

E	5.01024.01
---	------------

**BAUMÜLLER**

**Ωmega CANopen**

**Function block**

**Add-on for**

**Option Board CAN-M-01**

Technical Description  
and Operating Instructions

Edition: July 2001



# BAUMÜLLER

## ΩMEGA CANOPEN FUNCTION BLOCK ADD-ON FOR OPTION BOARD CAN-M-01

### Technical Description and Operating Instructions

**Edition: July, 1st 2001**

**Document no. 5.01024.01**

This operation manual is intended as a complement to the technical description and the operation manual of the apparatus.

**BEFORE CARRYING OUT COMMISSIONING, CAREFULLY  
READ AND OBSERVE THE OPERATING INSTRUCTIONS  
AND SAFETY INFORMATION**

This document contains all the information necessary to correctly use the products it describes. It is intended for specially trained, technically qualified personnel who are well-versed in all warnings and commissioning activities.

The equipment is manufactured using state-of-the-art technology and is safe in operation. It can safely be installed and commissioned and functions without problems if the safety information is followed.

You may not carry out commissioning until it has been established that the machine into which this component is to be installed complies with the specifications of the EC machine guidelines.

This technical description/these operating instructions invalidate all previous descriptions of the corresponding product. Within the scope of further development of our products, Baumüller GmbH reserves the right to change their technical data and handling.

**Manufacturer and  
Supplier's Address:** Baumüller Nürnberg GmbH  
Ostendstr. 80  
D-90482 Nürnberg  
Tel. ++49 (0)9 11/54 32 - 0  
Fax ++49 (0)9 11/54 32 - 1 30

**Copyright:** These operating instructions or extracts from them may not be copied or duplicated without our permission.

**Country of Origin:** Germany

**Date of Manufacture:** Determined from the serial number on the equipment



---

## TABLE OF CONTENTS

<b>1</b>	<b>Safety Information .....</b>	<b>5</b>
<b>2</b>	<b>Technical data .....</b>	<b>7</b>
2.1	General .....	7
2.2	Technical data, option board CAN-M-01 .....	8
<b>3</b>	<b>Installation .....</b>	<b>9</b>
3.1	Sample configuration .....	9
3.2	Pin assignment .....	10
3.3	Dip switch .....	11
3.4	Connection cables .....	11
3.5	Accessories .....	12
3.6	Wiring .....	13
<b>4</b>	<b>CANopen node .....</b>	<b>15</b>
<b>5</b>	<b>CANopen communication .....</b>	<b>17</b>
5.1	Characteristics .....	17
5.2	Programming under PROPROG wt II .....	18
5.3	Special functions .....	21
<b>6</b>	<b>Function blocks for CANopen .....</b>	<b>23</b>
6.1	Overview .....	23
6.2	CANop405_COB_ID .....	24
6.3	CANop405_EMERGENCY .....	26
6.4	CANop405_INIT .....	29
6.5	CANop405_NMT .....	32
6.6	CANop405_NODE_GUARDING .....	34
6.7	CANop405_PDO_READ .....	37
6.8	CANop405_PDO_WRITE .....	40
6.9	CANop405_SDOx_READ .....	45
6.10	CANop405_SDOx_WRITE .....	49
6.11	CANop405_SYNC .....	53
<b>7</b>	<b>Index .....</b>	<b>55</b>



## 1 SAFETY INFORMATION

### General Information

These operating instructions contain all the information necessary for correct operation of the products described. The document is intended for specially trained, technically qualified personnel who are well-versed in all warnings and commissioning activities.

The equipment is manufactured using state-of-the-art technology and is safe in operation. It can safely be installed and commissioned and functions without problems if the safety information in these operating instructions is followed.

### Danger Information

On the one hand, the information below is for your own personal safety and on the other to prevent damage to the described products or to other connected equipment.

In the context of the operating instructions and the information on the products themselves, the terms used have the following meanings:



#### DANGER

This means that **death, severe personal injury, or damage to property will** occur unless appropriate safety measures are taken.



#### WARNING

This means that **death, severe personal injury, or damage to property may** occur unless appropriate safety measures are taken.



#### NOTE

This draws your attention to **important information** about the product, handling of the product or to a particular section of the documentation.

## Qualified Personnel

In the context of the safety-specific information in this document or on the products themselves, qualified personnel are considered to be persons who are familiar with setting up, assembling, commissioning and operating the product and who have qualifications appropriate to their activities:

- Trained or instructed or authorized to commission, ground and mark circuits and equipment in accordance with recognized safety standards.
- Trained or instructed in accordance with recognized safety standards in the care and use of appropriate safety equipment.

## Appropriate Use



### WARNING

You may only use the equipment/system for the purposes specified in the operating instructions and in conjunction with the third-party equipment and components recommended or authorized by BAUMÜLLER NÜRNBERG GmbH.

For safety reasons, you must not change or add components on/to the equipment/system.

The machine minder must report immediately any changes that occur which adversely affect the safety of the equipment/system.



## 2 TECHNICAL DATA

### 2.1 General


Utilisation of the CANopen function block add-on requires the option board CAN-M-01 (CAN master) with software function 1 <sup>1)</sup> for the **Omega Drive-Line II** as the CAN nodes integrated in the **Omega Drive-Line II** are to be used for CANsync interfacing.

The option board CAN-M-01 allows communication with up to 32 network nodes with CANopen communication profile.

---

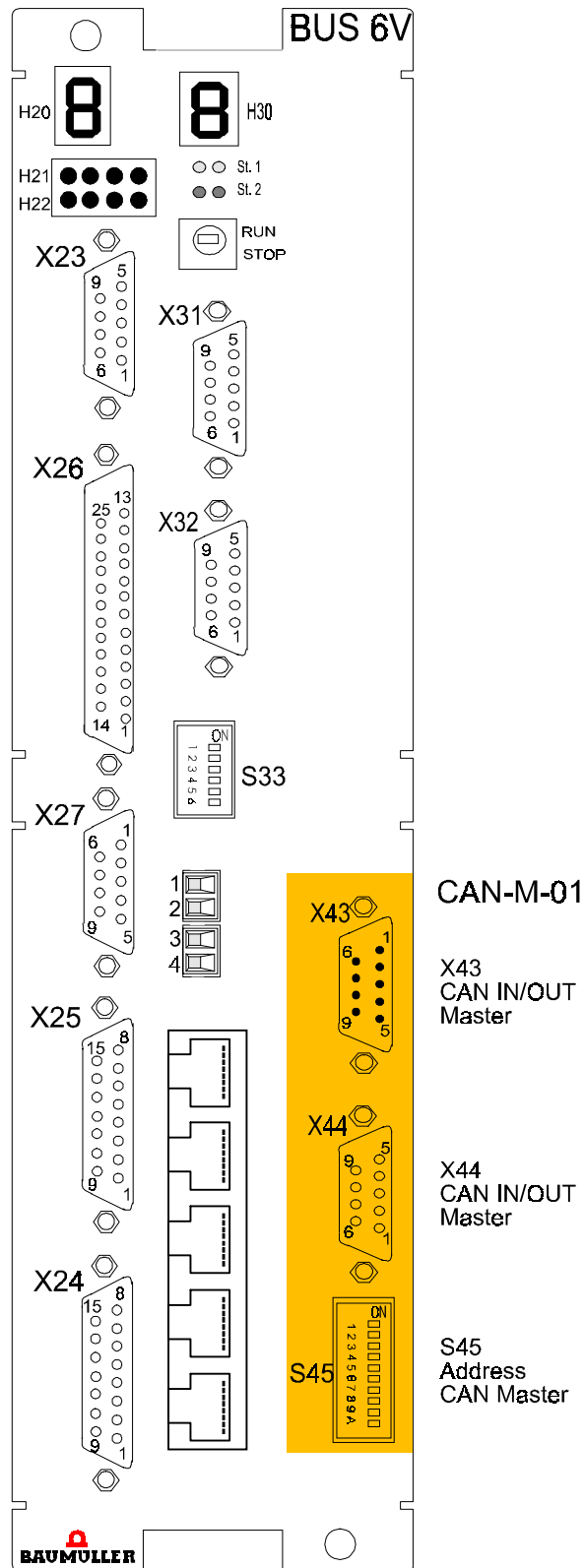
<sup>1)</sup> 01xx in the last four digits of the type key.

### 2.2 Technical data, option board CAN-M-01

CPU	8-Bit CPU 16 MHz
Operating voltage	+5 V
Power consumption	max. 1A
Ambient temperature	0 ... 55°C
Storage temperature	-15 ... 70°C
Humidity in air	max. 90%
Memory	32 kByte RAM, 64 kByte EPROM
Linkup with  mega Drive-Line II	Dual Port Ram 2k x 16
CAN controller	SJA1000T
Physical Layer	ISO 11898
Baud rate	125 kBit/s, 250 kBit/s
Galvanic isolation	Optocouplers, DC/DC converter
CAN bus connector	SUB-D plug and female connector 9pole

### 3 INSTALLATION

#### 3.1 Sample configuration



Omega Drive-Line II with CAN-M-01 in option slot 2



## NOTE

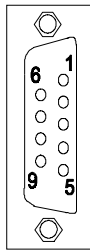
The option board CAN-M-01 with software function 1 must be used.

### 3.2 Pin assignment

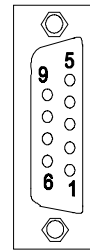
- CANopen-IN / -OUT

**X 43 SUB-D plug connector**

**X 44 SUB-D female connector**



X 43



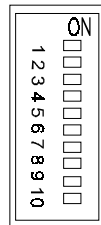
X 44

Pin no.	Assignment
1	Reserved
2	CAN LOW bus line (dominant low)
3	GND Ground
4	Reserved
5	Reserved
6	Reserved
7	CAN HIGH bus line (dominant high)
8	Reserved
9	Reserved

### 3.3 Dip switch

- Address, CANopen master

S45



Reserved



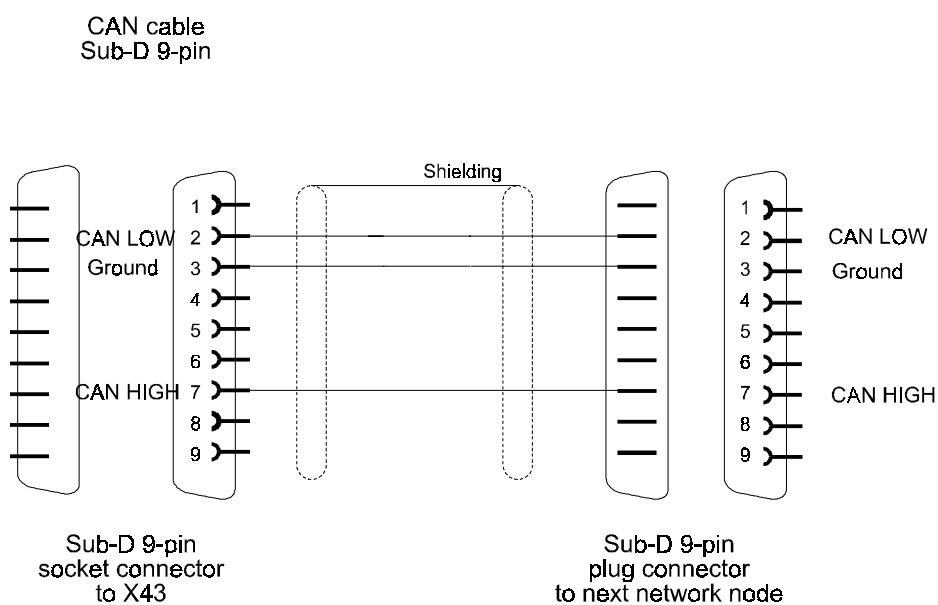
#### NOTE

In the case of the CAN-M-01 option board with software function 1, the switch position is of no significance for data exchange with CANopen communication profile, but may be read out by the application program.

### 3.4 Connection cables

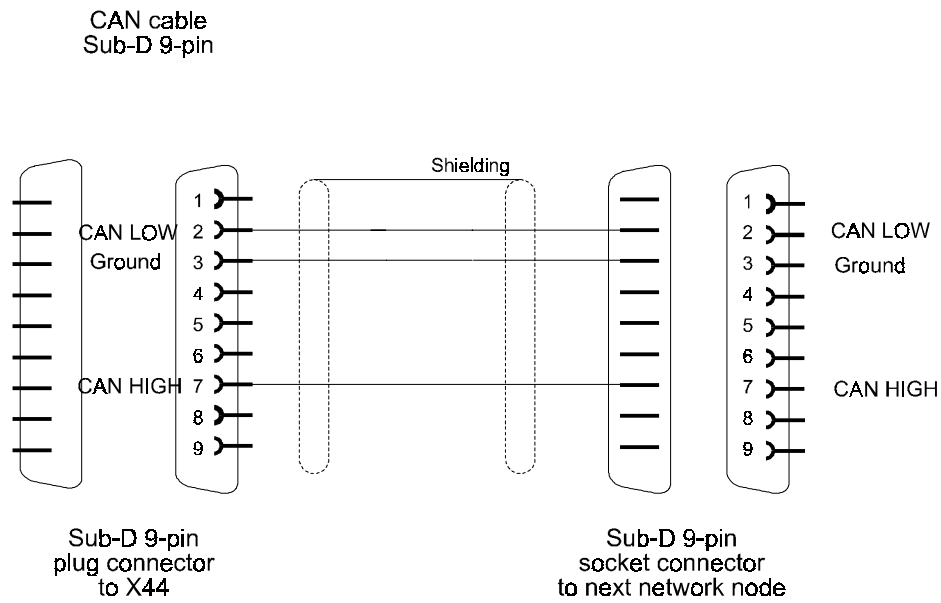
#### Connection cables for further CANopen network nodes

- 9pole connection for X43



# Installation

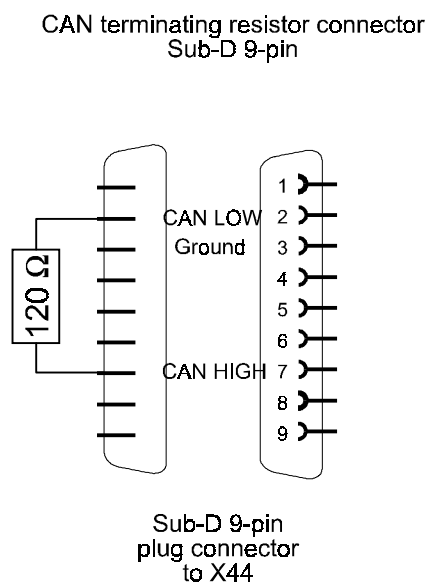
- 9pole connection for X44



## 3.5 Accessories

### Terminating resistor connectors 120 $\Omega$

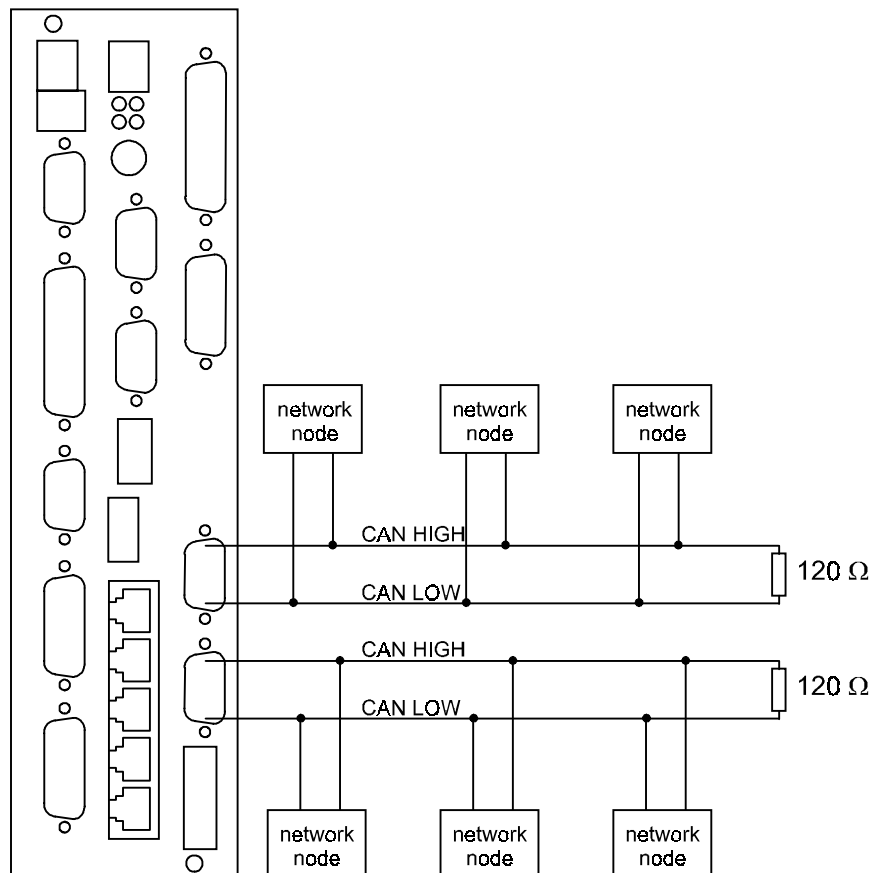
- 9pole connection



### 3.6 Wiring

- All network nodes must be connected parallel with each other.
- Avoid branches.
- Both ends of the CANopen bus must be terminated with terminating resistor connectors ( $120\ \Omega$ ).
- Observe the maximum bus length for the selected Baud rate: 300 m at 250 kBit/s and 600 m at 125 kBit/s.

Block diagram:







## **4 CANOPEN NODE**

The Basic-CAN controller SJA1000T is used as a controller. The bus linkup is as per ISO 11898 (CAN High-Speed). The bus linkup is isolated via high-speed optocouplers and supplied by an internal DC/DC converter.

### **Features**

- Communication with up to 32 network nodes with CANopen communication profile
- Simultaneous transmission of up to 20 process data objects (PDO)
- Reception of up to 40 different process data objects (PDO)
- Simultaneous processing (transmitting or receiving) of up to 8 service data objects (SDO)

### **CANopen characteristics**

- Serial asynchronous bus system
- Real time capability (max. 250 kBit/s at 300 m bus length)
- Broadcast / multicast and point-to-point communication
- Powerful error detection and handling
- High reliability (Hamming distance = 6)
- Priority-controlled bus allocation
- Guaranteed maximum latency for high-priority messages
- Open system
- International standard



## 5 CANOPEN COMMUNICATION

### 5.1 Characteristics

CANopen is an open communication profile based on the *Controller Area Network (CAN)* bus system and published as DS-301 profile by the international CAN organisation *CAN in Automation e.V. (CiA)*. CANopen uses a subset of the communication objects offered on the CAL application layer for CAN networks. In a CAN network, an object is clearly identified via a COB identifier (communication object identifier). COB identifiers have been predefined to enable the setup of peer-to-peer communication between master and network nodes directly after a bootup. The COB identifier consists of the 'function code' defining the object priority, and the node ID, the node number within the network. This identifier allocation may be reconfigured for some objects.

The DS-301 profile specifies two object types for data exchange as well as several special objects. The following object types are supported with the option board CAN-M-01 and the associated library CANop405\_DLII\_20bd01 (and higher) for PROPROG wt II:

- Process data objects (PDO): Real-time exchange with high-priority identifiers (function codes 0011 to 1010) and up to 8 bytes per message. A maximum of 20 PDOs can be written simultaneously and 40 PDOs be read at the same time.
- Service data objects (SDO): Parameter data exchange with low-priority identifiers (function codes 1011 and 1100) and data that can be addressed via index/subindex. The transfer type is 'expedited', i. e. up to 4 bytes per message may be transmitted. Depending on the network configuration, up to 8 different SDOs may be read or written at the same time.
- Network management (NMT): Special object type (function code 0000) for the implementation of network communication functions such as e. g. initialisation, start and reset of network nodes. The **Omega Drive-Line II** as CANopen master transmits the commands as broadcast to the respective network nodes.
- Synchronisation (SYNC): Special object type (function code 0001) for the synchronisation of real-time data exchange with PDOs. The **Omega Drive-Line II** as CANopen master transmits the SYNC command as broadcast to the respective network nodes.
- Error handling (EMERGENCY): Special object type (function code 0001) for the detection of errors in a network node. If an error occurs in a network node, and the network node sends an emergency telegram, this telegram may be received and evaluated by the **Omega Drive-Line II**.
- Node guarding (NODE GUARDING): Special object type (function code 1110) for network node guarding. The **Omega Drive-Line II** as CANopen master can cyclically request network node feedback via a telegram to detect network node failures.

The **Omega Drive-Line II** can communicate with up to 32 CANopen network nodes (e. g. I/O modules). For the exact COB identifier allocation, object dictionary structure and performed network management functions refer to the manufacturer-specific descriptions of the network nodes. Note that data exchange is from the point of view of the network nodes; for example, the **Omega Drive-Line II** writes receive PDOs (RxPDOs) and reads transmit PDOs (TxPDOs).

CiA published the 'Draft Standard Proposal 405' for the implementation of CANopen communication on IEC 61131-3 programmable systems. This proposal makes suggestions as to the structure of data types and function blocks for IEC 61131-3 programming environments. These suggestions were taken into account for the **Omega Drive-Line II**, producing the block library CANop405\_DLII\_20bd01 for the PROPROG wt II programming platform as the CANopen function block add-on for option board CAN-M-01. This allows an application on the **Omega Drive-Line II** to be easily integrated into a CANopen network as a master for I/O modules.

## 5.2 Programming under PROPROG wt II

### Overview

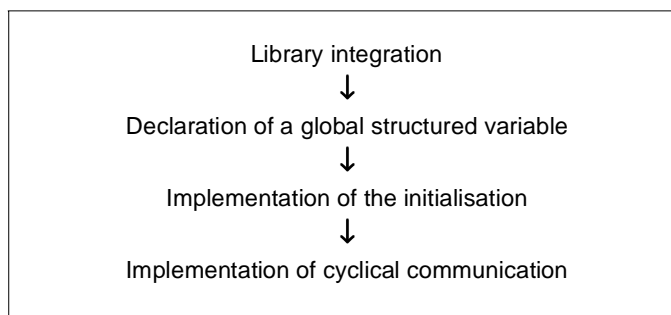
The programming system PROPROG wt II is available for programming the **Omega** Drive-Line II under IEC 61131-3. The generation of projects with PROPROG wt II as well as configuration options and functions of the **Omega** Drive-Line II are described in detail in the PROPROG wt II manual and the technical description of the **Omega** Drive-Line II and enhanced there with sample applications. Refer to these two descriptions for more detailed information on these topics.

The library CANop405\_DLII\_20bd01 (or higher) is available to implement data exchange with CANopen communication profile under PROPROG wt II. The individual blocks are used as follows:

Block	Function	Task type
CANop405_COB_ID	Computes the COB identifier from the function code of the process data object (PDO) and the selected node number	Cold and warm starts Cyclical / event task
CANop405_INIT	Initialises option board CAN-M-01 for a CANopen interface as the master	Cold and warm starts
CANop405_NMT	Network management functions (initialisation, start, etc.)	Cyclical task
CANop405_SYNC	Synchronises the real-time data exchange (PDO)	Cyclical / event task
CANop405_EMERGENCY	Receives and evaluates emergency telegrams	Cyclical task
CANop405_NODE_GUARDING	Node guarding	Cyclical task
CANop405_SDOx_READ	Reads service data objects	Cyclical task
CANop405_SDOx_WRITE	Writes service data objects	Cyclical task
CANop405_PDO_READ	Reads process data objects	Cyclical / event task
CANop405_PDO_WRITE	Writes process data objects	Cyclical / event task

For SDOs, there are 8 function blocks each available for reading and writing (SDOx = SDO1 to SDO8).

Implementation of CANopen communication into an application happens in four stages:



The individual stages are briefly described below; for details refer to the relevant functional block documentation.

## Library integration

The library BM\_TYPES\_20bd01 (or higher) must be integrated for the setup of CANopen communication under PROPROG wt II. This library provides important data types for communication setup and operation. The library CANop405\_DLII\_20bd01 (or higher) with the function blocks for initialisation and communication is also required.

## Declaration of a global structured variable

The function blocks of the library CANop405\_DLII\_20bd01 exchange data with the option board CAN-M-01 via a dual-port RAM. A CANop405\_CTRL\_BMSTRUCT data-type global variable with a specific basic address must be generated for this purpose. This data type allows efficient access to the dual-port RAM within the function blocks. The basic address must be %MB3.3000000 independent of the option board CAN-M-01 slot.

Declaration in a global variable work sheet:

```
globVar      AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT;
```

where:

*globVar*                                    the name of the variable to be declared

CANop405\_CTRL\_BMSTRUCT    the data type

%MB3.3000000                            the basic address of option board CAN-M-01

The same global variable must be linked to the same input of all function blocks from the library CANop405\_DLII\_20bd01. Structure and contents of this global variable are of no further significance for the general case of application. Chapter 5.3 on Page 21 describes individual structural elements in further detail to assist in the application of special functions.

## Implementation of the initialisation

Generate a program POE in which the option board CAN-M-01 is initialised for data exchange with CANopen communication profile. This program POE must be requested in a cold and in a warm start task. Its contents consist of the following sections:

- Allocation of COB identifiers for PDOs to be received by assigning array elements.
- The function block CANop405\_COB\_ID may be used to compute the COB identifier.
- Implementation and configuration of the function block CANop405\_INIT (setting the Baud rate for the CAN bus and a monitoring period).

## Implementation of cyclical communication

Several function blocks are available for cyclical communication:

The function block CANop405\_NMT can be used to control the network node statuses. The following status transitions are defined as per *CiA Draft Standard 301*: 'Start remote node', 'Stop remote node', 'Enter pre-operational', 'Reset node' and 'Reset communication'. In general, the network nodes are automatically in 'pre-operational' status after power-on and self-initialisation, so that this function block only needs to send the command 'Start remote node'. This function block must be instantiated once only. Implementation would be appropriate in a program POE associated with a cyclical task.

The function block CANop405\_EMERGENCY is used to receive an emergency telegram from a network node. The emergency telegram is evaluated and its contents displayed: A general error specification via an error code and an error register as well as a manufacturer-specific error message. One function block must be used per network node.

The function block CANop405\_NODE\_GUARDING can detect a network node failure. The function block requests a guarding telegram from the network node at specific time intervals (node guarding time). In this guarding telegram, the network node transmits a control bit (toggle bit) among other information. The function block displays an error if the control bit is faulty or the guarding telegram is not received. One function block must be used per network node.

SDOs are read with the function blocks CANop405\_SDO1\_READ to CANop405\_SDO8\_READ and written with the function blocks CANop405\_SDO1\_WRITE to CANop405\_SDO8\_WRITE, so that SDO commands may be started at up to 8 network nodes simultaneously. Reading from or writing to network node object dictionaries is via the specification of node numbers, indices and data width. Implementation of the function blocks would be appropriate in a program POE associated with a cyclical task.

PDOs are read with the function block CANop405\_PDO\_READ and written with the function block CANop405\_PDO\_WRITE. Up to 20 PDOs can be written simultaneously and up to 40 PDOs be read at the same time. The COB identifiers for the PDOs to be written may be changed at runtime. For PDOs to be read, the COB identifier must already be specified during the initialisation stage. In general, network nodes have a default assignment ('default mapping') for the first two PDOs. For I/O modules, this may be as follows:

TxPDO1 for digital inputs (function code 0011)

RxPDO1 for digital outputs (function code 0100)

TxPDO2 for analog inputs (function code 0101)

RxPDO2 for analog outputs (function code 0110)

Any assignments not used by the I/O module (e. g. no digital output terminals) remain spare. The function blocks may be used in program POEs assigned to a cyclical or an event task.

The function block CANop405\_SYNC may be used to synchronise real-time data exchange with PDOs. The function block sends a SYNC telegram. The recipient can synchronise with this telegram, if set accordingly, and write actual values (TxPDOs) and/or accept setpoints (RxPDOs). As a rule, the setpoints are valid only after receipt of the next SYNC telegram. The function block should be used in the same task as the PDOs to be synchronised.

## 5.3 Special functions

Declaration of the global structured variable with the basic address `%MB3 . 3000000` (refer 'Programming under PROPROG wt II' on page 18) allows readout of internal data of the option board CAN-M-01. This process is detailed below, with '\*' standing for the actually declared global variable.

### Checking the software version of the option board CAN-M-01

The version of the CAN software of the option board CAN-M-01 may be read out via the structural element `*.i_SW1_NR`. For data exchange with CANopen communication profile, this must be `*.i_SW1_NR = 1195`.

### Checking the software release of the option board CAN-M-01

Both the incompatible and the compatible release of the CAN software of the option board CAN-M-01 may be read out via the structural element `*.i_SW1_RELEASE`. For data exchange with CANopen communication profile, this must be `*.i_SW1_RELEASE = 201` (or higher).

### Reading out the DIP switches

The DIP switches (S45) may be read out via the structural element `*.w_OMEGA_NR`. These DIP switches may be freely used for application purposes.





## 6 FUNCTION BLOCKS FOR CANOPEN

### 6.1 Overview


If you have integrated manufacturer-defined libraries in a project, you may use manufacturer-defined functions in addition to the standard functions.



#### NOTE

Refer general help for how to integrate libraries.

The following function blocks are available for CANopen:

Function	Brief description
CANop405_COB_ID	Computes COB identifier to initialise function block CANop405_INIT and function block CANop405_PDO_WRITE
CANop405_EMERGENCY	Receives emergency telegrams from a network node
CANop405_INIT	Initialises an option board CAN-M-01 with firmware function 1 at the  mega Drive-Line II for CANopen communication
CANop405_NMT	Executes network management functions during CANopen communication
CANop405_NODE_GUARDING	Guards the network nodes
CANop405_PDO_READ	Receives a process data object (PDO) from a network node
CANop405_PDO_WRITE	Writes a process data object (PDO) to a network node
CANop405_SDO1_READ	Read a service data object (SDO) from a network node
to CANop405_SDO8_READ	
CANop405_SDO1_WRITE	Write a service data object (SDO) to a network node
to CANop405_SDO8_WRITE	
CANop405_SYNC	Transmits SYNC object

## 6.2 CANop405\_COB\_ID

### Description

You may use this function block for CANop405 to compute the COB identifier for CANop405 initialisation and function block CANop405\_PDO\_WRITE. The COB identifier consists of the function code of the process data object (PDO) and the selected node number.

Parameter input	Data type	Description
us_DEVICE	USINT 1 to 32	Node number (node ID)
us_PDO_NR	USINT 1 to 4	Number of the process data object
x_RX	BOOL	TRUE = PDO (rx) FALSE = PDO (tx)

Parameter output	Data type	Description
x_ERROR	BOOL	Error bit
w_PDO_COB_ID	WORD	COB identifier

The COB identifier consists of the function code of the process data object and the selected node number. The function block CANop405\_COB\_ID generates the COB identifier as per the *Predefined Connection Set* of the *Cia Draft Standard 301*. An impermissible node number or impermissible process data object number sets the output x\_ERROR to TRUE and w\_PDO\_COB\_ID to 0.

Input us\_DEVICE:


us\_DEVICE specifies the node number of the network node, supporting node numbers from 1 to 32.

Input us\_PDO\_NR:

Number of the process data object (PDO), supporting numbers 1 to 4.

Input x\_RX:

Identifies whether the PDO is received or transmitted by the network node.

x\_RX = TRUE      PDO is received by the network node and transmitted by the mega Drive-Line II.

x\_RX = FALSE     PDO is transmitted by the network node and received by the mega Drive-Line II.

Output x\_ERROR:

Errors are signalled with x\_ERROR = TRUE. An error means that impermissible values were specified for us\_DEVICE or us\_PDO\_NR.

Output w\_PDO\_COB\_ID:

The computed COB identifier may be specified in the cyclical program at function block CANop405\_PDO\_WRITE (x\_RX = TRUE).

When the CANop405 is initialised with function block CANop405\_INIT, the entries may be made with this starting value in the array at input a\_PDO\_COB\_ID.



### NOTE

For runtime optimisation, the function block CANop405\_COB\_ID should be requested only if required, e. g. during initialisation.

## 6.3 CANop405\_EMERGENCY

### Description

You may use this function block for CANop405 to receive emergency telegrams from a network node during data exchange with CANopen communication profile.



### NOTE

The function block CANop405\_EMERGENCY uses the library BM\_TYPES\_20bd01 or higher.

Parameter input	Data type	Description
x_RESET	BOOL	Reset
us_DEVICE	USINT 1 to 32	Node number (node ID)
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
x_EMERGENCY	BOOL	Signals an emergency
w_EMCY_ERROR_CODE	WORD	Error code
b_ERROR_REGISTER	BYTE	Error register
a_ERROR_FIELD	BYTE_8_BMARRAY	Manufacturer-specific error information
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

As soon as an error occurs in a network node, the network node transmits an emergency telegram (refer documentation on the respective network node). The function block CANop405\_EMERGENCY is used to receive emergency telegrams from a network node. The number of the network node from which emergency telegrams are to be received, is specified at us\_DEVICE. Once an emergency telegram has been received, the received error code, the error registers and the manufacturer-specific error information is output. If the received error code is  $\neq 0$ , the output x\_EMERGENCY is set to TRUE. Any further error code 0 emergency telegram resetting the error messages sets the output x\_EMERGENCY to FALSE.

Input x\_RESET:

x\_RESET = TRUE resets outputs x\_EMERGENCY, w\_EMCY\_ERROR\_CODE and b\_ERROR\_REGISTER to 0.

Input: us\_DEVICE

This input specifies the node number of the network node from which emergency telegrams are to be received, supporting nodes from 1 to 32.

Input/output `_CANop405_CTRL`:

A `CANop405_CTRL_BMSTRUCT` data-type global variable must be linked to `_CANop405_CTRL`, that maps the operating data for the CAN interface. This variable must be linked to the basic address of the CAN interface via the global variable declaration.

Example:

Option board CAN-M-01 for **Omega Drive-Line II**

```
_CANop405Base AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT;
```

where:

<code>_CANop405Base</code>	the name of the variable with data type identifier '_' for Struct
<code>CANop405_CTRL_BMSTRUCT</code>	the data type
<code>%MB3.3000000</code>	the basic address of the CAN interface on the option board CAN-M-01

Output `x_EMERGENCY`:

If the error code received in the emergency telegram is  $>0$ , the output `x_EMERGENCY` is set to TRUE. Outputs `w_EMCY_ERROR_CODE`, `b_ERROR_REGISTER` and `a_ERROR_FIELD` further describe the error message.

A further error code 0 emergency telegram from the network node resetting the error messages sets output `x_EMERGENCY` to FALSE and outputs `w_EMCY_ERROR_CODE`, `b_ERROR_REGISTER` and `a_ERROR_FIELD` to 0.

An impermissible value was specified for `us_DEVICE` if `x_EMERGENCY` = TRUE and the error outputs `w_EMCY_ERROR_CODE` and `b_ERROR_REGISTER` are 0. In this case, no emergency telegram monitoring is possible.

Output `w_EMCY_ERROR_CODE`:

`w_EMCY_ERROR_CODE` corresponds to the *Error Code of Draft Standard 301*.

The error code last received is output.

Error code	Description
16#00xx	Error Reset or No Error
16#10xx	Generic Error
16#20xx	Current
16#21xx	Current, device input side
16#22xx	Current inside the device
16#23xx	Current, device output side
16#30xx	Voltage
16#31xx	Mains Voltage
16#32xx	Voltage inside the device
16#33xx	Output Voltage
16#40xx	Temperature
16#41xx	Ambient Temperature
16#42xx	Device Temperature
16#50xx	Device Hardware

## Function blocks for CANopen

---

Error code	Description
16#60xx	Device Software
16#61xx	Internal Software
16#62xx	User Software
16#63xx	Data Set
16#70xx	Additional Modules
16#80xx	Monitoring
16#81xx	Communication
16#8110	CAN Overrun (Objects lost)
16#8120	CAN in Error Passive Mode
16#8130	Life Guard Error or Heartbeat Error
16#8140	recovered from bus off
16#8150	Transmit COB-ID
16#82xx	Protocol Error
16#8210	PDO not processed due to length error
16#8220	PDO length exceeded
16#90xx	External Error
16#F0xx	Additional Functions
16#FFxx	Device specific

This error message corresponds to object 16#1003 (predefined) of the respective network node.

Output `b_ERROR_REGISTER`:

Corresponds to object 16#1001 (predefined) of the respective network node.

The error code last received is output.

Output `a_ERROR_FIELD`:

The manufacturer-specific errors are entered in the first 5 bytes (index 0 to 4) of this array. The last 3 bytes (index 5 to 7) are reserved.

For information on manufacturer-specific errors refer to the description of the respective network node.

The error field values last received are output.

## 6.4 CANop405\_INIT

### Description

You may use this function block for CANop405 to initialise an option board CAN-M-01 with firmware function 1 at the **Omega Drive-Line II** for data exchange with CANopen communication profile.



### NOTE

The function block CANop405\_INIT uses the library BM\_TYPES\_20bd01 or higher.

Parameter input	Data type	Description
us_BAUDRATE	USINT 3,4	Baud rate
t_SEND_ABORT_TIME	TIME	Transmit command abort time
a_PDO_READ_COB_ID	WORD_64_BMARRAY	COB identifiers for PDOs to be received
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
w_ERROR	WORD	Error number
x_CONFIRM	BOOL	Initialisation completed successfully
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

### General

The function block CANop405\_INIT and the other function blocks of the library CANop405\_DLII\_20bd00 may be used to implement data exchange with CANopen communication profile, supporting process data objects (PDO), service data objects (SDO) and network management functions. The **Omega Drive-Line II** is the CANopen master and can communicate with up to 32 CANopen network nodes (e. g. I/O modules). The individual blocks are used as follows:

Block	Function	Task type
CANop405_INIT	Initialises option board CAN-M-01 for a CANopen interface as the master	Cold and warm starts
CANop405_NMT	Network management functions (initialisation, start, etc.)	Cyclical task
CANop405_SDOx_READ	Reads service data objects	Cyclical task
CANop405_SDOx_WRITE	Writes service data objects	Cyclical task
CANop405_PDO_READ	Reads process data objects	Cyclical / event task
CANop405_PDO_WRITE	Writes process data objects	Cyclical / event task
CANop405_EMERGENCY	Receives emergency telegrams	Cyclical / event task
CANop405_NODE_GUARDING	Node guarding	Cyclical / event task

## Function blocks for CANopen

---

Block	Function	Task type
CANop405_SYNC	Transmits SYNC objects	Event task
CANop405_COB_ID	Computes COB identifier	Cold and warm starts, if possible

For SDOs, there are 8 function blocks each available for reading and writing,

so that a maximum of 20 PDOs can be written simultaneously and 40 PDOs be read at the same time. The respective number must be specified at the associated communication function block. Numbers must be used once only. This number is of internal significance only and not related to COB identifier, node ID or similar of the *CiA Draft Standard 301*.

### Using function block CANop405\_INIT

The function block CANop405\_INIT performs the initialisation of option board CAN-M-01 with firmware function 1 at the  $\Omega$ mega Drive-Line II for data exchange with CANopen communication profile. The function block is requested in cold and warm start tasks and can be parameterised via various input values. The required Baud rate for the CAN bus is set at **us\_BAUDRATE**. Currently, the possible rates are 125 kBit/s at **us\_BAUDRATE** = 3 or 250 kBit/s at **us\_BAUDRATE** = 4. Input values other than 3 or 4 result in an error message. A monitoring time may be specified at **t\_SEND\_ABORT\_TIME** within which transmit attempts are made. An error message is output if the monitoring time has expired and a transmission not been successfully completed. If this input is not assigned, a default setting of 9 ms at 125 kBit/s or 5 ms at 250 kBit/s transmission rate will be applied. The PDOs to be read are configured with **a\_PDO\_READ\_COB\_ID**. For this purpose, the COB identifier for up to 40 PDOs is specified in the associated index element of the array linked to **a\_PDO\_READ\_COB\_ID**.

Example:

The variable **a\_PdoReadConfig** is linked to **a\_PDO\_READ\_COB\_ID**. Three PDOs with the COB identifiers 385, 387 and 650 are to be read during the cyclical program section. A possible allocation prior to CANop405\_INIT request then looks as follows:

```
a_PdoReadConfig[1] := WORD#385;
a_PdoReadConfig[2] := WORD#387;
a_PdoReadConfig[3] := WORD#650;
```

The index values (1,2,3) must correspond to the numbers that are specified at the associated function blocks CANop405\_PDO\_READ (**us\_PDO\_NR**).

The function block ignores an index value = 0 or > 40. The composition of the COB identifier complies with the definition in the *CiA Draft Standard 301*. The COB identifier value range is monitored for a PDO range from 385 to 1407 and leads to an error message if exceeded/not reached. PDOs to be written are configured directly at the function block CANop405\_PDO\_WRITE.

The function block CANop405\_COB\_ID may also be used to compose the COB identifier.

A CANop405\_CTRL\_BMSTRUCT data-type global variable must be linked to **\_CANop405\_CTRL** and linked to the basic address `%MB3.3000000` of the CAN-M-01 via the declaration (`globVar AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT`). The same global variable must be linked to the same input of the other function blocks from the library CANop405\_DLII\_20bd00. The variable is required for data exchange with the option board CAN-M-01 and of no further significance to the user.

The function block CANop405\_INIT confirms successful initialisation with **x\_CONFIRM** = TRUE. An error number is output at **w\_ERROR**. The error numbers comply with the definition of CIA405\_CANOPEN\_KERNEL\_ERROR of the *CiA Draft Standard Proposal 405*.



<b>w_ERROR</b>	<b>Definition</b>
16#0000	No error
16#0001 - 16#0061	Not used
16#0062	Timeout on CAN-M-01 handshake
16#0063	Input t_SEND_ABORT_TIME outside permissible range ( > 255 ms)
16#0064	Wrong Baud rate
16#0065	PDO COB identifier outside permissible range
16#0066	Timeout on function block execution
16#0062 - 16#00FF	Not used

Any error causes **x\_CONFIRM** = FALSE.

## 6.5 CANop405\_NMT

### Description

You may use this function block for CANop405 to execute network management functions during data exchange with CANopen communication profile.



### NOTE

The function block CANop405\_NMT uses the library BM\_TYPES\_20bd01 or higher.

Parameter input	Data type	Description
us_DEVICE	USINT 0 to 32	Node number (node ID)
us_TRANSITION_STATE	USINT 1 to 5	Command to the network
x_ENABLE	BOOL	Send command
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
x_CONFIRM	BOOL	Confirms execution
w_ERROR	WORD	Error number
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

### General

The function block CANop405\_NMT allows the execution of management functions for a network with CANopen communication profile. These functions or commands are defined in the *CiA Draft Standard 301* and are as follows:

- 'Start Remote Node'
- 'Stop Remote Node'
- 'Enter Pre Operational'
- 'Reset Node'
- 'Reset Communication'

In general, the network nodes are in 'pre-operational' status after power-on and self-initialisation, so that only the command 'Start remote node' remains to be sent.

### Using function block CANop405\_NMT

The option board CAN-M-01 must be initialised with the function block CANop405\_INIT before the function block CANop405\_NMT can send commands to the network nodes during operation. The function block CANop405\_NMT itself is used in a cyclical task. Input **us\_DEVICE** specifies the node number to which the command is to be transmitted. Number 0 causes the command to be broadcast to all network nodes. Values above 32 cause an error message. The command to be sent is specified at **us\_TRANSITION\_STATE** as follows:

us_TRANSITION_STATE	Command
1	Start Remote Node
2	Stop Remote Node
3	Enter Pre Operational
4	Reset Node
5	Reset Communication

Values outside 1 to 5 cause an error message.

A CANop405\_CTRL\_BMSTRUCT data-type global variable must be linked to **\_CANop405\_CTRL** and linked to the basic address %MB3.3000000 of the CAN-M-01 via the declaration ( *globVar* AT %MB3.3000000 : CANop405\_CTRL\_BMSTRUCT ). The same global variable must be linked to the same input of the other function blocks from the library CANop405\_DLII\_20bd00. The variable is required for data exchange with the option board CAN-M-01 and of no further significance to the user.

The command is sent once with **x\_ENABLE** = TRUE and confirmed by the function block with **x\_CONFIRM** = TRUE. The system does not wait for an acknowledgement from the network node. Any errors are indicated via **w\_ERROR** and **x\_CONFIRM** remains FALSE. The error numbers comply with the definition of CIA405\_CANOPEN\_KERNEL\_ERROR of the *CiA Draft Standard Proposal 405*.

w_ERROR	Definition
16#0000	No error
16#0001 - 16#000F	Not used
16#0010	CAN Bus off
16#0011	CAN Error Passive
16#0012 - 16#001F	Not used
16#0021 - 16#0060	Not used
16#0061	Invalid command
16#0062 - 16#0072	Not used
16#0073	Node number > 32
16#0074 - 16#00FF	Not used

The errors 'CAN bus off' and 'CAN error passive' are displays only and do not abort processing. If no other errors occur, **x\_CONFIRM** = TRUE.

The outputs are reset with **x\_ENABLE** = FALSE. This is required also if another command is to be transmitted.

## 6.6 CANop405\_NODE\_GUARDING

### Description

You may use this function block for CANop405 to implement node guarding during data exchange with CANopen communication profile.



### NOTE

The function block CANop405\_NODE\_GUARDING uses the library BM\_TYPES\_20bd01 or higher.

Parameter input	Data type	Description
x_RESET	BOOL	Reset
us_DEVICE	USINT 0 to 32	Node number (node ID)
t_NODE_GUARD_TIME	TIME	'Node guard time'
u_LIFE_TIME_FACTOR	UINT	'Node life time' factor
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
us_NODE_STATE	USINT	Network node status
x_NODE_OK	BOOL	OK message
w_ERROR	WORD	Error word
dw_ERRORINFO	DWORD	Error information
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

The function block CANop405\_NODE\_GUARDING monitors the network node with the node number us\_DEVICE. On expiry of the period t\_NODE\_GUARD\_TIME, the function block CANop405\_NODE\_GUARDING requests a guarding telegram from the network node with a remote telegram. In this guarding telegram, the network node transmits its current status (us\_NODE\_STATE) and a toggle bit. The sent toggle bit alternates between TRUE and FALSE from telegram to telegram to identify the loss of guarding telegrams. The function block CANop405\_NODE\_GUARDING monitors the alternating toggle bit. An error is output at w\_ERROR if no telegram with the expected toggle bit is received during the 'node life time' (= t\_NODE\_GUARDING\_TIME \* u\_LIFE\_TIME\_FACTOR). This error can be reset only with x\_RESET = TRUE.



### NOTE

The function block CANop405\_NODE\_GUARDING does not transmit the values for node guard time and life time factor to the network node. These values may be sent with function block CANop405\_SDO\_WRITE.

Input `x_RESET`:

`x_RESET = TRUE` resets the function block, setting outputs `x_NODE_OK`, `w_ERROR` and `ud_ERRORINFO` to `FALSE` and/or 0.

If the function block `CANop405_GUARDING_NODE` is re-enabled with `x_RESET = FALSE`, steps must be taken to ensure that the toggle bit = `FALSE` in the next guarding telegram. This is always the case in the first guarding telegram after `CAN ON` or after Reset communication (refer function block `CANop405_NMT`).

Input `us_DEVICE`:

This input specifies the node number of the network node to guarded, supporting nodes from 1 to 32. This input value is accepted only on initial request or at `x_RESET = TRUE`.

Input `t_NODE_GUARD_TIME`:

Within this period, the function block transmits remote telegrams to the network node.

Input `u_LIFE_TIME_FACTOR`:

The node life time for the network node is computed from

$$u\_LIFE\_TIME\_FACTOR * t\_NODE\_GUARD\_TIME$$

If no guarding telegram with the correct toggle bit is received within this period of time, an appropriate error message is output at `w_ERROR` and `dw_ERRORINFO`. This error message is reset only at `x_RESET = TRUE`.

Input/output `_CANop405_CTRL`:

A `CANop405_CTRL_BMSTRUCT` data-type global variable must be linked to `_CANop405_CTRL`, that maps the operating data for the CAN interface. This variable must be linked to the basic address of the CAN interface via the global variable declaration.

Example:

Option board CAN-M-01 for **Omega** Drive-Line II

```
_CANop405Base AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT;
```

where:

<code>_CANop405Base</code>	the name of the variable with the data type identifier ‘_’ for Struct
<code>CANop405_CTRL_BMSTRUCT</code>	the data type
<code>%MB3.3000000</code>	the basic address of the CAN interface on the option board CAN-M-01

# Function blocks for CANopen

---

Output us\_NODE\_STATE:

This output makes available the network node status (the status transmitted in the last valid guarding telegram from the network node).

us_NODE_STATE	Description
4	Stopped
5	Operational
127	Pre-Operational

Output x\_NODE\_OK:

x\_NODE\_OK = TRUE indicates that a valid guarding telegram has been received within the node life time. x\_NODE\_OK is set to TRUE by the first valid guarding telegram. x\_NODE\_OK is set to FALSE if no valid guarding telegram is received within the node life time.

Output w\_ERROR:

Any error outputs an error number at w\_ERROR. The error numbers comply with the definition of *CIA405\_CANOPEN\_KERNEL\_ERROR* of *CiA Draft Standard Proposal 405*. A further description of the errors is made available at output ud\_ERRORINFO.

Output ud\_ERRORINFO:

At w\_ERROR = 16#0001:

These error messages comply with the SDO abort code of *CiA Draft Standard Proposal 301*.

ud_ERRORINFO	Description
16# 0503 0000	Toggle bit not alternated.

At w\_ERROR = 16#0021:

Manufacturer-specific error messages

ud_ERRORINFO	Description
16# 0000 0072	us_Device = USINT#0
16# 0000 0073	us_Device > USINT#32
16# 0000 0074	Unexpected network node status (us_NODE_STATE not 4,5 or 127)
16# 0000 0075	Communication timeout – no transmission possible within the t_NODE_GUARD_TIME time frame.

## 6.7 CANop405\_PDO\_READ

### Description

You may use this function block for CANop405 to receive a process data object (PDO) from a network node during data exchange with CANopen communication profile.



### NOTE

The function block CANop405\_PDO\_READ uses the library BM\_TYPES\_20bd01 or higher.

Parameter input	Data type	Description
x_ENABLE	BOOL	Enable
us_PDO_NR	USINT 1 to 40	Read command number
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
dw_DATA0	DWORD	Data bytes 0 to 3
dw_DATA1	DWORD	Data bytes 4 to 7
us_DATALENGTH	USINT	Data length in bytes
w_ERROR	WORD	Error number
x_CONFIRM	BOOL	Confirms execution
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

PDO reading is enabled at x\_ENABLE = TRUE. During initialisation, function block CANop405\_INIT specifies which telegram with what COB identifier is associated with what read command number. The composition of the COB identifier complies with the definition in the *CiA Draft Standard 301*. The function block CANop405\_PDO\_READ specifies the read command number at us\_PDO\_NR. The function block CANop405\_PDO\_READ indicates with x\_CONFIRM = TRUE that a network node is sending the telegram in question. The data received are output at dw\_DATA0 and dw\_DATA1. The number of data bytes received is output at us\_DATALENGTH. Errors are indicated at output w\_ERROR.



### NOTE

A maximum of 40 PDOs may be read. The read command number (1 to 40) is indicated at us\_PDO\_NR. This number must not be active at several CANop405\_PDO\_READ function blocks simultaneously.



## NOTE

While the function block allows PDO reception, it does not request PDOs. The PDOs are sent independently by the network node or requested via remote telegrams via function block CANop405\_SDO\_WRITE.

Input: x\_ENABLE

PDO reading is enabled via x\_ENABLE = TRUE.

Input: us\_PDO\_NR

A maximum of 40 PDOs may be read. The read command number (1 to 40) is indicated at us\_PDO\_NR. This number must not be active at several CANop405\_PDO\_READ function blocks simultaneously.

During initialisation (with function block CANop405\_INIT), the read command number is allocated a COB identifier. The composition of the COB identifier complies with the definition in the *CiA Draft Standard 301*.

Example:

If the COB identifier 16#0181 is entered at input a\_PDO\_READ\_COB\_ID with an index 3 at function block CANop405\_INIT, output is made of the data of the telegrams with COB identifier 16#0181 at function block CANop405\_READ with us\_PDO\_NR = 3.

The read command number is of internal significance only and not related to COB identifier, node ID or similar of the *CiA Draft Standard 301*.

Input/output \_CANop405\_CTRL:

A CANop405\_CTRL\_BMSTRUCT data-type global variable must be linked to \_CANop405\_CTRL, that maps the operating data for the CAN interface. This variable must be linked to the basic address of the CAN interface via the global variable declaration.

Example:

Option board CAN-M-01 for **Ω**mega Drive-Line II

```
_CANop405Base AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT;
```

where:

_CANop405Base	the name of the variable with data type identifier '_' for Struct
CANop405_CTRL_BMSTRUCT	the data type
%MB3.3000000	the basic address of the CAN interface on the option board CAN-M-01

Output: dw\_DATA0

The read data are made available at this output (bytes 0 to 3).



Output: dw\_DATA1

The read data are made available at this output (bytes 4 to 7).

Output: us\_DATALENGTH

Corresponds to the number of bytes read from dw\_DATA0 and dw\_DATA1.

Valid bytes of outputs dw\_DATA0 and dw\_DATA1:

us_DATALENGTH	BYTE 3 dw_ DATA1	BYTE 2 dw_ DATA1	BYTE 1 dw_ DATA1	BYTE 0 dw_ DATA1	BYTE 3 dw_ DATA0	BYTE 2 dw_ DATA0	BYTE 1 dw_ DATA0	BYTE 0 dw_ DATA0
1	-	-	-	-	-	-	-	Yes
2	-	-	-	-	-	-	Yes	Yes
3	-	-	-	-	-	Yes	Yes	Yes
4	-	-	-	-	Yes	Yes	Yes	Yes
5	-	-	-	Yes	Yes	Yes	Yes	Yes
6	-	-	Yes	Yes	Yes	Yes	Yes	Yes
7	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Output w\_ERROR:

Any error outputs an error number at w\_ERROR.

w_ERROR	Description
16# 0008	us_PDO_NR = 0 or us_PDO_NR > 40

Output: x\_CONFIRM

x\_CONFIRM = TRUE indicates a successful PDO read.

## 6.8 CANop405\_PDO\_WRITE

### Description

You may use this function block for CANop405 to write a process data object (PDO) to a network node, or to request PDOs from the network node, during data exchange with CANopen communication profile.



### NOTE

The function block CANop405\_PDO\_WRITE uses the library BM\_TYPES\_20bd01 or higher.

Parameter input	Data type	Description
x_ENABLE	BOOL	Enable
x_REMOTE	BOOL	Requests remote telegram
dw_DATA0	DWORD	Data bytes 0 to 3
dw_DATA1	DWORD	Data bytes 4 to 7
us_DATALENGTH	USINT	Data length in bytes
w_PDO_COB_ID	WORD	COB-ID
us_PDO_NR	USINT 1 to 20	Write command number
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
w_ERROR	WORD	Error number
x_CONFIRM	BOOL	Confirms execution
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

At x\_ENABLE = TRUE, one PDO is written with

COB identifier: w\_PDO\_COB\_ID,  
data: dw\_DATA0 and dw\_DATA1 and  
length: us\_DATALENGTH

For the composition of the COB identifier (node number and function code) refer to the definition in the *CiA Draft Standard 301*. Successful PDO transmission is indicated by x\_CONFIRM = TRUE. Errors are output at w\_ERROR.

At x\_REMOTE = TRUE and x\_ENABLE = TRUE, one PDO with

COB identifier: w\_PDO\_COB\_ID and  
length: us\_DATALENGTH

is requested from a network node via a remote telegram.



## NOTE

A maximum of 20 PDOs may be written simultaneously. The write command number (1 to 20) is indicated at `us_PDO_NR`. This number must not be active at several `CANop405_PDO_WRITE` function blocks simultaneously.

Input: `x_ENABLE`

Enable for PDO transmit or remote telegram transmit via `x_ENABLE = TRUE`.

Input: `x_REMOTE`

If this input = TRUE, a remote telegram is transmitted at `x_ENABLE = TRUE`, thus requesting from a network node the telegram with `w_PDO_COB_ID`. If COB identifier allocation is effected as per definition in the *CiA Draft Standard 301*, `w_PDO_COB_ID` consists of the number of the network node in question and the function code as follows.

<code>w_PDO_COB_ID</code>	Definition
Bit 0..6	Node number
Bit 7..10	Function Code
Bit 11..15	Reserved

Object	Function Code
PDO1 (tx)	2#0011
PDO2 (tx)	2#0101
PDO3 (tx)	2#0111
PDO4 (tx)	2#1001

Example:

Requesting a PDO2 (tx) from network node 3

<code>w_PDO_COB_ID</code>	
Bit 0..6	2#0000011
Bit 7..10	2#0101
Bit 11..15	2#00000

results in: `2#0000_0010_1000_0011 = 16#0283`

## Function blocks for CANopen

---

Input: dw\_DATA0

This input specifies bytes 0 to 3 of the data to be written.

Input: dw\_DATA1

This input specifies bytes 4 to 7 of the data to be written.

Input: us\_DATALENGTH

Corresponds to the number of bytes to be transmitted from dw\_DATA0 and dw\_DATA1. This data length must be identical to the target object.

us_DATALENGTH	BYTE 3 dw_DATA1	BYTE 2 dw_DATA1	BYTE 1 dw_DATA1	BYTE 0 dw_DATA1	BYTE 3 dw_DATA0	BYTE 2 dw_DATA0	BYTE 1 dw_DATA0	BYTE 0 dw_DATA0
1	-	-	-	-	-	-	-	Yes
2	-	-	-	-	-	-	Yes	Yes
3	-	-	-	-	-	Yes	Yes	Yes
4	-	-	-	-	Yes	Yes	Yes	Yes
5	-	-	-	Yes	Yes	Yes	Yes	Yes
6	-	-	Yes	Yes	Yes	Yes	Yes	Yes
7	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Input: w\_PDO\_COB\_ID

If COB identifier allocation is effected as per definition in the *CiA Draft Standard 301*, w\_PDO\_COB\_ID consists of the number of the network node in question and the function code as follows.

w_PDO_COB_ID	Definition
Bit 0..6	Node number
Bit 7..10	Function Code
Bit 11..15	Reserved

Object	Function Code
PDO1 (rx)	2#0100
PDO2 (rx)	2#0110
PDO3 (rx)	2#1000
PDO4 (rx)	2#1010

Examples:

1. Transmitting a PDO1 (rx) to network node 1

w_PDO_COB_ID	
Bit 0..6	2#0000001
Bit 7..10	2#0100
Bit 11..15	2#00000

results in: 2#0000\_0010\_0000\_0001 = 16#0201

2. Transmitting a PDO4 (rx) to network node 32

w_PDO_COB_ID	
Bit 0..6	2#0100000
Bit 7..10	2#1010
Bit 11..15	2#00000

results in: 2#0000\_0101\_0010\_0000 = 16#0520

The function block CANop405\_COB\_ID may also be used to compose the COB identifier.

Input: us\_PDO\_NR

A maximum of 20 PDOs may be written simultaneously. The write command number (1 to 20) is indicated at us\_PDO\_NR. This number must not be active at several CANop405\_PDO\_WRITE function blocks simultaneously. The write command number is of internal significance only and not related to COB identifier, node ID or similar of the *CiA Draft Standard 301*.

Input/output \_CANop405\_CTRL:

A CANop405\_CTRL\_BMSTRUCT data-type global variable must be linked to \_CANop405\_CTRL, that maps the operating data for the CAN interface. This variable must be linked to the basic address of the CAN interface via the global variable declaration.

Example:

Option board CAN-M-01 for **Omega** Drive-Line II

```
_CANop405Base AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT;
```

where:

_CANop405Base	the name of the variable with data type identifier '_' for Struct
CANop405_CTRL_BMSTRUCT	the data type
%MB3.3000000	the basic address of the CAN interface on the option board CAN-M-01

## Function blocks for CANopen

---

Output w\_ERROR:

Any error outputs an error number at w\_ERROR.

w_ERROR	Description
16# 0001	us_DATALENGTH > 8
16# 0002	w_PDO_COB_ID < 16#0081
16# 0004	w_PDO_COB_ID > 16#067F
16# 0008	us_PDO_NR = 0 or us_PDO_NR > 20

## 6.9 CANop405\_SDOx\_READ

Applies analogously for function blocks CANop405\_SDO1\_READ to CANop405\_SDO8\_READ.

### Description

You may use this function block for CANop405 to read a service data object (SDO) from a network node during data exchange with CANopen communication profile. Data transmission is in the expedited transfer mode.



### NOTE

The function block CANop405\_SDOx\_READ uses the library BM\_TYPES\_20bd01 or higher (x = 1 to 8).

Parameter input	Data type	Description
us_DEVICE	USINT 1 to 32	Node number (node ID)
w_INDEX	WORD	Index of the object to be read
b_SUBINDEX	BYTE	Subindex of the object to be read
x_ENABLE	BOOL	Enable
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
dw_DATA	DWORD	Read data
us_DATALENGTH	USINT	Length of read data in bytes
x_CONFIRM	BOOL	Confirms execution
w_ERROR	WORD	Error number
ud_ERRORINFO	UDINT	Error description
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

A rising edge at input x\_ENABLE starts the read. For this purpose, the SDO with w\_INDEX and b\_SUBINDEX is requested from the network node with the node number us\_DEVICE. x\_CONFIRM = TRUE indicates a successful read. The read data are made available at output dw\_DATA, and their length at output us\_DATALENGTH. Any errors are displayed at output w\_ERROR and described in more detail at output ud\_ERRORINFO. x\_ENABLE = FALSE terminates the read and resets the outputs of the function block. Also refer function block description CANop405\_INIT.



### NOTE

The CAN interface CAN-M-01 can start up to 8 SDO commands simultaneously, with one function block being available for each SDO command.

CANop405\_SDOx\_READ and CANop405\_SDOx\_WRITE (x = 1 to 8).

The function blocks CANop405\_SDO1\_READ and CANop405\_SDO1\_WRITE must not be active simultaneously.

No multiple instances of function block CANop405\_SDO1\_READ must be active at the same time.

Function blocks CANop405\_SDOx\_READ and function block CANop405\_SDOx\_WRITE must not have the same node number (us\_DEVICE) at the same time (x = 1 to 8).

Input: us\_DEVICE

This input specifies the node number of the network node from which a service data object (SDO) is to be read, supporting nodes from 1 to 32.

Input: w\_INDEX

This input indicates the index of the object to be read.

Input: b\_SUBINDEX

This input indicates the subindex of the object to be read.

Input: x\_ENABLE

A rising edge at input x\_ENABLE starts the read.

Input/output \_CANop405\_CTRL:

A CANop405\_CTRL\_BMSTRUCT data-type global variable must be linked to \_CANop405\_CTRL, that maps the operating data for the CAN interface. This variable must be linked to the basic address of the CAN interface via the global variable declaration.



Example:

Option board CAN-M-01 for **Omega** Drive-Line II

```
_CANop405Base AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT;
```

where:

<code>_CANop405Base</code>	the name of the variable with data type identifier '_' for Struct
<code>CANop405_CTRL_BMSTRUCT</code>	the data type
<code>%MB3.3000000</code>	the basic address of the CAN interface on the option board CAN-M-01

Output `dw_DATA`:

This output makes the read data available. The length of the read data is output at `us_DATALENGTH`.

Output `us_DATALENGTH`:

Number of read bytes.

Valid bytes of output `dw_DATA`:

<code>us_DATALENGTH</code>	Bit 31 - Bit 24	Bit 23 - Bit 16	Bit 15 - Bit 8	Bit 7 - Bit 0
1	-	-	-	Yes
2	-	-	Yes	Yes
3	-	Yes	Yes	Yes
4	Yes	Yes	Yes	Yes

Output `x_CONFIRM`:

`x_CONFIRM = TRUE` confirms a successful SDO read.

Output `w_ERROR`:

Any error outputs an error number at `w_ERROR`. The error numbers comply with the definition of `CIA405_CANOPEN_KERNEL_ERROR` of the *CiA Draft Standard Proposal 405*.

Output `ud_ERRORINFO`:

At `w_ERROR = 16#0001`:

These error messages comply with the SDO abort code of *CiA Draft Standard Proposal 301*.

<code>ud_ERRORINFO</code>	Description
16# 0503 0000	Toggle bit not alternated.
16# 0504 0000	SDO protocol timed out.
16# 0504 0001	Client/server command specifier not valid or unknown.
16# 0504 0002	Invalid block size (block mode only).
16# 0504 0003	Invalid sequence number (block mode only).

## Function blocks for CANopen

---

<b>ud_ERRORINFO</b>	<b>Description</b>
16# 0504 0004	CRC error (block mode only).
16# 0504 0005	Out of memory.
16# 0601 0000	Unsupported access to an object.
16# 0601 0001	Attempt to read a write only object.
16# 0601 0002	Attempt to write a read only object.
16# 0602 0000	Object does not exist in the object dictionary.
16# 0604 0041	Object cannot be mapped to the PDO.
16# 0604 0042	The number and length of the objects to be mapped would exceed PDO length.
16# 0604 0043	General parameter incompatibility reason.
16# 0604 0047	General internal incompatibility in the device.
16# 0606 0000	Access failed due to an hardware error.
16# 0607 0010	Data type does not match, length of service parameter does not match
16# 0607 0012	Data type does not match, length of service parameter too high
16# 0607 0013	Data type does not match, length of service parameter too low
16# 0609 0011	Sub-index does not exist.
16# 0609 0030	Value range of parameter exceeded (only for write access).
16# 0609 0031	Value of parameter written too high.
16# 0609 0032	Value of parameter written too low.
16# 0609 0036	Maximum value is less than minimum value.
16# 0800 0000	general error
16# 0800 0020	Data cannot be transferred or stored to the application.
16# 0800 0021	Data cannot be transferred or stored to the application because of local control
16# 0800 0022	Data cannot be transferred or stored to the application because of the present device state.
16# 0800 0023	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error).

At w\_ERROR = 16#0021:

<b>ud_ERRORINFO</b>	<b>Description</b>
16# 0000 0072	us_Device = USINT#0
16# 0000 0073	us_Device > USINT#32

## 6.10 CANop405\_SDOx\_WRITE

Applies analogously for function blocks CANop405\_SDO1\_WRITE to CANop405\_SDO8\_WRITE.

### Description

You may use this function block for CANop405 to write a service data object (SDO) to a network node during data exchange with CANopen communication profile. Data transmission is in the expedited transfer mode.



### NOTE

The function block CANop405\_SDOx\_WRITE uses the library BM\_TYPES\_20bd01 or higher (x = 1 to 8).

Parameter input	Data type	Description
us_DEVICE	USINT 1 to 32	Node number (node ID)
w_INDEX	WORD	Index of the object to be written
b_SUBINDEX	BYTE	Subindex of the object to be written
x_ENABLE	BOOL	Enable
dw_DATA	DWORD	Data
us_DATALENGTH	USINT	Data length in bytes
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
x_CONFIRM	BOOL	Confirmation
w_ERROR	WORD	Error number
ud_ERRORINFO	UDINT	Error description
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

A rising edge at input x\_ENABLE starts the write. For this purpose, the SDO with w\_INDEX and b\_SUBINDEX is written to the network node with the node number us\_DEVICE. The data to be written are specified at input dw\_DATA, their length in bytes at input us\_DATALENGTH.

x\_CONFIRM = TRUE indicates a successful write. Any errors are displayed at w\_ERROR and described in more detail at ud\_ERRORINFO. x\_ENABLE = FALSE terminates the write and resets the outputs of the function block.



### NOTE

The CAN interface CAN-M-01 can start up to 8 SDO commands simultaneously, with one function block being available for each SDO command.

CANop405\_SDOx\_WRITE and CANop405\_SDOx\_READ (x = 1 to 8).

The function blocks CANop405\_SDO1\_WRITE and CANop405\_SDO1\_READ must not be active simultaneously.

No multiple instances of function block CANop405\_SDO1\_WRITE must be active at the same time.

Function blocks CANop405\_SDOx\_WRITE and function block CANop405\_SDOx\_READ must not have the same node number (us\_DEVICE) at the same time (x = 1 to 8).

Input: us\_DEVICE

This input specifies the node number of the network node to which a service data object (SDO) is to be written, supporting nodes from 1 to 32.

Input: w\_INDEX

This input indicates the index of the object to be written.

Input: b\_SUBINDEX

This input indicates the subindex of the object to be written.

Input: x\_ENABLE

A rising edge at input x\_ENABLE starts the write.

Input dw\_DATA:

This input specifies the data to be written.

Input us\_DATALENGTH:

Corresponds to the number of bytes to be transmitted from dw\_DATA. This data length must be identical to the target object.

Valid bytes of input dw\_DATA:

us_DATALENGTH	Bit 31 - Bit 24	Bit 23 - Bit 16	Bit 15 - Bit 8	Bit 7 - Bit 0
1	-	-	-	Yes
2	-	-	Yes	Yes
3	-	Yes	Yes	Yes
4	Yes	Yes	Yes	Yes

Input/output \_CANop405\_CTRL:

A CANop405\_CTRL\_BMSTRUCT data-type global variable must be linked to \_CANop405\_CTRL, that maps the operating data for the CAN interface. This variable must be linked to the basic address of the CAN interface via the global variable declaration.

Example:

Option board CAN-M-01 for mega Drive-Line II

```
_CANop405Base AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT;
```

where:

\_CANop405Base                      the name of the variable with data type identifier '\_' for Struct

CANop405\_CTRL\_BMSTRUCT          the data type

%MB3.3000000                      the basic address of the CAN interface on the option board CAN-M-01

Output x\_CONFIRM:

x\_CONFIRM = TRUE confirms a successful SDO write.

Output w\_ERROR:

Any error outputs an error number at w\_ERROR. The error numbers comply with the definition of CIA405\_CANOPEN\_KERNEL\_ERROR of the *CiA Draft Standard Proposal 405*.

Output ud\_ERRORINFO:

At w\_ERROR = 16#0001:

These error messages comply with the SDO abort code of *CiA Draft Standard Proposal 301*.

ud_ERRORINFO	Description
16# 0503 0000	Toggle bit not alternated.
16# 0504 0000	SDO protocol timed out.
16# 0504 0001	Client/server command specifier not valid or unknown.
16# 0504 0002	Invalid block size (block mode only).
16# 0504 0003	Invalid sequence number (block mode only).
16# 0504 0004	CRC error (block mode only).
16# 0504 0005	Out of memory.
16# 0601 0000	Unsupported access to an object.

## Function blocks for CANopen

---

<b>ud_ERRORINFO</b>	<b>Description</b>
16# 0601 0001	Attempt to read a write only object.
16# 0601 0002	Attempt to write a read only object.
16# 0602 0000	Object does not exist in the object dictionary.
16# 0604 0041	Object cannot be mapped to the PDO.
16# 0604 0042	The number and length of the objects to be mapped would exceed PDO length.
16# 0604 0043	General parameter incompatibility reason.
16# 0604 0047	General internal incompatibility in the device.
16# 0606 0000	Access failed due to an hardware error.
16# 0607 0010	Data type does not match, length of service parameter does not match
16# 0607 0012	Data type does not match, length of service parameter too high
16# 0607 0013	Data type does not match, length of service parameter too low
16# 0609 0011	Sub-index does not exist.
16# 0609 0030	Value range of parameter exceeded (only for write access).
16# 0609 0031	Value of parameter written too high.
16# 0609 0032	Value of parameter written too low.
16# 0609 0036	Maximum value is less than minimum value.
16# 0800 0000	general error
16# 0800 0020	Data cannot be transferred or stored to the application.
16# 0800 0021	Data cannot be transferred or stored to the application because of local control
16# 0800 0022	Data cannot be transferred or stored to the application because of the present device state.
16# 0800 0023	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error).

At w\_ERROR = 16#0021:

<b>ud_ERRORINFO</b>	<b>Description</b>
16# 0000 0070	us_Datalength > USINT#4
16# 0000 0071	us_Datalength = USINT#0
16# 0000 0072	us_Device = USINT#0
16# 0000 0073	us_Device > USINT#32

## 6.11 CANop405\_SYNC

### Description

You may use this function block for CANop405 to transmit a SYNC object during data exchange with CANopen communication profile.



### NOTE

The function block CANop405\_SYNC uses the library BM\_TYPES\_20bd01 or higher.

Parameter input	Data type	Description
x_ENABLE	BOOL	Enable
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

Parameter output	Data type	Description
x_CONFIRM	BOOL	Confirmation
_CANop405_CTRL	CANop405_CTRL_BMSTRUCT	CAN interface operating data

### General

The function block CANop405\_SYNC may be used for SYNC object transmission. These SYNC objects may be used for network node synchronisation. At x\_ENABLE = TRUE, one SYNC telegram is transmitted. x\_CONFIRM = TRUE indicates a successful transmit.

A CANop405\_CTRL\_BMSTRUCT data-type global variable must be linked to \_CANop405\_CTRL, that maps the operating data for the CAN interface. This variable must be linked to the basic address of the CAN interface via the global variable declaration.

Example:

Option board CAN-M-01 for mega Drive-Line II

```
_CANop405Base AT %MB3.3000000 : CANop405_CTRL_BMSTRUCT;
```

where:

\_CANop405Base                      the name of the variable with data type identifier '\_' for Struct

CANop405\_CTRL\_BMSTRUCT          the data type

%MB3.3000000                      the basic address of the CAN interface on the option board CAN-M-01





## 7 INDEX

### A

Appropriate Use ..... 6

### B

Basic address ..... 19

Baud rate ..... 30

### C

CAN controller ..... 15

    Technical data ..... 8

CAN-M-01

    Technical data ..... 8

CANop405\_DLII\_20bd01 ..... 18

CANopen

    Characteristics ..... 15

CANopen communication ..... 19

Compute COB identifier ..... 24

Connection cables ..... 11

### D

Danger Information ..... 5

Declaration of a global structured variable ... 19

DIP switch

    Read out ..... 21

### E

EMERGENCY (error handling) ..... 17

Emergency telegram ..... 26

Error handling (EMERGENCY) ..... 17

### F

Function block CANop405\_COB\_ID ..... 24

Function block CANop405\_EMERGENCY . 20,  
26

Function block CANop405\_INIT ..... 29

Function block CANop405\_NMT ..... 32

Function block CANop405\_NODE\_GUARDING  
20, ..... 34

Function block CANop405\_PDO\_READ .... 20,  
37

Function block CANop405\_PDO\_WRITE ... 20,  
40

Function block CANop405\_SDO1\_READ ... 20

Function block CANop405\_SDO1\_WRITE . 20

Function block CANop405\_SDOx\_READ ... 45

Function block CANop405\_SDOx\_WRITE .. 49

Function block CANop405\_SYNC ..... 20, 53

Function blocks for CANopen

    Overview ..... 23

### G

Global structured variable ..... 19

Guarding telegram ..... 34

### I

Implement cyclical communication ..... 19

Implement initialisation ..... 19

Implementation of cyclical communication .. 19

Implementation of the initialisation ..... 19

Initialisation ..... 29

ISO 11898 ..... 15

### L

Library

    Integrate ..... 19

Library CANop405\_DLII\_20bd01 ..... 18

Library integration ..... 19

### M

Monitoring time ..... 30

### N

Network management (NMT) ..... 17

Network management functions ..... 32

NMT (network management) ..... 17

Node guarding ..... 34

NODE GUARDING (node guarding) ..... 17

Node guarding (NODE GUARDING) ..... 17

### O

Option board CAN-M-01

    Technical data ..... 8

## P

PDO .....	15, 17
Receive .....	37
Request .....	40
Write .....	40
Pin assignment .....	10
Process data objects	
See PDO	
Programming .....	18
PROPROG wt II .....	18

## Q

Qualified Personnel .....	6
---------------------------	---

## S

Safety Information .....	5
SDO .....	15, 17
Read .....	45
Write .....	49
Service data objects	
See SDO	
Software function .....	10
Software version of the option board	
Check .....	21
SYNC (synchronisation) .....	17
Synchronisation (SYNC) .....	17

## T

Terminating resistor connectors .....	12
Transmit SYNC object .....	53