wires,
shielded analogue wires,
unshielded DC wires < 60V,
unshielded AC wires < 25V,
coaxial wires for monitors

Group 2: unshielded DC wires > 60V and < 400V,
unshielded AC wires >25V and < 400V,

Group 3: unshielded AC / DC wires < 400V

Combination of groups

Based on this categorization the following combinations for arrangement in groups or cable ducts are possible:

Group 1 with group 1, group 2 with group 2, group 3 with group 3

The following groups may be combined in separate cable ducts or groups without a minimum spacing necessary:

Group 1 with group 2

Other combinations of groups can only be realized if they are arranged in separate cable ducts or groups observing the admissible limit values.

5.4 CAN Cable

To connect the devices to the CAN bus an ISO 11898-2 compliant cable must be used. The wires must comply with the following electrical specifications:

Cable characteristics	Value
Impedance	108 - 132 Ohms (nom. 120 Ohms)
Specific impedance	70 m Ohms/meter
Specific signal delay	5 ns/meter

Table 2: Characteristics CAN cable

5.5 Power Supply Voltage

The µCAN.8.dio-SNAP module is designed for industrial applications. The supply voltage may vary within a range from 9V to 36V. The input of the power supply is protected against reverse polarity.

Please make sure that the power supply is correctly connected to the respective screw terminals of the COMBICON plug. The positive line of the power supply for the µCAN module has to be connected to terminal **V+**. The positive line of the power supply for the module has to be connected to terminal **Vp**.

The reference potential of the power supply is connected to one of the **GND** terminals. The GND terminals are internally bridged.

The GND potential of the I/O signals has to be tapped from the GND terminal or the GND of the power supply respectively.

5

Fig. 6: Connection of the power supply

The output drivers may be supplied via their own separate power supply- voltage or a bridge may be placed between **V+** and **Vp**.

The maximum power supply voltage of the **electronics** and **output drivers** is **36V**. Application of higher voltages will destroy the µCAN module.

Even if the digital outputs of the µCAN module are not used, the power supply of the outputs has to be connected in any case.

5.6 CAN Bus

The CAN bus is connected to the appropriate terminal of the COMBICON plug via a two-core wire.

To avoid electromagnetic interference, please ensure that the CAN bus line does not cross the signal lines.

The CAN bus line with high potential must be connected to the terminal **CAN_H**. The CAN bus line with low potential must be connected to the terminal **CAN_L**.

The terminals CAN_H and CAN_L are internally bridged.

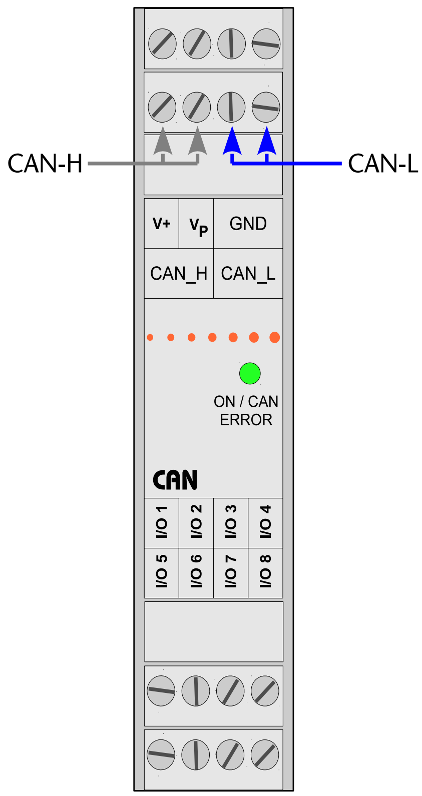


Fig. 7: Connection of CAN bus

Any reversal connection of the bus potentials will prevent communication on the bus.

A shielding must not lead into the µCAN module or be connected to one of the terminals. Shielding must be connected to the appropriate potential via special connectors outside the casing.

If you use a 9-pole Sub-D connector, the high potential has to be connected to pin 7 and the low potential to pin 2 (according to CiA standards). The pinout is shown in the following picture.

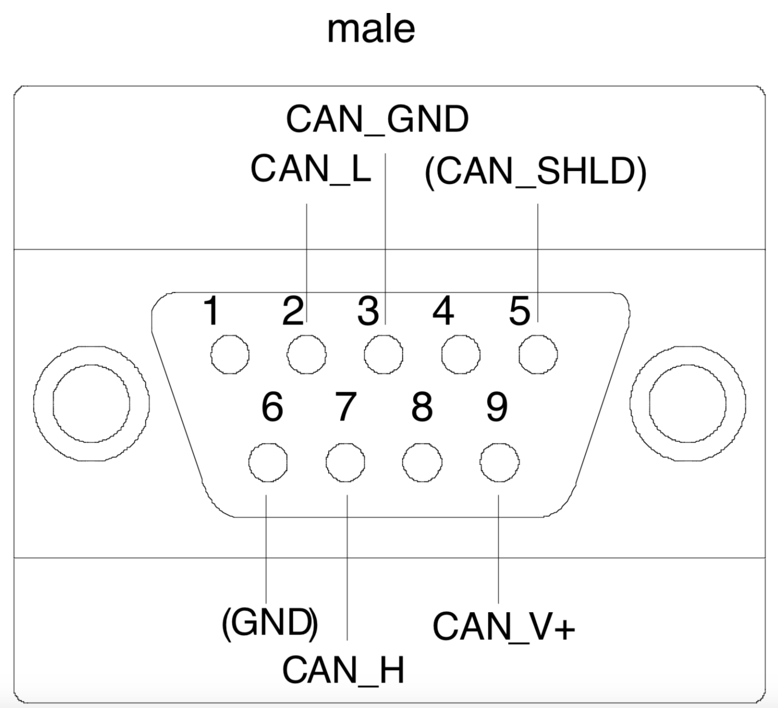


Fig. 8: SUB-D pinout CAN (male)

5.7 Configuration of Address

The address of the µCAN modules is selected via an 8-pin DIP switch which is located on the left in the middle of the board.

To select the address (node-ID or NID) you should use a small-sized screw driver.

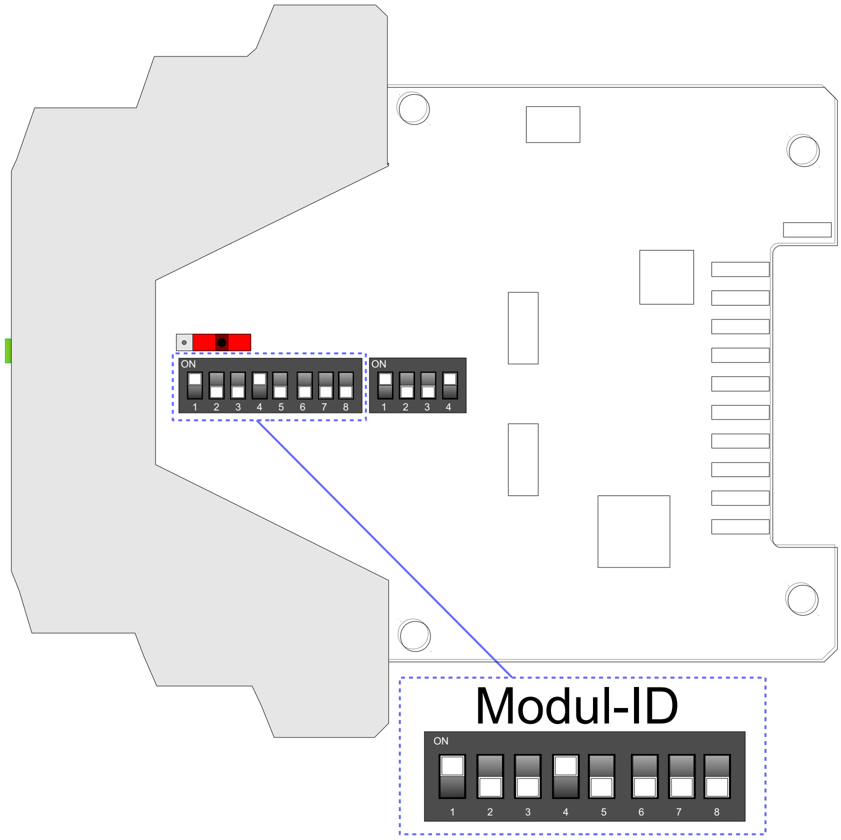


Fig. 9: Set-up of module address (address 9 is shown here)

The 8-pin DIP switch sets the binary code for the module address. The first pin of the switch (marked '1') represents bit 0 of a byte. The last pin of the switch (marked '8') represents bit 7 of a byte.

A valid node-ID is within the range from 1..127 if the CANopen / CANopen FD protocol is used, resp. 01_h..$7F_h$. Valid module addresses are within the range from 0..253 if the J1939 protocol is used, resp. 00_h..FD_h. Each node within a CAN network must have a unique ID. Two nodes with the same ID on a bus line are not allowed.

The selected address is read during initialization of the µCAN module. The µCAN module uses the selected module address until a new module address is selected and a reset is performed.

If all node-ID switches are in OFF position and the DIP switches of the bit rates are switched to OFF as well, the µCAN.8.dio-SNAP will be started in LSS mode.

The µCAN module will not start if the switches are in a wrong position, which will be indicated by the "Error" LED (refer to “Diagnosis” on page 43).

5.8 Configuration of bit-rates

The bit rates of the µCAN modules are selected via a 4-pole DIP switch which is located on the left of the DIP switch for selecting the module address at the top of the board.

This switch is also used to select the communication profile (CANopen, CANopen FD or J1939)

To select the bit rate you should use a small-sized screw driver.

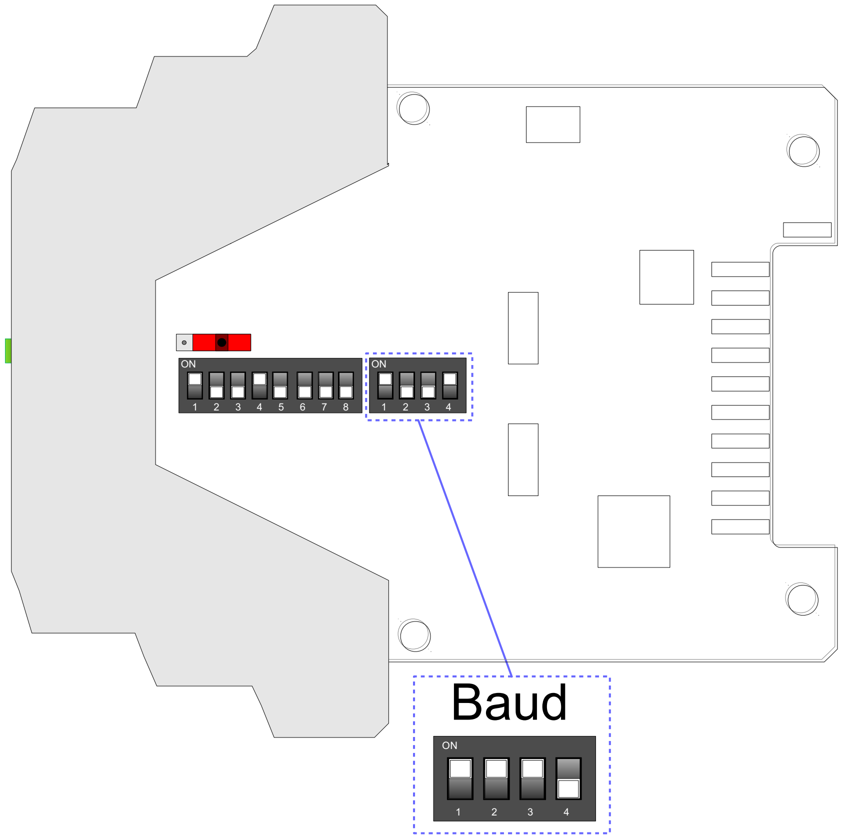


Fig. 10: Configuration of bit rate (500 kBit/s is shown here)

The supported bit rates of the µCAN field modules are listed in the table below. The values are recommended by the CiA.

Protocol	Bit rate	DIP switch position			
		1	2	3	4
CANopen	LSS	0	0	0	0
reserved	reserved	1	0	0	0
reserved	reserved	0	1	0	0
CANopen	50 kBit/s	1	1	0	0
CANopen	100 kBit/s	0	0	1	0
CANopen	125 kBit/s	1	0	1	0
CANopen	250 kBit/s	0	1	1	0
CANopen	500 kBit/s	1	1	1	0
CANopen	800 kBit/s	0	0	0	1
CANopen	1 MBit/s	1	0	0	1
CANopen FD	250 / 1000 kBit/s	0	1	0	1
CANopen FD	250 / 2000 kBit/s	1	1	0	1
CANopen FD	500 / 2000 kBit/s	0	0	1	1
CANopen FD	1000 / 4000 kBit/s	1	0	1	1
J1939	250 kBit/s	0	1	1	1
J1939	500 kBit/s	1	1	1	1

Table 3: Configuration of Bit Rates

Bit rates of 10 kBit/s and 20 kBit/s are not supported by the µCAN.8.dio-SNAP.

If the module is set to LSS mode, the bit rate and module address stored in the module will be applied.

If an inadmissible bit rate has been set on the module, this will be indicated by the "Error" LED (refer to “Diagnosis” on page 43).

5.9 Termination

The last module of a CAN line has to be terminated with a resistor (120 ohms). Hence, the CAN bus line is properly terminated and does not reflect back to the communication lines.

For termination of the module the slide switch has to be switched to the corresponding position using a small-sized screw driver.

Please ensure that the termina-tion is switched on only on µCAN modules which are placed at the end of a CAN line. If the module is disconnected from power supply, you will be able to measure a value of 60 Ohms between the lines CAN-H and CAN-L.

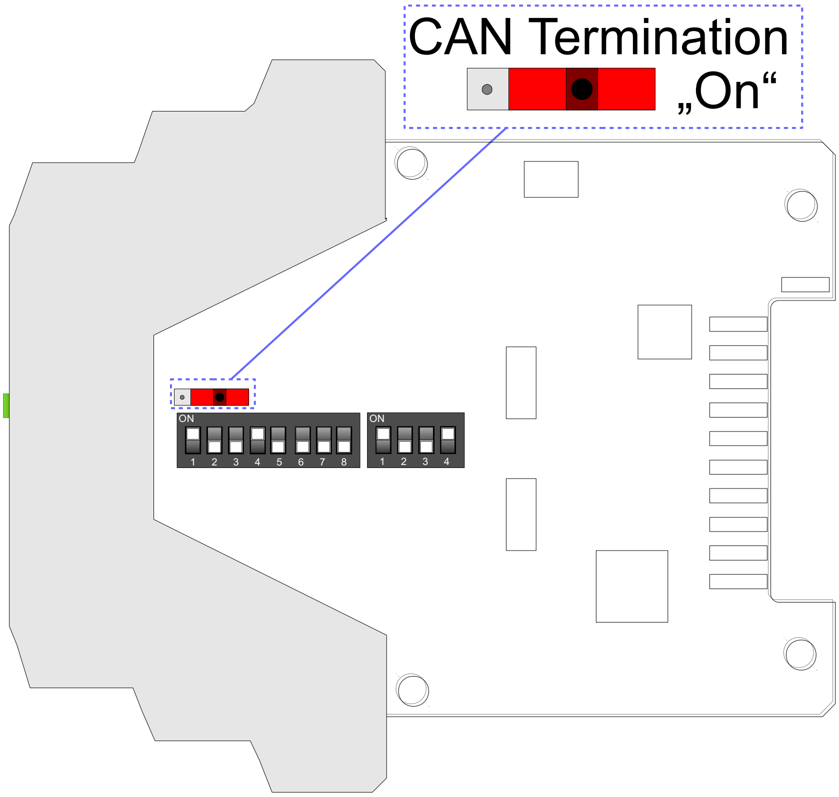


Fig. 11: Switching of termination

In the drawing the termination is switched on.

5

6. Digital Signals

The µCAN.8.dio-SNAP is equipped with eight terminals which can be configured as digital inputs or outputs. The individual terminals are marked with "I/O1" through to "I/O8".

When connecting signal lines, please observe the respective EMC-rules. Only proper, EMC-compliant wiring ensures smooth operation of the µCAN modules.

6.1 Functional Principle

The terminals (inputs or outputs) are configured via the CAN interface. The input voltage of the terminal is compared to a reference voltage which may be set via CANopen interface as well.

If the function "digital output" is selected, the microcontroller will trigger an integrated driver element. The error conditions over current and short circuit will be detected by logic.

In factory setting all terminals are configured as digital inputs. The terminals can only be configured via the CAN interface (refer to “Port direction” on page 93).
In the following, a block diagram explains in which way inputs and outputs are electrically connected with each other.

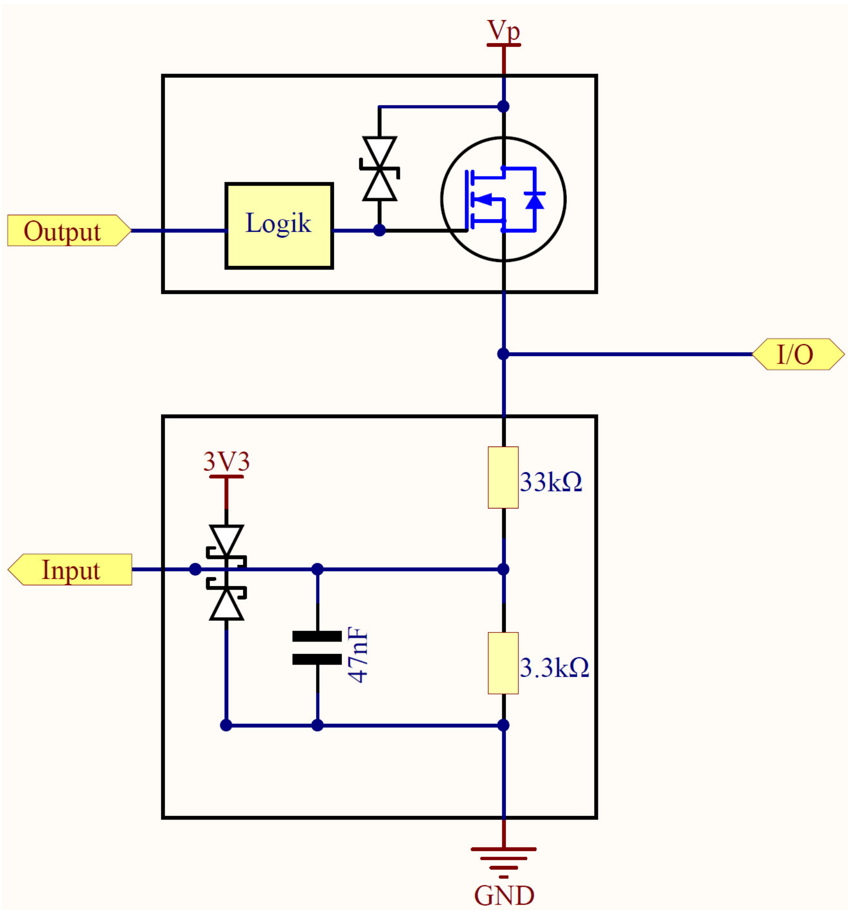


Fig. 12: Block diagram I/O

6.2 Digital Inputs

The digital inputs of the unit can only be switched against the power supply voltage (High-Side).

The maximum voltage applied to the terminals is 36V. Higher voltages cannot be detected or may damage the inputs.

The input voltage is lowered via a voltage divider and measured directly at the analogue-digital converter of the microcontroller.

The switching threshold Vref can be configured via the parameters 5FF0h (refer to “Input level, absolute” on page 90) or 5FF1h (refer to “Input level, relative” on page 91).

6.3 Digital Outputs

In "digital output" mode a voltage is switched to the I/O terminals which corresponds to the "Vp" voltage of the terminal.
A driver element transfers the voltage to the terminals. The driver's properties are as follows.

Parameter	Value
V+PWR	9 .. 36 V
I_{out}	1.0 A maximum

Table 4: Electric parameters high-side driver

The maximum total current of all outputs must not exceed 8.0 A.

6.4 Terminal configuration

The screw terminals of the µCAN.8.dio-SNAP are suited for connection of 8 digital signal lines.

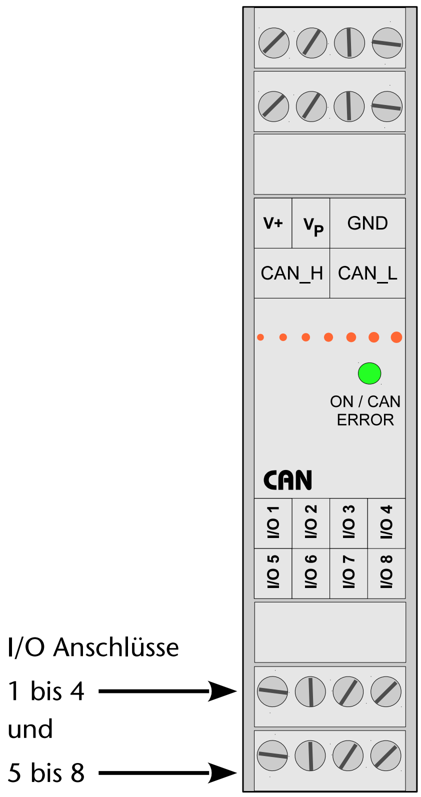


Fig. 13: Connection of signal lines

To connect signal lines, the µCAN modules must be disconnected from power supply to avoid destruction of electronic components.

6

7. Diagnosis

All µCAN modules of the µCAN.8.dio-SNAP type are equipped with an LED for indication of device status and errors.

The µCAN.8.dio-SNAP has a bi-color LED (green/red) marked "ON / CAN ERROR".

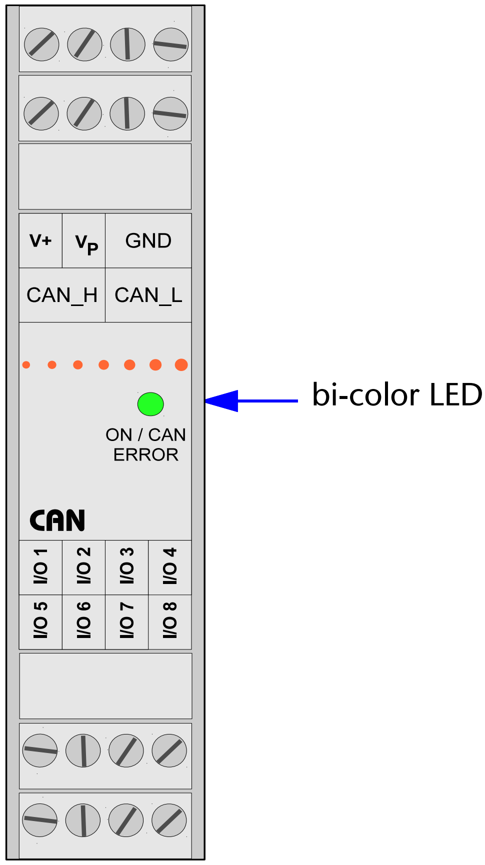
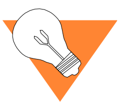


Fig. 14: Position of the bi-color LED on the µCAN module



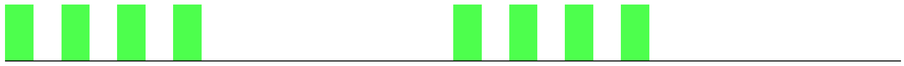
In normal operation the LED should either have a green color or be flashing. A red light, either permanent or flashing indicates an error condition.

7.1 Network State

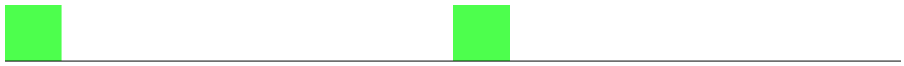
The LED marked "ON/CAN" indicates the condition of the CANopen NMT state machine and the error condition of the CAN controller.

7.1.1 Indication CANopen NMT state

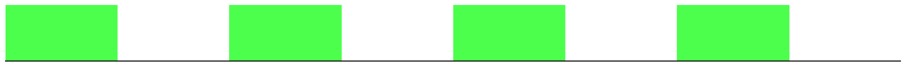
The green light of the LED indicates the CANopen Network Management (NMT) state.



Initialization (autobit detection)



NMT state: device in "Stopped" state

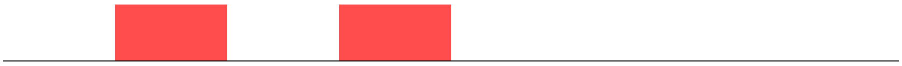


NMT state: device in "Pre-operational" state

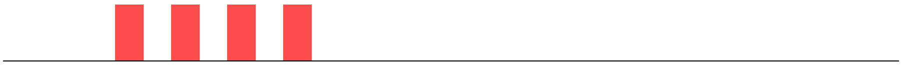
NMT state: device in "Operational" state

7.1.2 Indication CAN state

The red light of the LED indicates the state of the CAN controller and the µCAN module. In non-fault condition the LED is green.



CAN state: controller in "Warning" state

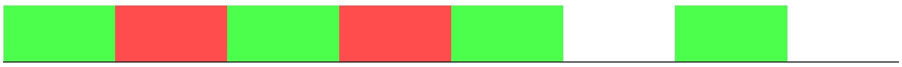


CAN state: controller in "Error Passive" state

CAN state: controller in "Bus-Off" state or
error in set bit rate/device address

7.1.3 Combined indication

The combined indication of red and green light of the LED shows the condition of the CAN controller and the CANopen NMT state.



Device in "Pre-operational" state, CAN controller in "Warning" state

Device in "Operational" state, controller in "Error Passive" state

7

8. CANopen Protocol

The chapter CANopen Protocol contains the most important information for the user on connecting the modules of the CAN series to a CANopen manager and putting them into operation. The CANopen manager can be a PC with CAN card, a PLC or even a control unit or actuator.

For detailed information on the CANopen manager, please refer to the respective documentation of the devices in use.

This operating manual describes the functions of the µCAN.8.dio-SNAP module implemented in the firmware version 3.00. µCAN modules with a later version of the firmware may contain additional objects, services and functions.

8.1 General Information

The CANopen protocol is selected by the DIP switches for Configuration of bit-rates.

The identifiers of the module are set up after initial start-up according the **Predefined Connection Set** which is described in detail in the CANopen communications profile CiA 301. The following table provides an overview of the supported services.

Object	COB-ID (dec.)	COB-ID (hex)
Network Management	0	0x000
SYNC	128	0x080
EMERGENCY	129 - 255	0x081 - 0x0FF
PDO 1 (Transmit)	385 - 511	0x181 - 0x1FF
PDO 1 (receive)	513 - 639	0x201 - 0x27F
PDO 2 (Transmit)	641 - 767	0x281 - 0x2FF
PDO 2 (Receive)	769 - 895	0x301 - 0x37F
PDO 3 (Transmit)	897 - 1023	0x381 - 0x3FF
PDO 3 (Receive)	1025 - 1151	0x401 - 0x47F
PDO 4 (Transmit)	1153 - 1279	0x481 - 0x4FF
PDO 4 (Receive)	1281 - 1407	0x501 - 0x57F
SDO (Transmit)	1409 - 1535	0x581 - 0x5FF
SDO (Receive)	1537 - 1663	0x601 - 0x67F
Heartbeat / Boot-up	1793 - 1919	0x701 - 0x77F

Table 5: Distribution of identifiers

The directions (Transmit / Receive) signify the transfer directions from the µCAN.8.dio-SNAP module to adjacent devices.

8.2 Network Management

The device state (Stop / Pre-Operational / Operational) can be changed by means of Network Management messages.

Start Node

Start Node

ID	*DLC*	*B0*	*B1*
0	2	01h	Node

Node = module address, 0 = all devices in the network

The „Start Node“ command sets the module to Operational state. The module is now able to communicate via PDOs.

	Sender		µCAN.8.dio-SNAP	
		ID [hex] DLC Data [hex]		Comment
1	->	000 2 01 7F		

Sequence 1: Start device with address 127 or 7Fh.

Stop Node

Stop Node

ID	*DLC*	*B0*	*B1*
0	2	02h	Node

Node = module address, 0 = all devices in the network

The „Stop Node“ command sets the module to Stopped state which will prevent communication via SDOs or PDOs.

	Sender		µCAN.8.dio-SNAP	
		ID [hex] DLC Data [hex]		Comment
1	->	000 2 02 7F		

Sequence 2: Set node with address 127 into Stopped state

Pre-Operational

Enter Pre-Operational

ID	DLC	B0	B1
0	2	80h	Node

Node = module address, 0 = all devices in the network

The „Enter Pre-Operational" sets the module to Pre-Operational state which will prevent communication via PDOs.

	Sender		µCAN.8.dio-SNAP
		ID [hex] DLC Data [hex]	Comment
1	->	000 2 80 7F	

Sequence 3: Set module with address 127 to Pre-Operational

Reset Node

Reset Node

ID	DLC	B0	B1
0	2	81h	Node

Node = module address, 0 = all devices in the network

The „Reset Node" will execute a hardware reset of the CAN-node. After reset the device will be set to Pre-Operational state automatically and will send a „Boot-up Message".

	Network Manager		µCAN.8.dio-SNAP
		ID [hex] DLC Data [hex]	Comment
1	->	000 2 81 7F	
2	<-	77F 1 00	Boot-up Message

Sequence 4: Restart the device with address 127

8

8.3 SDO Communication

The parameters of the device (Object Dictionary) are accessed via an SDO channel (Service Data Object). An SDO message is structured as follows:

ID	*DLC*	*B0*	*B1*	*B2*	*B3*	*B4*	*B5*	*B6*	*B7*
	8	CMD	Index		Sub-Index	Data bytes			

The Command Byte (**CMD**) is signified as follows:

SDO client	SDO server	Functions
22_h	60_h	write, size not defined
23_h	60_h	write, 4 bytes
27_h	60_h	write, 3 bytes
$2B_h$	60_h	write, 2 bytes
$2F_h$	60_h	write, 1 byte
40_h	42_h	read, size not defined
40_h	43_h	read, 4 bytes
40_h	47_h	read, 3 bytes
40_h	$4B_h$	read, 2 bytes
40_h	$4F_h$	read, 1 byte

Table 6: Command for SDO expedited message

In the fields **Index** and **Data** the LSB is transmitted first!

For further information about the SDO protocol, please refer to the specification CiA 301, chapter 7.2.4.

8.3.1 SDO abort protocol

In case of an erroneous access to objects, the system will send an error message. An error message is always structured as follows:

ID	*DLC*	*B0*	*B1*	*B2*	*B3*	*B4*	*B5*	*B6*	*B7*
	8	80h	Index		Sub-Index	Abort code			

The ID of the message as well as the index and the sub index refer to the node-ID which has been accessed erroneously.

The error messages may have the following contents:

Abort code	Identification
0504 0001h	Client / server command specifier unknown / not valid
0601 0000h	Attempt to access object not supported
0601 0001h	Attempt to read a write only object
0601 0002h	Attempt to write a read only object
0602 0000h	Object does not exist in the object dictionary
0609 0011h	Sub-index does not exist in the object dictionary

Table 7: SDO abort protocol

8.4 Emergency Message

Emergency messages (EMCY) will automatically be sent by the µCAN module every time an error occurs. Please note the difference between SDO error messages, indicating erroneous access to a SDO object, and a "real" emergency message. If an error occurs for the first time, an error message will be sent. If the error has been eliminated and does not exist anymore, an error message will be sent as well (error code 0000_h).

The identifier of an EMCY message is calculated from the value of the set module address + 128 $_d$.

The emergency message has the following structure:

ID	DLC	B0	B1	B2	B3	B4	B5	B6	B7
	8	Error code		ER	Manufacturer Specific Error Field				

The following error codes are supported:

Error code	Identification
0000h	Error reset or no error
5000h	Module hardware
6000h	Module software
8100h	CAN controller in "Warning" mode
8110h	CAN controller overrun, too many messages
8120h	CAN controller in "Error Passive" mode
8130h	Heartbeat / Node-Guarding Event
8140h	Recover from Bus-Off
8150h	Identifier collision (transmit identifier received)

Table 8: Error codes of the emergency message

In field "ER" (error register) of the emergency message shows the content of the CANopen object 1001h. The „Manufacturer Specific Error Field" is not used.

The module will save all emergency messages in an error history which may be accessed via object 1003h of the CANopen object dictionary.

8

9. CANopen FD Protocol

The chapter CANopen FD Protocol contains the most important information for the user on connecting the modules of the CAN series to a CANopen FD manager and putting them into operation. The CANopen FD manager can be a PC with CAN card, a PLC or even a control unit or actuator.

For detailed information on the CANopen FD manager, please refer to the respective documentation of the devices in use.

This operating manual describes the functions of the µCAN.8.dio-SNAP module implemented in the firmware version 3.00. µCAN modules with a later version of the firmware may contain additional objects, services and functions.

9.1 General Information

The CANopen FD protocol is selected by the DIP switches for Configuration of bit-rates.

The identifiers of the module are set up after initial start-up according the **Predefined Connection Set** which is described in detail in the CANopen FD communications profile CiA 1301. The following of the table provides an overview of the supported services.

Object	COB-ID (dec.)	COB-ID (hex)
Network Management	0	0x000
SYNC	128	0x080
EMERGENCY	129 - 255	0x081 - 0x0FF
PDO 1 (Transmit)	385 - 511	0x181 - 0x1FF
PDO 1 (receive)	513 - 639	0x201 - 0x27F
PDO 2 (Transmit)	641 - 767	0x281 - 0x2FF
PDO 2 (Receive)	769 - 895	0x301 - 0x37F
PDO 3 (Transmit)	897 - 1023	0x381 - 0x3FF
PDO 3 (Receive)	1025 - 1151	0x401 - 0x47F
PDO 4 (Transmit)	1153 - 1279	0x481 - 0x4FF
PDO 4 (Receive)	1281 - 1407	0x501 - 0x57F
USDO (Transmit)	1409 - 1535	0x581 - 0x5FF
USDO (Receive)	1537 - 1663	0x601 - 0x67F
Heartbeat / Boot-up	1793 - 1919	0x701 - 0x77F

Table 9: Distribution of identifiers

9.2 Network Management

The device state (Stop / Pre-Operational / Operational) can be changed by means of Network Management messages.

Start Node

Start Node

ID	*DLC*	*B0*	*B1*
0	2	01h	Node

Node = module address, 0 = all devices in the network

The „Start Node" command sets the module to Operational state. The module is now able to communicate via PDOs.

	Sender		**µCAN.8.dio-SNAP**
		ID [hex] DLC Data [hex]	Comment
1	->	000 2 01 7F	

Sequence 5: Start device with address 127 or 7Fh.

Stop Node

Stop Node

ID	*DLC*	*B0*	*B1*
0	2	02h	Node

Node = module address, 0 = all devices in the network

The „Stop Node" command sets the module to Stopped state which will prevent communication via USDOs or PDOs.

	Sender		**µCAN.8.dio-SNAP**
		ID [hex] DLC Data [hex]	Comment
1	->	000 2 02 7F	

Sequence 6: Set device with address 127 into Stopped state

9

Pre-Operational

Enter Pre-Operational

ID	DLC	B0	B1
0	2	80h	Node

Node = module address, 0 = all devices in the network

The „Enter Pre-Operational“ sets the module to Pre-Operational state which will prevent communication via PDOs.

	Sender	ID [hex]	DLC	Data [hex]	µCAN.8.dio-SNAP Comment
1	->	000	2	80 7F	

Sequence 7: Set module with address 127 to Pre-Operational

Reset Node

Reset Node

ID	DLC	B0	B1
0	2	81h	Node

Node = module address, 0 = all devices in the network

The „Reset Node“ will execute a hardware reset of the module. After reset the device will be set to Pre-Operational state automatically and will send a „Boot-up Message“.

	Network Manager	ID [hex]	DLC	Data [hex]	µCAN.8.dio-SNAP Comment
1	->	000	2	81 7F	
2	<-	77F	1	00	Boot-up Message

Sequence 8: Restart the device with address 127

9

9.3 USDO Communication

The parameters of the device (Object Dictionary) are accessed via an USDO channel (Universal Service Data Object). The USDO message is divided into

- USDO Upload - read data from device
- USDO Download - write data to device

In both cases the destination address is transmitted in byte 0 of the protocol. Admissible values for "Destination address" are 1 to 127 for a specific device or the value 0 for all devices in the network.

The protocol USDO Upload is shown in figure 15 .

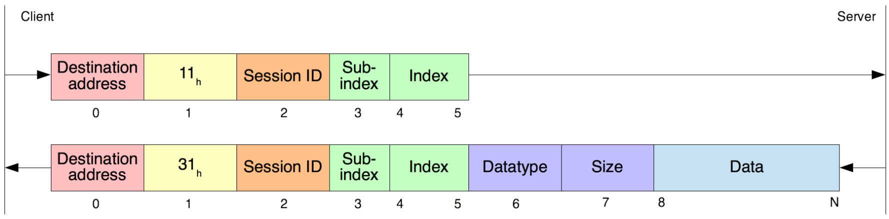


Fig. 15: USDO Upload

The protocol USDO Download is shown in figure 16 .

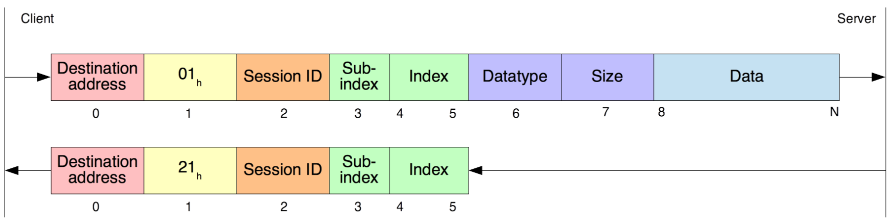
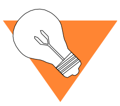


Fig. 16: USDO download

9

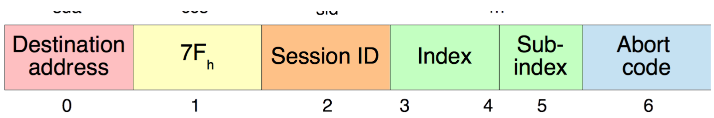


In the fields **Index** and **Data** the LSB is transmitted first!

For further information about the USDO protocol, please refer to the specification CiA 1301, chapter 8.5.

9.3.1 USDO abort protocol

In case of an erroneous access to objects, the system will send an error message. An error message is always structured as follows:



The field Abort Code can have the following content:

Abort Code	**Identification**
13_h	Client / server command specifier unknown / not valid
$1D_h$	USDO node unknown
31_h	Attempt to read a write only object
32_h	Attempt to write a read only object
33_h	Object does not exist in the object dictionary
34_h	Sub-index does not exist in the object dictionary

Table 10: USDO abort protocol

10. J1939 Protocol

The chapter J1939 protocol contains the most important information for the user on connecting the modules of the µCAN series to a J1939 network and putting them into operation.

10.1 General information

The J1939 protocol is selected by the DIP switches for Configuration of bit-rates. The Source Address (SA) is selected via the DIP switch for Configuration of Address.

10.2 Access to the CANopen object dictionary

The J1939 parameter groups CAM11 and CAM21 are defined in the J1939-71/DA as CANopen Application Messages. The CANopen Standard CiA 510 specifies in which way the two parameter groups are allocated to the SDO protocol.

This allocation allows access to the complete CANopen Object Dictionary by the J1939 protocol.

10.3 Parameter groups

The module µCAN.8.dio-SNAP support the parameter groups listed in Table 11.

PGN	Acronym	PG Name
1280	CAM11	CANopen application message 1
1536	CAM21	CANopen application message 2
65242	SOFT	Software identification
65300	PropB_14	Proprietary digital in
65302	PropB_16	Proprietary digital out

Table 11: Parameter groups

CANopen application message 1

PGN 1280

PGN 1280 is used to initiate a SDO Communication (SDO request message).

PGN	Priority	Data Length	Transmission Rate	Multipacket
1280	7	8	On request	No

The data content of the PDU is defined by Table 12.

Byte	Data	Comment
1	SDO command specifier	see CiA 301
2	Object index, LSB	see CiA 301
3	Object index, MSB	see CiA 301
4	Object sub-index	see CiA 301
5	Object data, LSB	
6	Object data	
7	Object data	
8	Object data, MSB	

Table 12: Data content PGN 1280

The response to PGN 1280 is defined by CANopen application message 2.

Sender **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	1C050108	8	40 18 10 04 00 00 00 00	
2	<-	1C060801	8	43 18 10 04 42 14 0A 1E	

Sequenz 9: Read CANopen object 1018:04h_h - device serial number

CANopen application message 2

PGN 1536

PGN 1536 is sent in response to PGN 1280 (see also SDO Communication, SDO Response Message).

PGN	Priority	Data Length	Transmission Rate	Multipacket
1536	7	8	On request	No

The data content of the PDU is defined by Table 13.

Byte	Data	Comment
1	SDO command specifier	see CiA 301
2	Object index, LSB	see CiA 301
3	Object index, MSB	see CiA 301
4	Object sub-index	see CiA 301
5	Object data, LSB	
6	Object data	
7	Object data	
8	Object data, MSB	

Table 13: Data content PGN 1536

Software identification

PGN 65242

PGN 65242 is used to verify the software version of the device (data type **String**).

The software version (data type **Unsigned32**) can also be checked via object 1018:03$_h$ (refer to “Identity object” on page 80).

PGN	Priority	Data Length	Transmission Rate	Multipacket
65242	6	Variable	On request	Yes

The data content of the PDU is defined by J1939-71/DA, the number of independent software version fields is show in byte 1 of the response.

The following CAN trace shows the communication between a sender with the address 8 and the µCAN.8.dio-SNAP, the PGN is queried by a RQST message.

Requester **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	18EAFF08	3	DA FE 00	RQST 65242
2	<-	1CECFF04	8	20 19 00 04 FF DA FE 00	BAM, size=25 byte
3	<-	1CEBFF04	8	01 02 56 65 72 73 69 6F	TPDT, sequence 1
4	<-	1CEBFF04	8	02 6E 20 33 2E 30 34 2A	TPDT, sequence 2
5	<-	1CEBFF04	8	03 32 30 32 31 2F 30 34	TPDT, sequence 3
6	<-	1CEBFF04	8	04 2F 30 38 2A FF FF FF	TPDT, sequence 4

Sequenz 10: Query PGN 65242

The response is interpreted as

Number of software identification fields	2
Field 1	Version 3.04
Field 2	2021/04/08

10

Proprietary digital in

PGN 65300

PGN 65300 is used to query the state of the digital inputs.

The digital inputs also indicate a change in state when the terminal is configured as digital output and the respective output changes its state.

PGN	Priority	Data Length	Transmission Rate	Multipacket
65300	5	8	200 ms	No

The data content of the PDU is defined by Table 15.

Byte	Data	Comment
1.1	Digital Input 1	4 states / 2 bit
1.3	Digital Input 2	4 states / 2 bit
1.5	Digital Input 3	4 states / 2 bit
1.7	Digital Input 4	4 states / 2 bit
2.1	Digital Input 5	4 states / 2 bit
2.3	Digital Input 6	4 states / 2 bit
2.5	Digital Input 7	4 states / 2 bit
2.7	Digital Input 8	4 states / 2 bit
3 .. 8	FF_h	not used

Table 14: Data content PGN 65300

PGN 65300 is sent cyclically every 200 ms.

The following CAN trace shows the communication of a µCAN.8.dio-SNAP with the address 4, in the second CAN message the digital input 1 is reported to the system as set.

	Recipient				µCAN.8.dio-SNAP
		ID [hex]	DLC	Data [hex]	Comment
1	<-	14FF1404	8	00 00 FF FF FF FF FF FF	no input active
2	<-	14FF1404	8	01 00 FF FF FF FF FF FF	input 1 active

Sequenz 11: State of the digital inputs - PGN 65300

10

Proprietary digital out

PGN 65302

PGN 65302 is used to set the state of the digital outputs.

The digital outputs can only be set if the terminals have been defined as outputs via object 5FF5h (refer to “Port direction” on page 93).

PGN	Priority	Data Length	Transmission Rate	Multipacket
65302	5	8	On request	No

The data content of the PDU is defined by Table 15.

Byte	Data	Comment
1.1	Digital Output 1	4 states / 2 bit
1.3	Digital Output 2	4 states / 2 bit
1.5	Digital Output 3	4 states / 2 bit
1.7	Digital Output 4	4 states / 2 bit
2.1	Digital Output 5	4 states / 2 bit
2.3	Digital Output 6	4 states / 2 bit
2.5	Digital Output 7	4 states / 2 bit
2.7	Digital Output 8	4 states / 2 bit
3 .. 8	FF_h	not used

Table 15: Data content PGN 65302

The following CAN trace shows the communication between a-sender with the address 8 and the µCAN.8.dio-SNAP, the second CAN message sets digital output 1.

Sender **µCAN.8.dio-SNAP**

		ID [hex] DLC Data [hex]	Comment
1	->	14FF1608 8 00 00 FF FF FF FF FF FF	no output active
2	->	14FF1608 8 01 00 FF FF FF FF FF FF	output 1 active

Sequenz 12: Set a digital output - PGN 65302

11. Object Dictionary

This chapter describes the objects implemented in the µCAN.8.dio-SNAP module. For further information, please refer to the CANopen communication profiles CiA 301 and CiA 1301 as well as the device profile CiA 401.

EDS

The objects implemented in the µCAN.8.dio-SNAP module are listed in an "Electronic Data Sheet" (EDS). The EDS file named mcan8dio_snap_v3r00.eds may be downloaded from the MicroControl homepage.

11

11.1 Communication profile

The µCAN.8.dio-SNAP module comprises the following objects of the communication profile CiA 301 and CiA 1301:

Index	Name
1000_h	Device type
1001_h	Error register
1002_h	Manufacturer status register
1003_h	Error history
1005_h	COB-ID SYNC
1006_h	Communication cycle period
1008_h	Manufacturer device name
1009_h	Manufacturer hardware version
$100A_h$	Manufacturer software version
1010_h	Store parameters
1011_h	Restore default parameters
1014_h	COB-ID EMCY
1016_h	Consumer heartbeat time
1017_h	Producer heartbeat time
1018_h	Identity object
1029_h	Error behavior
1400_h	RPDO 1 communication parameter
1401_h	RPDO 2 communication parameter
1402_h	RPDO 3 communication parameter
1403_h	RPDO 4 communication parameter
1600_h	RPDO 1 mapping parameter
1601_h	RPDO 2 mapping parameter
1602_h	RPDO 3 mapping parameter
1603_h	RPDO 4 mapping parameter

Table 16: Supported objects of the communications profile

Index	Name
1800_h	TPDO 1 communication parameter
1801_h	TPDO 2 communication parameter
1802_h	TPDO 3 communication parameter
1803_h	TPDO 4 communication parameter
$1A00_h$	TPDO 1 mapping parameter
$1A01_h$	TPDO 2 mapping parameter
$1A02_h$	TPDO 3 mapping parameter
$1A03_h$	TPDO 4 mapping parameter
$1F80_h$	NMT Startup

Table 16: Supported objects of the communications profile

Device type

Index 1000_h

Index 1000_h contains the device type.

Sub-index	Data type	Access	Name	Default value
0	Unsigned32	ro	Device type	0003 0191h

The object is read-only. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

Sender		µCAN.8.dio-SNAP
	ID [hex] DLC Data [hex]	Comment
1	-> 67F 8 40 00 10 00 00 00 00 00	
2	<- 5FF 8 43 00 10 00 91 01 03 00	Response 0003 0191h

Sequence 13: Read object 1000_h from module with node-ID 127

The module response is structured as follows:

Byte 4 + byte 5 = 0191h = 401d = Device-profile number
Byte 6 + byte 7 = 0003h = 3d = Additional information

The value 3_d in byte 6 + 7 denotes that digital inputs as well as digital outputs are defined in the module.

Device type - CANopen FD

Index 1000_h

Index 1000_h contains the device type.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Number of logical devices supported	01h
1	Unsigned32	ro	Device type logical device 1	0003 0191h

The object is read-only. Sub-indices 0 to 1 are supported. Access to other sub-indices will result in an error message.

Sender		µCAN.8.dio-SNAP
	ID [hex] DLC Data [hex]	Comment
1	-> 601 6 7F 11 01 01 00 10	
2	<- 5FF 12 01 31 01 01 00 10 07 04 91 01 03 00	Response 0003 0191h

Sequence 14: Node-ID 127 - read object 1000_h

The module response is equivalent to classical CANopen.

Error register

Index 1001_h

Index 1001h contains the error register of the device.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Error register	00h

The object is read-only. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

Sender **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	40 01 10 00 00 00 00 00	
2	<-	5FF	8	4F 01 10 00 00 00 00 00	

Sequence 15: Read object 1001h from module with node-ID 127

As a response the module will send the state of the error register of the device. In non-fault condition the value 00_h will be read.

For details on the error register refer to the CANopen / CANopen FD communication profiles CiA 301 and CiA 1301.

Manufacturer status register

Index 1002_h

Index 1002_h contains the manufacturer status register of the device.

Sub-index	Data type	Access	Name	Default value
0	Unsigned32	ro	Manufacturer status register	0000 0000h

The object is read-only. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

Sender				µCAN.8.dio-SNAP	
		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	40 02 10 00 00 00 00 00	
2	<-	5FF	8	43 02 10 00 00 00 00 00	

Sequence 16: Read object 1002h from module with node-ID 127

As a response you will get the manufacturer specific status of the µCAN.8.dio-SNAP module. The value 00000000_h will always be read.

11

Error history

Index 1003_h

Via index 1003_h the user may access the error history. Sub-indices 1...4 contain the last 4 errors occurred.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of errors	00h
1..4	Unsigned32	ro	*Pre-defined error field*	0000 0000h

Sub-index 0 is always supported, whereas sub-indices 1 to 4 are supported only if errors have been recorded and sub-index 0 contains a value larger than 0. Access to other sub-indices will result in an error message. Writing a value of 0 to sub-index 0 will erase the error history.

Sender **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	40 03 10 00 00 00 00 00	
2	<-	5FF	8	4F 03 10 00 **03** 00 00 00	number of logged errors
3	->	67F	8	**2F** 03 10 00 00 00 00 00	clear all logged errors
4	<-	5FF	8	60 03 10 00 00 00 00 00	

Sequence 17: Read object 1003h:00h from module with node-ID 127 and clear error history

11

Manufacturer device name

Index 1008_h

Index 1008_h contains the manufacturer device name.

Sub-index	Data type	Access	Name	Default value
0	Visible String	ro	Manufacturer device name	mCAN.8.dio-SNAP

The object is read-only. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

Sender		µCAN.8.dio-SNAP	
	ID [hex] DLC Data [hex]		Comment
1	->	67F 8 40 08 10 00 00 00 00 00	
2	<-	5FF 8 41 08 10 00 **0F** 00 00 00	15 chars to read
3	->	67F 8 60 00 00 00 00 00 00 00	request data
4	<-	5FF 8 00 **6D 43 41 4E 2E 38 2E**	receive "mCAN.8."
5	->	67F 8 70 00 00 00 00 00 00 00	request data
6	<-	5FF 8 10 **64 69 6F 2D 53 4E 41**	receive "dio-SNAP"
7	->	67F 8 60 00 00 00 00 00 00 00	request data
8	<-	5FF 8 0D **50 00 00 00 00 00 00**	receive "P"

Sequence 18: Read object 1008h (device name) from module with node-ID 127

The device name of the µCAN module is transmitted as SDO segmented transfer and is a Visible String.

11

Manufacturer hardware version

Index 1009_h

Index 1009_h contains the manufacturer hardware version.

Sub-index	Data type	Access	Name	Default value
0	Visible String	ro	Manufacturer hardware version	3.04

The object is read-only. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

Sender		µCAN.8.dio-SNAP	
		ID [hex] DLC Data [hex]	Comment
1	->	67F 8 40 09 10 00 00 00 00 00	
2	<-	5FF 8 42 09 10 00 **33 2E 30 34**	receive "3.04"

Sequence 19: Read object 1009h (hardware version) from module with node-ID 127

The hardware version of the µCAN module is transmitted as Visible String and is "3.04" for the present module.

11

Manufacturer software version

Index $100A_h$

Index $100A_h$ contains the manufacturer software version.

Sub-index	Data type	Access	Name	Default value
0	Visible String	ro	Manufacturer software version	-

The object is read-only. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

	Sender	ID [hex]	DLC	Data [hex] µCAN.8.dio-SNAP	Comment
1	->	67F	8	40 0A 10 00 00 00 00 00	
2	<-	5FF	8	42 0A 10 00 **33 2E 30 30**	receive "3.00"

Sequence 20: Read object 100Ah (software version) from module with node-ID 127

The software version of the µCAN module is transmitted as Visible String and is "3.00" in the example sequence.

11

Store parameters

Index 1010_h

Index 1010_h may trigger storing of parameters in a non-volatile memory.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Number of objects	04h
1	Unsigned32	rw	Save all parameters	0000 0001h
2	Unsigned32	rw	Save communication	0000 0001h
3	Unsigned32	rw	Save application	0000 0001h
4	Unsigned32	rw	Save manufacturer	0000 0001h

Storage is triggered by sending index 1010_h with the message "save" (in ASCII code) on the respective sub index.

	Sender			µCAN.8.dio-SNAP	
		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	23 10 10 02 **73 61 76 65**	write "save"
2	<-	5FF	8	60 10 10 02 00 00 00 00	storage was successful

Sequence 21: Save all parameters of module with node-ID 127

As soon as the storage function is initiated the parameters are stored in the non-volatile memory (EEPROM) and a SDO response is sent.

11

Restore default parameters

Index 1011_h

The object at index 1011h supports the restore operation of default parameters.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Number of objects	04h
1	Unsigned32	rw	Restore all param.	0000 0001h
2	Unsigned32	rw	Restore communic.	0000 0001h
3	Unsigned32	rw	Restore application	0000 0001h
4	Unsigned32	rw	Restore manufacturer	0000 0001h

In order to avoid the restoring of default parameters by mistake, restoring is only executed when a specific signature is written to the appropriate sub-index. The signature is "load".

	Sender	ID [hex]	DLC	Data [hex]	µCAN.8.dio-SNAP Comment
1	->	67F	8	23 11 10 02 **6C 6F 61 64**	write "load"
2	<-	5FF	8	60 11 10 02 00 00 00 00	restore was successful

Sequence 22: Restore communications parameters for node-ID 127

The default parameter settings will be applied after restart of the µCAN module.

COB-ID EMCY

Index 1014_h

This object defines the COB-ID for the emergency messages.

Sub-index	Data type	Access	Name	Default value
0	Unsigned32	rw	COB-ID EMCY	80h + node-ID

The default value of the identifier of the emergency message is 80_h + selected node-ID (1 - 127).

Identity object

Index 1018_h

Index 1018_h contains the identity object of the device. Sub-indices 0 to 4 are supported. Access to other sub-indices will result in an error message.

Sub-Index	Datentyp	Zugriff	Bedeutung	Defaultwert
0	Unsigned8	ro	Largest Sub-Index	4
1	Unsigned32	ro	Vendor ID	0000 $000E_h$
2	Unsigned32	ro	Product Code	-
3	Unsigned32	ro	Revision Number	-
4	Unsigned32	ro	Serial Number	-

Vendor ID

The vendor ID is a unique manufacturer specific identification number which is centrally assigned and managed by the CAN in Automation (CiA). Vendor-ID `0000000E`$_h$ has been assigned to MicroControl.

Product Code

The product code is a manufacturer specific code which in case of MicroControl products corresponds to the order number stated in our product catalogue.

Revision Number

The revision number states the software version. The number consists of two 16 bit values. The upper 16 bit values signify a modification in the CAN part of the software, the lower 16 bit values signify a modification of the "application software" of the device.

Serial Number

In response to a query you will receive the serial number of the device.

Error behavior

Index 1029_h

If a serious CANopen device failure is detected and the µCAN module is in Operational mode, the module will be switched to Pre-Operational mode automatically. Via index 1029_h this error behavior may be changed.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	number of entries	1
1	Unsigned8	rw	Communication error	00h

The following values are admissible:

Value	Description
00_h	Standard behavior, change over to pre-operational
01_h	The current NMT mode is not changed
02_h	Change over to NMT mode "stopped"

The following device failures are considered:
error in heartbeat

Sender				µCAN.8.dio-SNAP	
		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	2F 29 10 01 **01** 00 00 00	
2	<-	5FF	8	60 29 10 01 00 00 00 00	

Sequence 23: Set error behavior for node-ID 127 to 01_h

11

NMT Startup

Index $1F80_h$

This object defines the NMT startup behavior of the device.

Sub-index	Data type	Access	Name	Default value
0	Unsigned32	rw	NMT Startup	0000 0000h

Only sub-index 0 is supported. Access to other sub-indices will result in an error message. The object defined the startup behavior after initialization of the device (Power-Up / Reset-Node). The following values are admissible:

Value	Description
00_h	Standard behavior, change over to "pre-operational"
02_h	Send NMT "Start All Nodes"
08_h	Change over to NMT mode "Operational"

11

11.2 Manufacturer specific objects

The µCAN.8.dio-SNAP modules contain the following manufacturer specific objects:

Index	Name
2010_h	Customer data
$201A_h$	COB-ID storage
$2E00_h$	PDO data format
$2E10_h$	Disable bootup message
$2E22_h$	CAN bus statistics
5020_h	Device supply voltage
$5FF0_h$	Input level, absolute
$5FF1_h$	Input level, relative
$5FF2_h$	Input level selection
$5FF4_h$	Input debounce
$5FF5_h$	Port direction
$5FF6_h$	Default output 8-Bit

Table 17: Manufacturer specific objects

11

Customer data

Index 2010_h

Via index 2010h up to 8 32-bit words can be stored in the EE-PROM of the device.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	04h
1	Unsigned32	rw	Customer Data 1	-
2	Unsigned32	rw	Customer Data 2	-
..	..	..	..	..
8	Unsigned32	rw	Customer Data 8	-

Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

Writing access to the sub-indices 1 to 8 will automatically save the value to the EEPROM. Access to object 1010h is not necessary.

Sender		µCAN.8.dio-SNAP			
		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	23 10 20 06 78 56 34 12	write 12345678h
2	<-	5FF	8	60 10 20 06 00 00 00 00	
3	<-	77F	1	00	power reset here
4	<-	67F	8	40 10 20 06 00 00 00 00	
5	<-	5FF	8	43 10 20 06 78 56 34 12	read 12345678h

Sequence 24: Write customer data "12345678_h" into sub-index 6 for node-ID 127

11

COB-ID storage

Index $201A_h$

This object defines the behavior of stored identifiers for PDO and EMCY services when changing the module address.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	COB-ID Storage	00h

Only sub-index 0 is supported. Access to other sub-indices will result in an error message. The following values are admissible:

Value	Description
00_h	Keep stored identifiers (PDO/EMCY) when changing the module address
01_h	Discard stored identifiers (PDO/EMCY) when changing the module address, change over to pre-defined connection set.
02_h	Calculate identifier PDO/EMCY from module address + stored value

The object 201Ah is used in combination with the following objects:

- 1014_h - COB-ID EMCY
- 1400_h - RPDO 1 communication parameter
- 1800_h - TPDO 1 communication parameter

11

PDO data format

Index $2E00_h$

This object stipulates which format, Intel (Little-Endian) or Motorola (Big-Endian), the PDO will use.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	PDO data format	00h

Only sub-index 0 is supported. Access to other sub-indices will result in an error message. The following values are admissible:

Value	Description
00_h	PDO data is transferred in Intel format
01_h	PDO data is transferred in Motorola format

Disable bootup message

Index $2E10_h$

This object defines whether or not the µCAN.8.dio-SNAP module will send a bootup message after switching it on or a reset node command.

Sub-Index	Data type	Access	Name	Default value
0	Unsigned8	rw	Disable Bootup Message	00h

Only sub-index 0 is supported. Access to other sub-indices will result in an error message. The following values are adimissible:

Value	Description
00_h	Bootup message is sent after initialization
01_h	No bootup message is sent

After switching the device on or after a reset the µCAN.8.dio-SNAP runs the initializations procedure and sends the boot-up message when reaching the Pre-Operational state. Sending this message can be deactivated as follows:

Sender | **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	000	2	81 7F	reset node
2	<-	77F	1	00	bootup message
3	->	67F	8	40 00 10 00 00 00 00 00	request device type
4	<-	5FF	8	43 00 10 00 94 01 08 00	
5	->	67F	8	2F 10 2E 00 01 00 00 00	disable bootup message
6	<-	5FF	8	60 10 2E 00 00 00 00 00	response OK
7	<-	5FF	8	22 10 10 01 73 61 76 65	store all parameters
8	<-	5FF	8	60 10 10 01 00 00 00 00	storage OK
9	->	000	2	81 7F	reset node
10	->	67F	8	40 00 10 00 00 00 00 00	request identity
11	<-	5FF	8	43 00 10 00 94 01 08 00	response identity

Sequence 25: Deactivate bootup message for node-ID 127

In the above sequence the µCAN.8.dio-SNAP is re-started via the Reset Node“ *(1)* command. In the next line *(2)* the module sends a bootup message. For testing purposes in line *(3)* the Device type is read.
The boot-up message can be switched off in line *(5)*. The setting is stored afterwards *(7)*. A final reset of the device *(9)* shows, that the module does not sent any boot-up message after re-start.

CAN bus statistics

Index $2E22_h$

Via index $2E22_h$ the end user may view the CAN bus statistics.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Number of entries	03h
1	Unsigned32	ro	CAN Receive Count	-
2	Unsigned32	ro	CAN Transmit Count	-
3	Unsigned32	ro	CAN Error Count	-

Sub-indices 0 to 3 are supported. Access to other sub-indices will result in an error message.

All received, sent or incorrect messages of the module will be counted. The number of counted messages may be read via this object and the correspondent sub-index.

Device supply voltage

Index 5020_h

Index 5020_h may be used to read out the supply voltage of the module. The voltage will be displayed with one decimal place (a multiple of 100 mV).

Sub-index	Data type	Access	Name	Default value
0	Unsigned16	ro	Device supply voltage	-

The object is read-only. No sub-indices are supported. Access to other sub-indices or an attempt to write to this object will result in an error message.

Please note that this value is not calibrated or production tested, so no accuracy is given in the technical specification section.

Input level, absolute

Index $5FF0_h$

Index $5FF0_h$ sets the absolute value of the input reference voltage of the module.

Sub-index	Data type	Access	Name	Default value
0	Unsigned16	rw	Input level, absolute	25

The voltage may be set with a resolution of 100mV.

If a reference voltage is to be set to an absolute value of 4.5V, the message is as follows:

	Sender		µCAN.8.dio-SNAP	
		ID [hex]	DLC Data [hex]	Comment
1	->	67F	8 2B F0 5F 00 2D 00 00 00	set level 4.5 V
2	<-	5FF	8 60 F0 5F 00 00 00 00 00	

Sequence 26: Set input level to 4.5 V

The object is read and write. No sub-indices are supported. Access to other sub-indices will result in an error message.

11

Input level, relative

Index $5FF1_h$

Index $5FF1_h$ sets the relative value of the supply voltage of the input reference voltage.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Input level, relative	50

The relative value may be set from 0% to 80%.

The object is read and write. No sub-indices are supported. Access to other sub-indices will result in an error message.

Input level selection

Index 5FF2h

Index $5FF2_h$ can be switched from absolute and relative value of the input reference voltage.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Input level, selection	0

The object is read and write. No sub-indices are supported. Access to other sub-indices will result in an error message.

Admissible values are as follows:
- 0 - absolute input level
- 1 - relative input level

Input debounce

Index $5FF4_h$

In index $5FF4_h$ the debounce time for each input can be set in milliseconds.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Port direction	00h

The object is read and write. Only sub-index 0 is supported. Access to other sub-indices will result in an error message. The value is given in miltiples of 1 milli-seconds.

The following example shows setting of the debounce time to 50 ms.

Sender **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	2F F4 5F 00 32 00 00 00	write 50 ms
2	<-	5FF	8	60 F4 5F 00 00 00 00 00	

Sequence 27: Set debounce time of 50 ms

In factory default the value is set to 0.

Port direction

Index $5FF5_h$

The object at index $5FF5_h$ is used to modify the port direction of each terminal.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Port direction	00h

The object is read and write. Only sub-index 0 is supported. Access to other sub-indices will result in an error message. By writing a logic 1 the terminal is set as output.

In the following example Terminals 1 - 4 are defined as outputs:

Sender **μCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	2B F5 5F 00 0F 00 00 00	write 0Fh
2	<-	5FF	8	60 F5 5F 00 00 00 00 00	

Sequence 28: Configure terminal 1 - 4 as outputs

In factory default all terminals are defined as digital inputs. The output can only be set if the object $5FF5_h$ has been set accordingly.

Default output 8-Bit

Index $5FF6_h$

index $5FF6_h$ is used to configure a default state of the output terminals after power-on or NMT reset.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Default output	00h

The object is read and write. Only sub-index 0 is supported. Access to other sub-indices will result in an error message. By writing a logic 1 the terminal is set as output.

The following example shows configuration of terminals 1 - 4 having the default state 1.

Sender **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	2B F6 5F 00 0F 00 00 00	write 0Fh
2	<-	5FF	8	60 F6 5F 00 00 00 00 00	

Sequence 29: Set terminals 1 .. 4 as output with default 1

In factory setting all terminals are defined as digital inputs. By writing to this object, also the value in Port direction is updated.

11.3 Device Profile

The µCAN.8.dio-SNAP module contains the following objects of the device profile CiA 401:

Index	Name
6000_h	Read input 8-Bit
6002_h	Polarity input 8-Bit
6005_h	Global interrupt enable
6006_h	Interrupt mask any change
6007_h	Interrupt Mask Low to High
6008_h	Interrupt mask High to Low
6200_h	Write output 8-Bit
6202_h	Polarity output 8-Bit
6206_h	Error mode output
6207_h	Error value output

Table 18: Supported objects of the device profile CiA 401

Read input 8-Bit

Index 6000_h

By a read operation from index 6000_h the current state of the digital inputs can be retrieved.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	ro	Read Input 1 - 8	-

The object is read-only. Sub-indices 0 and 1 are supported. Access to other sub-indices will result in an error message.

The following example shows the read operation:

Sender					µCAN.8.dio-SNAP
		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	40 00 60 01 00 00 00 00	read digital inputs
2	<-	5FF	8	60 00 60 01 01 00 00 00	

Sequence 30: Read digital inputs from module with node-ID 127

In this example, input 1 has a logic high level, all other inputs have the value 0.

11

Polarity input 8-Bit

Index 6002_h

By means of index 6002_h the polarity of the digital inputs can be changed.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	rw	Polarity Input 1 - 8	00h

The object is read and write. Sub-indices 0 and 1 are supported. Access to other sub-in-dices will result in an error message.

Global interrupt enable

Index 6005_h

The object at index 6005_h enables and disables globally the interrupt behavior without changing the interrupt masks.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Global Interrupt ena.	01h

The object is read and write. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

The default value of 01_h enables transmission of a PDO for each digital input. Setting a value of 00_h will disable the transmissions of a PDO.

The object is used in combination with the objects at index 6006_h, 6007_h and 6008_h.

Interrupt mask any change

Index 6006_h

This object determines which input port lines activate an interrupt by positive and negative edge detection.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	rw	Interrupt Any Change	FFh

The object is read and write. Sub-indices 0 and 1 are supported. Access to other sub-indices will result in an error message.

In factory default setting each input will send a PDO message on rising and falling signals. Sending PDO messages can be suppressed if the bit of the respective input is set to zero.

Interrupt Mask Low to High

Index 6007_h

This object determines which input port lines activates an interrupt by positive edge detection (logical 0 to 1). This is done for groups of 8 lines.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	rw	Interrupt Low to High	00h

The object is read and write. Sub-indices 0 and 1 are supported. Access to other sub-in-dices will result in an error message.

Interrupt mask High to Low

Index 6008_h

This object determines which input port lines activates an interrupt by negative edge detection (logical 1 to 0). This is done for groups of 8 lines.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	rw	Interrupt High to Low	00h

The object is read and write. Sub-indices 0 and 1 are supported. Access to other sub-in-dices will result in an error message.

Write output 8-Bit

Index 6200_h

The object at index 6200_h accesses the digital outputs of the module.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	rw	Write Output 1 - 8	00h

The object is read and write. Sub-indices 0 and 1 are supported. Access to other sub-indices will result in an error message.

The follwong example shows setting of output line 8:

Sender **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	2F 00 62 01 80 00 00 00	set output line 8
2	<-	5FF	8	60 00 62 00 00 00 00 00	

Sequence 31: Set output line 8

The digital outputs can only be set if the terminals have been defined as outputs via object 5FF5h (refer to “Port direction” on page 93).

Polarity output 8-Bit

Index 6202_h

By means of index 6202_h the polarity of the digital outputs can be changed.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	rw	Polarity Output 1 - 8	00h

The object is read and write. Sub-indices 0 and 1 are supported. Access to other sub-indices will result in an error message.

Error mode output

Index 6206_h

This object indicates whether an output is set to a pre-defined error value (see 6207_h) in case of an internal device failure or a 'Stop Remote Node' indication.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	rw	Error mode output 1 - 8	FFh

The object is read and write. Sub-indices 0 and 1 are supported. Access to other sub-indices will result in an error message.

Error value output

Index 6207_h

This object indicates to which value the outputs shall be set at device failures if the corresponding error mode (see 6206_h) is active.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	01h
1	Unsigned8	rw	Error value output 1 - 8	00h

The object is read and write. Sub-indices 0 and 1 are supported. Access to other sub-indices will result in an error message.

11.4 Network Variables

The µCAN.8.dio-SNAP module contains the following objects of the application profile CiA 302-4:

Index	Name	PDO
$A040_h$	Input network variable - Unsigned8	TPDO
$A100_h$	Input network variable - Unsigned16	TPDO
$A200_h$	Input network variable - Unsigned32	TPDO
$A4C0_h$	Output network variable - Unsigned8	RPDO
$A580_h$	Output network variable - Unsigned16	RPDO
$A680_h$	Output network variable - Unsigned32	RPDO

Table 19: Supported objects of the application profile CiA 302-4

All input network variables may be mapped to the TPDOs, all output network variables may be mapped to the RPDOs to relocate the process value within the PDO or to indicate gaps between two process values.

Input network variable - Unsigned8

Index $A040_h$

Via index $A040_h$ eight input network variables of data type `UNSIGNED8` are defined.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	08h
1	Unsigned8	rw	Network variable 1	0
2	Unsigned8	rw	Network variable 2	0
..	..	..	..	..
8	Unsigned8	rw	Network variable 8	0

Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

A writing access to sub-index 1 to 8 will temporarily save the value in the respective network variable. The content of the network variable is set to the value 0 in case of a NMT Reset Node command.

Input network variable - Unsigned16

Index $A100_h$

Via index $A100_h$ four output network variables of data type `UNSIGNED16` are defined.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	04h
1	Unsigned16	rw	Network variable 1	0
2	Unsigned16	rw	Network variable 2	0
3	Unsigned16	rw	Network variable 3	0
4	Unsigned16	rw	Network variable 4	0

Sub-indices 0 to 4 are supported. Access to other sub-indices will result in an error message.

A writing access to sub-index 1 to 4 will temporarily save the value in the respective network variable. The content of the network variable is set to the value 0 in case of a NMT Reset Node command.

Input network variable - Unsigned32

Index A200$_h$

Via index A200h two output network variables of data type UNSIGNED32 are defined.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	02h
1	Unsigned32	rw	Network variable 1	0
2	Unsigned32	rw	Network variable 2	0

Sub-indices 0 to 2 are supported. Access to other sub-indices will result in an error message.

A writing access to sub-index 1 to 2 will temporarily save the value in the respective network variable. The content of the network variable is set to the value 0 in case of a NMT Reset Node command.

Output network variable - Unsigned8

Index A4C0$_h$

Via index A4C0$_h$ eight output network variables of data type UNSIGNED8 are defined.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	08h
1	Unsigned8	rw	Network variable 1	0
2	Unsigned8	rw	Network variable 2	0
..	..	..	..	..
8	Unsigned8	rw	Network variable 8	0

Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

A writing access to sub-index 1 to 8 will temporarily save the value in the respective network variable. The content of the network variable is set to the value 0 in case of a NMT Reset Node command.

Output network variable - Unsigned16

Index $A580_h$

Via index $A580_h$ four output network variables of data type `UNSIGNED16` are defined.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	04h
1	Unsigned16	rw	Network variable 1	0
2	Unsigned16	rw	Network variable 2	0
3	Unsigned16	rw	Network variable 3	0
4	Unsigned16	rw	Network variable 4	0

Sub-indices 0 to 4 are supported. Access to other sub-indices will result in an error message.

A writing access to sub-index 1 to 4 will temporarily save the value in the respective network variable. The content of the network variable is set to the value 0 in case of a NMT Reset Node command.

Output network variable - Unsigned32

Index $A680_h$

Via index A680h two output network variables of data type `UNSIGNED32` are defined.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	02h
1	Unsigned32	rw	Network variable 1	0
2	Unsigned32	rw	Network variable 2	0

Sub-indices 0 to 2 are supported. Access to other sub-indices will result in an error message.

A writing access to sub-index 1 to 2 will temporarily save the value in the respective network variable. The content of the network variable is set to the value 0 in case of a NMT Reset Node command.

11.5 Device Monitoring

For monitoring of CANopen devices the Heartbeat protocol is used.

CAN in Automation recommends using the heartbeat protocol for monitoring only (acc. to CiA AN 802 V1.0: CANopen statement on the use of RTR-messages).

11.5.1 Heartbeat protocol

Via Heartbeat protocol other nodes on the network are able to check proper functioning and condition of the module.

Heartbeat ID

The identifier through which the module sends a heartbeat is set to 700h + module ID and cannot be changed. The message repetition time (called Producer Heartbeat Time) may be set via index 1017_h.

The heartbeat protocol transmits one byte of user data which represents the network state.

Network state	Code (dec.)	Code (hex)
Bootup	0	00_h
Stopped	4	04_h
Operational	5	05_h
Pre-Operational	127	$7F_h$

After Power-on the module will automatically send a „bootup message“.

Sender **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	702	1	00	

Sequence 32: Bootup message of a node with module address 2

11

Consumer heartbeat time

Index 1016_h

The object at index 1016_h defines the consumer heartbeat time.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Number of objects	2
1	Unsigned32	rw	Heartbeat cons. 1	0000 0000h
2	Unsigned32	rw	Heartbeat cons. 2	0000 0000h

The µCAN.8.dio-SNAP can monitor the presence of two other devices (heartbeat producer) in the network. If a heartbeat producer message is not received within an adjustable period, an emergency message with value 8130_h (life guard error or heartbeat error) is transmitted. The 32-bit value of the object defines heartbeat time and the producers node address.

Bit 31 ... 24	Bit 23 ... 16	Bit 15 ... 0
reserved (00h)	node-ID	time

The time value is stated in milliseconds. If the value 0 is selected for time or a value 0 or higher than 127 for the node-ID, the consumer heartbeat is not activated. The consumer heartbeat monitoring will be activated after the first producer heartbeat has been received.

Producer heartbeat time

Index 1017_h

The object at index 1017_h defines the cycle time of the heartbeat. Selecting a time value of 0 will switch off the heartbeat protocol. The time is a multiple of 1ms.

Sub-index	Data type	Access	Name	Default value
0	Unsigned16	rw	Producer Time	0000_h

The object is read and write. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

Sender		ID [hex]	DLC	Data [hex]	µCAN.8.dio-SNAP Comment
1	->	67F	8	22 17 10 E8 03 00 00 00	heartbeat time 1000 ms
2	<-	5FF	8	60 17 10 00 00 00 00 00	

Sequence 33: Set producer heartbeat time to 1000ms

The producer heartbeat time is not automatically stored in a non-volatile memory. Store parameters has to be triggered via index 1010_h.

11.6 RPDO Communication

Process data sent to the device is received by means of Receive Process Data Objects (RPDOs).

RPDO communication is possible only in "Operational" mode of the device.

The µCAN.8.dio-SNAP module is equipped with four RPDOs.

11.6.1 RPDO communication parameter

By means of the RPDO communication parameter it is possible:

- to enable or disable the RPDO
- to configure the RPDO identifier

The RPDO communication parameters are configured via the following objects:

- 1400_h - RPDO 1 communication parameter
- 1401_h - RPDO 2 communication parameter
- 1402_h - RPDO 3 communication parameter
- 1403_h - RPDO 4 communication parameter

COB-ID for PDO

The CAN identifier for the RPDO is set via sub index 1 and defined by the following table.

Bit 31	Bit 30	Bit 29	Bit 28 - 0
PDO valid, 0 = valid 1 = not valid	RTR allowed 0 = yes 1 = no RTR	Frame type 0 = 11 Bit 1 = 29 Bit	Identifier

To enable the PDO the most significant bit (bit 31) must be cleared. To disable the PDO the most significant bit must be set. Bit 30 (no RTR) is always set and can not be cleared.

Transmission type

The transmission type is set via sub-index 2.

Transmission type	Description
01h .. F0h (1 - 240 dec)	cyclic synchronous, The data of the last received PDO is processed after every n SYNC message
FEh .. FFh (254 - 255 dec)	event-driven, The PDO data is process on reception

RPDO 1 communication parameter

Index 1400_h

Via index 1400_h the communication parameters of RPDO 1 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	2
1	Unsigned32	rw	COB-ID for PDO	200_h + Node-ID
2	Unsigned8	rw	Transmission type	FF_h

The object is read and write. Sub-indices 0 to 2 are supported. Access to other sub-indices will result in an error message. In standard setting the RPDO 1 is active.

RPDO 2 communication parameter

Index 1401_h

Via index 1401_h the communication parameters of RPDO 2 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	2
1	Unsigned32	rw	COB-ID for PDO	80000300_h + Node-ID
2	Unsigned8	rw	Transmission type	FF_h

The object is read and write. Sub-indices 0 to 2 are supported. Access to other sub-indices will result in an error message. In standard setting the RPDO 2 is inactive.

RPDO 3 communication parameter

Index 1402_h

Via index 1402_h the communication parameters of RPDO 3 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	2
1	Unsigned32	rw	COB-ID for PDO	80000400_h + Node-ID
2	Unsigned8	rw	Transmission type	FF_h

11

The object is read and write. Sub-indices 0 to 2 are supported. Access to other sub-indices will result in an error message. In standard setting the RPDO 3 is inactive.

RPDO 4 communication parameter

Index 1403_h

Via index 1403_h the communication parameters of RPDO 4 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Largest sub-index	2
1	Unsigned32	rw	COB-ID for PDO	80000500_h + Node-ID
2	Unsigned8	rw	Transmission type	FF_h

The object is read and write. Sub-indices 0 to 2 are supported. Access to other sub-indices will result in an error message. In standard setting the RPDO 4 is inactive.

11.6.2 RPDO mapping parameter

The user may perform an individual mapping of the data for each RPDO. The µCAN.8.dio-SNAP module supports various objects which can be mapped into the RPDOs.

On the one hand, there are the process values which are used to set the digital outputs, and on the other hand the network variables form markers which are used to design the RPDO.

The following objects for the process value may be mapped into the RPDOs:

Name	Channel	Mapping entry
Write output 8-Bit	1	`6200 01 08h`

Table 20: Supported objects for RPDO mapping of process values

In addition to the mapping entries for the process values, the user can also use entries for the network variables. These may be used to "move" the process values within a PDO or to form "gaps" between two process values.

For this reason, network variable can have a length of 1 byte, 2 byte and 4 byte. Each network variable may repeatedly be used for mapping.

The contents of the network variables are not analyzed in the µCAN.8.dio-SNAP module.

The following objects for the network variables may be mapped into the receive-PDOs:

Name	Variable	Mapping entry
Output network variable - Unsigned8	1	A4C0 01 08h
Output network variable - Unsigned8	2	A4C0 02 08h
Output network variable - Unsigned8	3	A4C0 03 08h
Output network variable - Unsigned8	4	A4C0 04 08h
Output network variable - Unsigned8	5	A4C0 05 08h
Output network variable - Unsigned8	6	A4C0 06 08h
Output network variable - Unsigned8	7	A4C0 07 08h
Output network variable - Unsigned8	8	A4C0 08 08h
Output network variable - Unsigned16	1	A580 01 10h
Output network variable - Unsigned16	2	A580 02 10h
Output network variable - Unsigned16	3	A580 03 10h
Output network variable - Unsigned16	4	A580 04 10h
Output network variable - Unsigned32	1	A680 01 20h
Output network variable - Unsigned32	2	A680 02 20h

Table 21: Supported objects for RPDO mapping of network variables

All the objects listed above can be mapped into each of the four RPDOs. For each RPDO between 0 and 8 mapping entries may be configured. Please note that the total number of entries within a PDO must not exceed 8 byte.

Each mapping entry consists of *index*, *sub-index* and *bit-length*, the respective values are listed in Table 20 and Table 21.

RPDO 1 mapping parameter

Index 1600_h

Via index 1600_h the mapping parameters of RPDO 1 can be configured.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of map-ped objects	1
1	Unsigned32	rw	1^{st} mapped object	6200 01 08h
2	Unsigned32	rw	2^{nd} mapped object	-
3	Unsigned32	rw	3^{rd} mapped object	-
4	Unsigned32	rw	4^{th} mapped object	-
5	Unsigned32	rw	5^{th} mapped object	-
4	Unsigned32	rw	6^{th} mapped object	-
4	Unsigned32	rw	7^{th} mapped object	-
4	Unsigned32	rw	8^{th} mapped object	-

The object is read and write. Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

In standard setting RPDO 1 is *active* and contains *one* mapping entry.

Table 20 and Table 21 show all supported entries of the µCAN.8.dio-SNAP module. Chapter “RPDO mapping configuration” on page 116 describes all steps necessary to perform the mapping.

RPDO 2 mapping parameter

Index 1601_h

Via index 1601h the mapping parameters of RPDO 2 can be configured.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of mapped objects	0
1	Unsigned32	rw	1^{st} mapped object	-
2	Unsigned32	rw	2^{nd} mapped object	-
3	Unsigned32	rw	3^{rd} mapped object	-
4	Unsigned32	rw	4^{th} mapped object	-
5	Unsigned32	rw	5^{th} mapped object	-
4	Unsigned32	rw	6^{th} mapped object	-
4	Unsigned32	rw	7^{th} mapped object	-
4	Unsigned32	rw	8^{th} mapped object	-

The object is read and write. Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

In standard setting the RPDO 2 is *inactive* and contains *no* mapping entry.

Table 20 and Table 21 show all supported entries of the µCAN.8.dio-SNAP module. Chapter “RPDO mapping configuration” on page 116 describes all steps necessary to perform the mapping.

RPDO 3 mapping parameter

Index 1602h

Via index 1602h the mapping parameters of RPDO 3 can be configured.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of mapped objects	0
1	Unsigned32	rw	1^{st} mapped object	-
2	Unsigned32	rw	2^{nd} mapped object	-
3	Unsigned32	rw	3^{rd} mapped object	-
4	Unsigned32	rw	4^{th} mapped object	-
5	Unsigned32	rw	5^{th} mapped object	-
4	Unsigned32	rw	6^{th} mapped object	-
4	Unsigned32	rw	7^{th} mapped object	-
4	Unsigned32	rw	8^{th} mapped object	-

The object is read and write. Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

In standard setting the RPDO 3 is *inactive* and contains *no* mapping entries.

Table 20 and Table 21 show all supported entries of the µCAN.8.dio-SNAP module. Chapter “RPDO mapping configuration” on page 116 describes all steps necessary to perform the mapping.

RPDO 4 mapping parameter

Index 1603h

Via index 1603h the mapping parameters of RPDO 2 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of mapped objects	0
1	Unsigned32	rw	1st mapped object	-
2	Unsigned32	rw	2nd mapped object	-
3	Unsigned32	rw	3rd mapped object	-
4	Unsigned32	rw	4th mapped object	-
5	Unsigned32	rw	5th mapped object	-
4	Unsigned32	rw	6th mapped object	-
4	Unsigned32	rw	7th mapped object	-
4	Unsigned32	rw	8th mapped object	-

The object is read and write. Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

In standard setting the RPDO 4 is *inactive* and contains *no* mapping entry.

Table 20 and Table 21 show all supported entries of the µCAN.8.dio-SNAP module. Chapter “RPDO mapping configuration” on page 116 describes all steps necessary to perform the mapping.

11.6.3 RPDO mapping configuration

In default setting, the µCAN.8.dio-SNAP module contains a standard mapping for the first PDO. The standard mapping allows the digital outputs being written to (see “RPDO 1 mapping parameter” on page 113).

If the environment requires:

- a different mapping of the process values within a PDO
- several mappings distributed among several PDOs
- different data type mappings of the process values

these requirements may be adapted through a PDO mapping configuration.

Table 20 and Table 21 show all supported entries of the µCAN.8.dio-SNAP module.

The following conditions must be fulfilled to be able to configure mapping successfully:

- PDO communication must be disabled
- PDO mapping must be disabled
- The mapping entries may not contain any gaps
- The total number of mapping entries within a PDO must not exceed 8 entries.

The PDO mapping entries are adapted according to the following procedure:

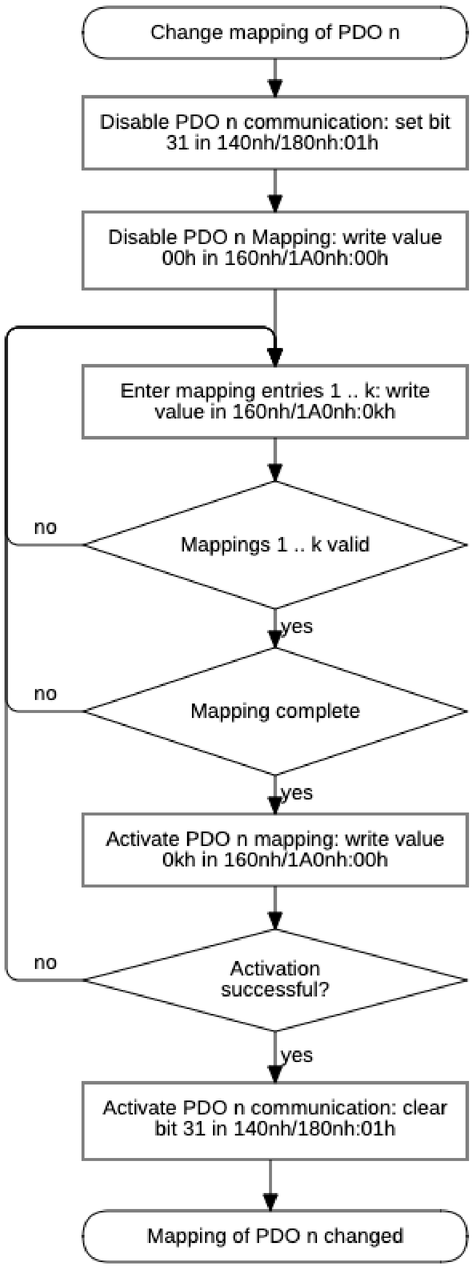


Fig. 17: Workflow adaptation of the PDO mapping

The workflow describes the usual procedure for adapting the PDO mapping, where *n* represents the desired PDO number 1..4 and *k* the mapping entry 1..8.

The communication and mapping parameters of the PDO are not automatically saved in a non-volatile memory. Storage must be triggered via Index 1010h.

11.6.4 RPDO reception example

In default setting only the first RPDO is active and object Write output 8-Bit is mapping in the first byte.

Sender					µCAN.8.dio-SNAP
		ID [hex]	DLC	Data [hex]	Comment
1	<-	77F	1	00	bootup message
2	->	000	2	01 00	set all nodes to OP
3	->	27F	1	01	set digital output 1

Sequence 34: Set new process values for module with node-ID 127 via RPDO 1

As soon as the µCAN.8.dio-SNAP module is ready to operate it will send a bootup message (1). Afterwards, it is set to Operational mode via a "NMT - Start all nodes" message (2).
Sending the first RPDO will set the digital outputs (3).

11

11.7 TPDO Communication

Process data sent from the device is transmitted by means of Transmit Process Data Objects (TPDOs).

TPDO communication is possible only in "Operational" mode of the device.

The µCAN.8.dio-SNAP module is equipped with four TPDOs.

11.7.1 TPDO communication parameter

By means of the TPDO communication parameter it is possible:

- to enable or disable the TPDO
- to configure the TPDO identifier
- to change the TPDO transmission type
- to configure an event timer

The TPDO communication parameter are configured via the following objects:

- 1800_h - TPDO 1 communication parameter
- 1801_h - TPDO 2 communication parameter
- 1802_h - TPDO 3 communication parameter
- 1803_h - TPDO 4 communication parameter

COB-ID for PDO

The CAN identifier for the TPDO is set via sub index 1 and defined by the following table.

Bit 31	Bit 30	Bit 29	Bit 28 - 0
PDO valid, 0 = valid 1 = not valid	RTR allowed 0 = yes 1 = no RTR	Frame type 0 = 11 Bit 1 = 29 Bit	Identifier

Table 22: COB-ID setting of TPDO

To enable the PDO the most significant bit (bit 31) must be cleared. To disable the PDO the most significant bit must be set. Bit 30 (no RTR) is always set and can not be cleared.

Transmission type

The transmission type is set via sub-index 2.

Transmission type	Description
01h .. F0h (1 - 240 dec)	cyclic synchronous, TPDO is processed after every n SYNC message
FEh .. FFh (254 - 255 dec)	event-driven, The TPDO is processed on event (e.g timer)

Table 23: Transmission type setting of TPDO

TPDO 1 communication parameter

Index 1800_h

Via index 1800_h the communication parameter of TPDO 1 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Highest sub-index	5
1	Unsigned32	rw	COB-ID for PDO	180_h + Node-ID
2	Unsigned8	rw	Transmission type	FF_h
5	Unsigned16	rw	Event timer	0000_h

The object is read and write. Sub-indices 0 to 2 as well as 5 are supported. Access to other sub-indices will result in an error message. In standard setting TPDO 1 is active.

TPDO 2 communication parameter

Index 1801_h

Via index 1801_h the communication parameters of TPDO 2 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Highest sub-index	2
1	Unsigned32	rw	COB-ID for PDO	80000280_h + Node-ID
2	Unsigned8	rw	Transmission type	FF_h
5	Unsigned16	rw	Event timer	0000_h

The object is read and write. Sub-indices 0 to 2 as well as 5 are supported. Access to other sub-indices will result in an error message. In standard setting TPDO 2 is inactive.

TPDO 3 communication parameter

Index 1802_h

Via index 1802_h the communication parameters of TPDO 3 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Highest sub-index	5
1	Unsigned32	rw	COB-ID for PDO	80000400_h + Node-ID
2	Unsigned8	rw	Transmission type	FF_h
5	Unsigned16	rw	Event timer	0000_h

The object is read and write. Sub-indices 0 to 2 as well as 5 are supported. Access to other sub-indices will result in an error message. In standard setting TPDO 3 is inactive.

TPDO 4 communication parameter

Index 1803_h

Via index 1803_h the communication parameters of TPDO 4 are set.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	ro	Highest sub-index	5
1	Unsigned32	rw	COB-ID for PDO	80000480_h + Node-ID
2	Unsigned8	rw	Transmission type	FF_h
5	Unsigned16	rw	Event timer	0000_h

The object is read and write. Sub-indices 0 to 2 as well as 5 are supported. Access to other sub-indices will result in an error message. In standard setting TPDO 4 is inactive.

11.7.2 TPDO mapping parameter

The user may perform an individual mapping of the data for each TPDO. The µCAN.8.dio-SNAP module supports various objects which can be mapped into the TPDOs.

On the one hand, there are the process values which are used to read the digital inputs, and on the other hand the network variables form markers which are used to design the TPDO.

The following objects for the process value may be mapped into the TPDOs:

Name	Channel	Mapping entry
Read input 8-Bit	1	`6000 01 08h`

Table 24: Supported objects for TPDO mapping of process values

In addition to the mapping entries for the process values, the user can also use entries for the network variables. These may be used to "move" the process values within a PDO or to form "gaps" between two process values.

For this reason, network variable can have a length of 1 byte, 2 byte and 4 byte. Each network variable may repeatedly be used for mapping.

The contents of the network variables are not analyzed in the µCAN.8.dio-SNAP module.

The following objects for the network variables may be mapped into the receive-PDOs:

Name	Variable	Mapping entry
Output network variable - Unsigned8	1	A4C0 01 08h
Output network variable - Unsigned8	2	A4C0 02 08h
Output network variable - Unsigned8	3	A4C0 03 08h
Output network variable - Unsigned8	4	A4C0 04 08h
Output network variable - Unsigned8	5	A4C0 05 08h
Output network variable - Unsigned8	6	A4C0 06 08h
Output network variable - Unsigned8	7	A4C0 07 08h
Output network variable - Unsigned8	8	A4C0 08 08h
Output network variable - Unsigned16	1	A580 01 10h
Output network variable - Unsigned16	2	A580 02 10h
Output network variable - Unsigned16	3	A580 03 10h
Output network variable - Unsigned16	4	A580 04 10h
Output network variable - Unsigned32	1	A680 01 20h
Output network variable - Unsigned32	2	A680 02 20h

Table 25: Supported objects for TPFO mapping of network variables

All the objects listed above can be mapped into each of the four TPDOs-. For each TPDO between 0 and 8 mapping entries may be configured. Please note that the total number of entries within a PDO must not exceed 8 byte.

Each mapping entry consists of *index*, *sub-index* and *bit-length*, the respective values are listed in Table 20 and Table 21.

TPDO 1 mapping parameter

Index $1A00_h$

Via index $1A00_h$ the mapping parameters of TPDO 1 can be configured.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of mapped objects	1
1	Unsigned32	rw	1st mapped object	6200 0108h
2	Unsigned32	rw	2nd mapped object	-
3	Unsigned32	rw	3rd mapped object	-
4	Unsigned32	rw	4th mapped object	-
5	Unsigned32	rw	5th mapped object	-
4	Unsigned32	rw	6th mapped object	-
4	Unsigned32	rw	7th mapped object	-
4	Unsigned32	rw	8th mapped object	-

The object is read and write. Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

In standard setting the TPDO 1 is *active* and contains *one* mapping entry.

Table 24 and Table 25 show all supported entries of the µCAN.8.dio-SNAP module.

TPDO 2 mapping parameter

Index $1A01_h$

Via index 1A01h the mapping parameters of TPDO 2 can be configured.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of mapped objects	0
1	Unsigned32	rw	1st mapped object	-
2	Unsigned32	rw	2nd mapped object	-
3	Unsigned32	rw	3rd mapped object	-
4	Unsigned32	rw	4th mapped object	-
5	Unsigned32	rw	5th mapped object	-
4	Unsigned32	rw	6th mapped object	-
4	Unsigned32	rw	7th mapped object	-
4	Unsigned32	rw	8th mapped object	-

The object is read and write. Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

In standard setting the TPDO 2 is *inactive* and contains *no* mapping entry.

Table 24 and Table 25 show all supported entries of the µCAN.8.dio-SNAP module.

TPDO 3 mapping parameter

Index 1A02h

Via index 1A02h the mapping parameters of TPDO 3 can be configured.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of mapped objects	0
1	Unsigned32	rw	1^{st} mapped object	-
2	Unsigned32	rw	2^{nd} mapped object	-
3	Unsigned32	rw	3^{rd} mapped object	-
4	Unsigned32	rw	4^{th} mapped object	-
5	Unsigned32	rw	5^{th} mapped object	-
4	Unsigned32	rw	6^{th} mapped object	-
4	Unsigned32	rw	7^{th} mapped object	-
4	Unsigned32	rw	8^{th} mapped object	-

The object is read and write. Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

In standard setting the TPDO 3 is *inactive* and contains *no* mapping entries.

Table 24 and Table 25 show all supported entries of the µCAN.8.dio-SNAP module.

TPDO 4 mapping parameter

Index 1A03h

Via index 1A03h the mapping parameters of TPDO 4 can be configured.

Sub-index	Data type	Access	Name	Default value
0	Unsigned8	rw	Number of mapped objects	0
1	Unsigned32	rw	1^{st} mapped object	-
2	Unsigned32	rw	2^{nd} mapped object	-
3	Unsigned32	rw	3^{rd} mapped object	-
4	Unsigned32	rw	4^{th} mapped object	-
5	Unsigned32	rw	5^{th} mapped object	-
4	Unsigned32	rw	6^{th} mapped object	-
4	Unsigned32	rw	7^{th} mapped object	-
4	Unsigned32	rw	8^{th} mapped object	-

The object is read and write. Sub-indices 0 to 8 are supported. Access to other sub-indices will result in an error message.

In standard setting the TPDO 4 is *inactive* and contains *no* mapping entries.

Table 24 and Table 25 show all supported entries of the µCAN.8.dio-SNAP module.

11.8 Synchronization Service

On reception of a SYNC message the data transfer of a TPDO may be triggered. Please note that the transmission type for the TPDO communication parameter must be configured according to Table 23 on page 119.

COB-ID SYNC

Index 1005_h

The object at index 1005_h defines the identifier for the SYNC message.

Sub-index	Data type	Access	Name	Default value
0	Unsigned32	rw	*COB-ID SYNC*	80h

The object supports read and write acces. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

Sender **µCAN.8.dio-SNAP**

		ID [hex]	DLC	Data [hex]	Comment
1	->	67F	8	22 05 10 01 **0A** 00 00 00	
2	<-	5FF	8	60 05 10 01 00 00 00 00	

Sequence 35: Set the COB-ID SYNC to 180_h for a module with node-ID 127

The default value of the SYNC identifier is 80_h which will ensure high priority of the SYNC messages on the CAN bus.

The value of the COB_ID SYNC is not automatically stored in non-volatile memory, Store parameters has to be triggered for that purpose.

Communication cycle period

Index 1006_h

Via index 1006_h the cycle time for a SYNC producer can be configured.

Sub-index	Data type	Access	Name	Default value
0	Unsigned32	rw	Communication cycle period	0000 0000h

The object supports read and write acces. Only sub-index 0 is supported. Access to other sub-indices will result in an error message.

12. Technical Data

Power Supply	
Supply Voltage	9 .. 36 V DC, reverse polarity protected
Power consumption	1.5 W (60 mA @ 24 V DC) no load
Connection	Screw terminals at the COMBICON plug.

CAN bus	
Bit rates	50 kBit/s .. 1 MBit/s
State of Bus	active node
Protocol	CANopen/CANopen FD according to CiA 301 V4.02, CiA 1301 and CiA 401
Connection	Screw terminals at the COMBICON plugs.

EMC	
Electromagnetic immunity	according to EN 50082-2
Electrostatic discharge	8 kV air discharge, 4 kV contact discharge, according to EN 61000-4-2
Electromagnetic fields	10 V/m, according to ENV 50204
Burst	5 kHz, 2 kV according to EN 6100-4-4
Conducted RF disturbance	10 V, according to EN 61000-4-6
Electromagnetic emission	According to EN 50081-2, requirements according to EN 55022, class A

Digital inputs	
Type	0..36 V
Response time	< 1 ms

Digital outputs	
Type	9..36 V
max. Current	1 A per channel / 6 A total current

Casing	
Plastics	Polyamide
Temperature Resistance	-40°C to +105°C
Combustibility Class	V0 (according to UL 94)
Mounting	On supporting rail TS35 according to DIN EN 50022
Dimensions	128.8 * 22.5 * 102 mm (D * W * H)
Weight	approx. 150 g
Protection Class	IP20

Digital outputs	
Type	9..36 V
max. Current	1 A per channel / 6 A total current

Index

P

R

S

T

U

MicroControl shall neither assume liability for compliance of the content of this manual with the respective legal provisions nor for errors and technical specifications.

MicroControl
Systemhaus für Automatisierung

MicroControl GmbH & Co. KG
Junkersring 23
D-53844 Troisdorf
Fon: +49 / 2241 / 25 65 9 - 0

https://www.microcontrol.net