SIEMENS

MICROMASTER 420/430/440
CANopen Option Module

Operating Instructions **Issue 01/05**

User Documentation
6SE6400-5BC00-0BP0

SIEMENS

MICROMASTER 420/430/440
CANopen Option Module

Valid for Issue 01/05

Drive converter type
MICROMASTER 420/430/440

Issue 01/05

Additional information can be found in the Internet under:
http://www.siemens.de/micromaster

The certified Siemens Quality for Software And Training corresponds to DIN ISO 9001, Reg. No. 2160-01

MICROMASTER® is a registered trademark of Siemens AG.

It is possible that other functions are available which are not described in this documentation. However, this does not mean that we are responsible in providing such functions with a new control system or when servicing the equipment.

We have checked the contents of this document to ensure that they coincide with the described hardware and software. However, differences cannot be completely excluded, so that we do not accept any guarantee for complete conformance. However, the information in this document is regularly checked and necessary corrections will be included in subsequent editions. We are grateful for any recommendations for improvement.
Siemens Manuals are printed on chlorine-free paper from professionally managed forests. No solvents are used for printing or binding.

Changes to this documentation can be made without prior notice.

Order No.: 6SE6400-5BC00-0BP0
Printed in Germany

Siemens-Aktiengesellschaft.

Definitions and Warning Information

Safety Guidelines

This documentation contains notes which you should observe to ensure your own personal safety and also to prevent material damage. The information and instructions for your personal safety are identified by a warning triangle. Information and instructions on general material damage do not have a warning triangle. depending on the degree of danger, they are indicated as follows:

DANGER

For the purpose of this documentation and on the warning label of the product itself, "Danger" indicates that death, grievous injury or extensive damage to property **will** occur if the appropriate precautions are not taken.

WARNING

For the purpose of this documentation and on the warning label of the product itself, "Warning" indicates that death, grievous injury or extensive damage to property **may** occur if the appropriate precautions are not taken.

CAUTION

"Caution" with a warning triangle, means that minor personal injury may occur if the appropriate precautions are not taken.

CAUTION

"Caution" without a warning triangle, means that damage to property may occur if the appropriate precautions are not taken.

NOTICE

means that an undesirable situation or condition can occur if the appropriate information/instruction is not observed.

NOTE

For this purpose of this documentation, "Note" highlights an important item of information about the product or a section of the instructions which requires careful attention.

Qualified personnel

For the purpose of these Operating Instructions and on the product label, "qualified personnel" are those familiar with the installation, mounting, start-up and operation of the equipment and the hazards involved. He or she must have the following qualifications:

Trained and authorized to energize, de-energize, clear, ground and tag circuits and equipment in accordance with established safety procedures.

Trained in the proper care and use of protective equipment in accordance with established safety procedures.

Trained in rendering first aid.

User Documentation

WARNING

Before installing and commissioning the equipment, please carefully read the safety-related information/instructions and warnings as well as the warning labels on the equipment itself. Please ensure that the warning labels are kept in a condition where they can be easily read and missing or damaged labels/information are replaced.

Intended use

WARNING

When in operation, electrical equipment contains components that are under dangerous voltage.

Failure to adhere to these instructions can result in death, severe bodily injury or property damage!

Only appropriately qualified personnel may work on this equipment or be in its vicinity. These personnel must be completely knowledgeable about all warnings and service measures as specified in these Operating Instructions.

The successful and safe operation of this equipment is dependent on proper handling, installation, operation and service.

National safety guidelines and regulations must be carefully observed.

List of contents

1 Description

The CANopen communications module (CANopen option module) is used to connect MICROMASTER 420/430/440 drives to higher-level automation systems with a CAN-bus.

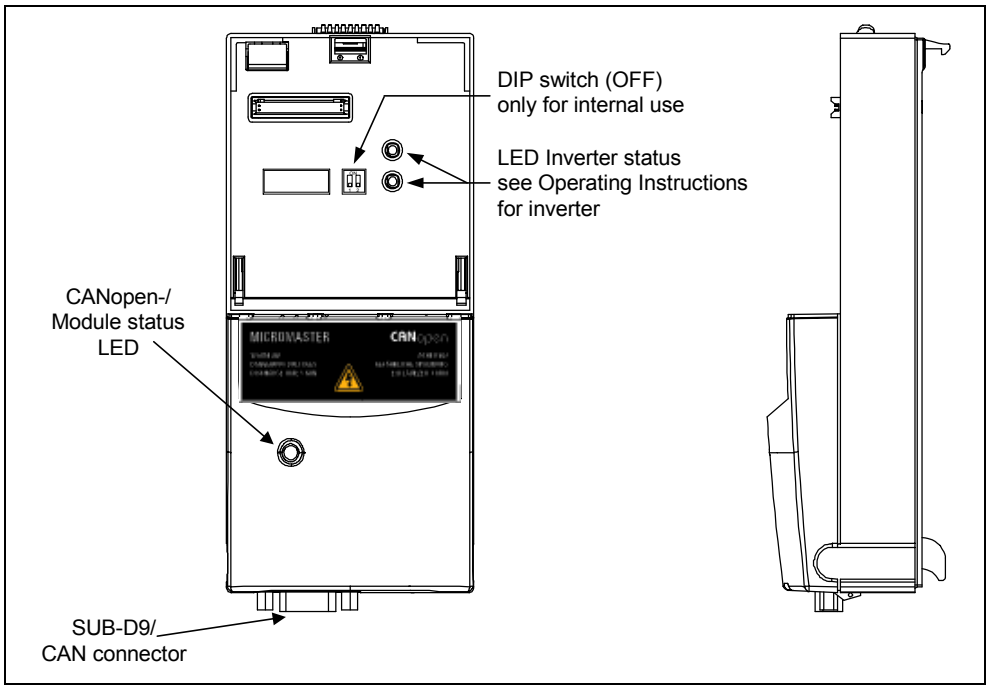


Fig. 1-1 View of the CANopen option module

Technical data

The user can obtain all of the necessary information about the instantaneous status of the CANopen option module using an LED which displays the status of both the module and communications. More detailed diagnostics information can be directly read-out of the diagnostic memory of the CANopen option module using the diagnostics parameters.

The CANopen option module uses a 9-pin sub-D connector to connect to the CAN bus system.

The following baud rates are supported: 10, 20, 50, 125, 250, 500, 800 kbaud and 1 Mbaud.

Functionality of CANopen

- The CANopen option module for MICROMASTER 420/430/440 supports the CANopen data transfer types with SDOs (Service Data Objects) as well as with PDOs (Process Data Objects)
- Further, the CANopen option module supports PDO mapping – with certain restrictions.
- The CANopen option module for MICROMASTER 420/430/440 supports the CANopen communications profile DS 301 Version 4.0, the device profile DSP 402 (Drives and Motion Control) Version 1.1 and the indicator profile DR303-3 Version 1.0.
- For communication monitoring, the CANopen option module supports the Node Guarding/Life Guarding as well as the Heartbeat protocol (Heartbeat Producer) and SYNC loss detection (using communication cycle time.
- The CANopen option module offers an SDO->parameter channel with which all of the MICROMASTER 420/430/440 parameters can be read or written into. In this case, it does not involve CANopen objects, as the numbering of the sub-indices differs with respect to CAN-open (subindex 0 does not contain the number of objects, but already the first object). However, the objects from the parameter channel can, just like CANopen objects, be read and written using SDO access operations - they are designated with a * in the following.
- The CANopen option module supports free process data objects in order to be able to read or write all of the process data, available in the drive, via the CAN bus.
- With the CANopen option module, MICROMASTER 440 supports the Profile Torque Mode and the Velocity Mode; whilst the MICROMASTER 420/430 only supports the Velocity Mode.
- The default settings of the drive are as follows: NodeID=3, baud rate=10kbit/s

2 General CAN/CANopen definitions

2.1 CAN (Controller-Area-Network)

Apart from its use in mobile systems, the Controller-Area-Network (CAN) protocol is also deployed in other applications when allied to additional specific and standardized higher protocol and profile specifications. Alongside its use as an internal bus in mobile systems, its other main applications are in internal communication between plant and machinery.

The major performance features and characteristics of the CAN protocol standardized to ISO-WS 11898 parts I, 2 and 3 are described below.

Message-oriented protocol

The CAN protocol does not exchange data by addressing the recipient of the message, but rather marks each transmitted message with a message identifier. All nodes in the network check the identifier when they receive a message to see whether it is relevant to them. Messages can therefore be accepted by none, one, several or all participants (multicasting, broadcasting).

Prioritization of messages

As the identifier on a message also determines its priority for accessing the bus, it is possible to specify a correspondingly rapid bus access for messages according to their importance. Especially important messages can thus gain access to the bus without a prolonged wait-time, regardless of the loading on the bus at that moment. This characteristic means that especially important messages are transmitted with priority even in exceptional situations (e.g. in the case of longer-lasting disturbances), thereby ensuring proper functioning of a system even during phases of restricted transmission capacity.

Multi-Master capability

Bus access rights are not issued by a meta-level control unit (bus master) per network. Each participant can rather start to send a message with equal rights as soon as the bus has become free. If several participants access the bus at the same time, an arbitration process allocates each participant the bus access right in line with the priority of the message they want to send at that particular moment. Each participant can therefore communicate directly with every other participant. As the transmission of a message can be initiated by the message source itself, then in the case of event-controlled transmission of messages, the bus is only occupied when a new message is on-hand.

No-loss bus arbitration

As the bus is accessed at random under the CAN protocol, it is possible that several participants want to occupy the bus at the same time. In other random bus access routines (e.g. CSMA/CD), this causes the destruction of the suppressed messages. In order to solve such a bus access conflict, a repeated occupation of the bus is required using an appropriate triggering strategy. The CAN protocol therefore deploys a routine to ensure that the message with the highest priority at any given time is sent without any destruction of message contents.

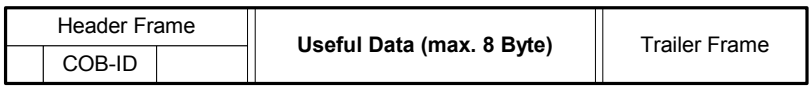
Short block lengths

The maximum data length of a CAN message (CAN communication object) is limited to 8 bytes. This data length is usually sufficient to transmit the information occurring in the lowest field area in a CAN message.

The maximum possible latency for the highest priority message in a CAN network based on 8 byte-long messages is 130 bit times.

Structure of a data telegram

A CAN data telegram consists of a header containing the CAN identifier which is a utility data area, this can contain zero to eights bytes of utility data, and a trailer.



2.2 CANopen

CANopen is a standardized application for distributed industrial automation systems based on CAN as well as the CAL communication Standards. CANopen is a CAN Standard in automation (CiA), and just shortly after it became available, enjoyed widespread use.

In Europe, CANopen can be considered as one of the essential Standards to implement industrial CAN-based system solutions.

CANopen is based on a so-called "communications profile", which specifies the communication mechanisms, used as basis, and their description [CiA DS 301].

The most important device types, used in industrial automation technology, include for example

- Digital and analog I/O modules [CiA DS 401]
- Drive and MotionControl [CiA DSP 402]
- Operator control devices [CiA DSP 403]
- Controllers [CiA DSP 404]
- Programmable control systems [CiA DSP 405]
- Encoders [CiA DSP 406]

are described in so-called "Device profiles".

The functionality of standard devices of the particular type are defined in the device profiles.

The central element of the CANopen Standard is the description of the device functionality using an "Object directory" (OD).

The object directory is sub-divided into an area which contains general data about the device – such as device identification, manufacturer's name and communication partners – and also defines part of the device functionality.

A 16-bit index and an 8-bit subindex are used to identify an entry ("Object") of the object directory.

Using the entries in this object directory, the "Application objects“ of a device, e.g. input and output signals, device parameters, device functions or network variables are made accessible, in a standardized form via the network.

Just the same as other fieldbus systems, CANopen differentiates between two basic data transfer mechanisms: The fast data transfer of short process data via so-called "Process data objects" (PDOs, Process Data Objects) as well as access to entries of the object directory via so-called "Service data objects" (SDOs, Service Data Objects). The latter are predominantly used to transfer parameters while configuring the device as well as generally transferring longer data areas. Generally, process data objects are event-orientated, cyclic or are transferred when requested as broadcast object without any additional protocol overhead.

A maximum of 8 bytes of data can be transferred in a PDO. In conjunction with a synchronization message, sending as well as accepting PDOs can be synchronized throughout the network, ("synchronous PDOs"). The assignment of application objects to a particular PDO (transfer object) can be set using a structure description ("PDO-Mapping") saved in the OD. This means it can be adapted to the particular application requirements of a specific device.

SDOs are transferred as acknowledged data transfer between two CAN objects in the form of a peer-to-peer coupling between two network nodes. The object directory entry is addressed by specifying the index and subindex of the OD entry. When SDO messages are transferred, this involves an additional overhead.

Standardized, event-orientated alarm messages ("Emergency_Messages") with a high priority are provided for messages associated with device faults/errors.

The functionality required to prepare and to start a distributed automation system in a coordinated fashion corresponds to the mechanisms defined under CAL network management (NMT) as well as the "Node-Guarding" principles or heartbeat which are used as basis for cyclic node monitoring.

CAN message identifiers can be allocated to PDOs and SDOs by directly entering identifiers in the data structures of the object directory or, for simple system structures, by using pre-defined identifiers.

General information

MICROMASTER, in conjunction with the CANopen option module, also provides, for the various MICROMASTER types, different CANopen modes from the DSP 402 profile (Drives and Motion Control).

For MICROMASTER 420/430), only the Velocity Mode is available. The MICROMASTER 440 supports the Velocity Mode and Profile Torque Mode modes.

When transferring net data via CANopen, a differentiation is made between process data (PDO) and parameter data (SDO).

In order to transfer process data, CAN telegrams can be set using so-called PDO mapping. After being powered-up, MICROMASTER has the following default settings:

- Receive PDO1 6040H control_word
- Receive PDO5 not mapped
- Receive PDO6 not mapped
- Send PDO1 6041H status_word
- Send PDO5 not mapped
- Send PDO6 not mapped

The mapping for PDO1 is fixed. Free PDO mapping in compliance with CANopen is not possible using MICROMASTER for this PDO.
The mapping for PDO5 and PDO6 can be modified using CANopen.

The mapping is subject to several restrictions that are described in more detail in Section 5.4.

The COB IDs of the PDOs can be changed by changing the node address or by writing into the communication objects. If the COB ID is changed in a communications object, then this is only saved in the volatile memory. This means that if the MICROMASTER drive unit is isolated from the supply voltage, these settings are lost and, after run-up, must be re-set via the bus. If you wish to permanently save the change to the COBID, then this must be written via the CANopen object 1010H (refer to Section 3.1).

An object of the object directory is accessed via the SDO channel. However, the communication objects (DS301) and the profile-specific objects of the drive profile (DSP402) are saved in the object directory of the CANopen option module. In order to be able to access additional objects in MICROMASTER, there is an SDO -> parameter mechanism. In this case, it does not involve CANopen objects, as the numbering of the sub-indices differs with respect to CAN-open (subindex 0 does not contain the number of objects, but already the first object). However, the objects from the parameter channel can, just like CANopen objects, be read and written using SDO access operations - they are designated with a * in the following.

- **Summary of Data transfer via CANopen**

  The messages associated with the CANopen option module can be sub-divided into three major areas:

  - Network management – this means starting and stopping the network as well as cyclic node monitoring (using NMT commands)
  - Process data – this means control words, setpoints, status information and actual values (generally use PDO messages)
  - Object data in order to read/write objects of the object directory or parameter values of the drive converter/drive inverter (using SDO messages)

3 Data transfer using CANopen

3.1 MICROMASTER object directory

All of the objects of the object directory are listed in the following table. The table contains the index and subindex of the object as well as a brief description of the functionality and the data type as well as the default value.

For each object, it is defined as to how it can be transferred. In this case, a differentiation is made between the two possibilities - SDO and PDO transfer. In addition, the parameters or connectors, in which the object data are located, are also listed. If both possibilities are specified in the table, then data transfer can either be realized as SDO or as PDO.

For more detailed information about the objects, please refer to the CANopen profiles DS 301 V4.0 and DSP 402 V1.1.

Comment to the following table:

O-Ind = Object index
S-Ind = Subindex
Access ro = read only
Access wo =write only
Access rw = read / write

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
1000H		Device Type	Device type	SDO	U32 / ro	10129H for MICROMASTER / -
1001H		Error Register	One bit for each type of error as defined for CANopen	SDO	U8 / ro	0 / bit 0 generic error bit 1 current bit 2 voltage bit 3 temperature bit 4 communication error bit 5 not used bit 6 reserved bit 7 other errors
1003H		Pre-defined error field	Display, parameters for the error code	SDO		
	.0	Number of errors	Number of errors which have occurred	SDO		0 / -
	.1	Standard error field	Error code of the last error which occurred	SDO		0 / This value is not the same as in MICROMASTER r0947 (refer to Table 3-14)
	.2	Standard error field	Error code of the last but one error which occurred	SDO		0 / -

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.3	Standard error field	Error code of the previous error which occurred	SDO		0 / -
1005H		COB-ID SYNC Message	Identifier of the SYNC telegram	SDO	U32 / rw	COB_ID = 8000080H / Changes can be permanently saved in the MICROMASTER using object 1010H
1006H		Communication cycle period	Time between two SYNC signals	SDO	U32 / rw	0 / Is activated by inputting a value > 0 (unit µs) and after receipt of the first SYNC signal. Triggers an error telegram "Synchronisation lost“ after 1.5 x the value in 1006H. The MICROMASTER's reaction to this error is the same as that which was set by the Guarding node.
1008H		Manufacturer Device Name	Manufacturer, device name	SDO		MICROMASTER 4 / -
100AH		Manufacturer Software Version	Software version of the CANopen module	SDO	U32 / ro	- / Example: 102 = SW1.02
100CH		Guard Time	Time between two guarding telegrams	SDO	U16 / rw	10 ms / Object 27F8* contains the product of object 100CH x 100DH
100DH		Life Time Factor	Number of permissible guarding telegram failures until a life time event is received	SDO	U8 / rw	2 / Object 27F8* contains the product of object 100CH x 100DH
1010H		Store parameters	Object to save the values written into the RAM.			
	.0	Number of entries	No. of sub-indices	SDO	U32 / ro	1 / -
	.1	Save all parameters	All objects are saved in the non-volatile memory using the instruction "save" 65766173H	SDO	U32 / rw	200E*/ In addition, the following can be set using parameter 200E*: 200E*=1 Subindex 1 = 00000003H (device saves parameters autonomously and objects which have not yet been saved can, if necessary, be stored with the "save" command) Parameter 200E* = 0, subindex 1 = 00000001H (device only saves parameters on command)
1011H		Restore parameters	Object to reset (restore) all objects to their factory settings (only in RAM).	SDO		
	.0	Number of entries	No. of sub-indices	SDO	U32 / ro	1 / -

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.1	Restore all default parameters	All of the parameters are reset to the factory setting using the command "load" 64616F6CH.	SDO	U32 / rw	1 / / Device sets values back to the factory setting..
1014H		COB-ID Emergency Message	Identifier for EMERGENCY messages	SDO	U32 / rw	80H + node-ID / Changes can be permanently set in MICROMASTER using object 1010H.
1015H		Inhibit Time Emergency Message	Delay time for error telegrams	SDO	U16 / rw	0 / Changes can be permanently set in MICROMASTER using object 1010H.
1017H		Producer Heartbeat time	This is the cycle time in which the MICROMASTER sends Heartbeat telegrams	SDO	U16 / rw	0 / Changes can be permanently set in MICROMASTER using object 1010H.
1018		Identity Object	Identification object			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	4 / -
	.1	Vendor-ID	CiA assigns the manufacturer's (vendor) number	SDO	U32 / ro	6000053H / Siemens A&D Drives
	.2	Product code	Drive converter/inverter type	SDO	U32 / ro	- / Value from r0203
	.3	Revision number	Revision number with software version number of the CANopen option module and CANopen software stack number	SDO	U32 / ro	- / Low word = CAN module Software Version, e.g. 205H = 2.05. High byte = CANopen Software stack Version, e.g. 43H = 4.3
	.4	Serial number	Serial number of the option module	SDO	U32 / ro	- / -
1029H		Error Behavior	Behavior of the communication status engine in case of error	SDO		
	.0	No. Of error classes	Number of error classes	SDO	U8 / ro	1 / -
	.1	Communication error	Behavior in case of a communication error	SDO	U8 / rw	- / 0 Pre-Operational (only if current state is operational) 1 no state change 2 Pre-Operational with OFF2 (pulses disabled)
1200H		SDO-Parameters	Identifier of the SDO telegram			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	2 / -

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.1	COB-ID Client>Server	Identifier of the SDO task (CANopen master to MICROMASTER)	SDO	U32 / ro	600H + node-ID / The default node number is 3
	.2	COB-ID Server>Client	Identifier of the SDO response (MICROMASTER to CANopen master)	SDO	U32 / ro	580H + node-ID / The default node number is 3
1400H		Receive PDO Communication Parameters	Setting parameter of the data transfer properties of the receive PDOs 1			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	2 / -
	.1	COB-ID PDO	Identifier of the PDOs	SDO	U32 / rw	200H + node-ID / Changes can be permanently set in MICROMASTER using object 1010H.
	.2	Transmission Type	Setting parameter of the data transfer type	SDO	U8 / rw	0 / (value also corresponds to object 27F9*.01 bit 8) Caution! After power-up, under certain circumstances, the value of this CAN parameter does not correspond to the value in the converter. After powering-up, this parameter should always be set again to the required value.
1404H		Receive PDO Communication Parameters	Setting parameter of the data transfer properties of the receive PDOs 5			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	2 / -
	.1	COB-ID PDO	Identifier of the PDOs	SDO	U32 / rw	300H + node-I / Changes can be permanently set in MICROMASTER using object 1010H.
	.2	Transmission Type	Setting parameter of the data transfer type	SDO	U32 / rw	0 / (value also corresponds to object 27F9*.01 bit 9) Caution! After power-up, under certain circumstances, the value of this CAN parameter does not correspond to the value in the converter. After powering-up, this parameter should always be set again to the required value.
1405H		Receive PDO Communication Parameters	Setting parameter of the data transfer properties of the receive PDOs 6			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	2 / -
	.1	COB-ID PDO	Identifier of the PDOs	SDO	U32 / rw	400H + node-ID / Changes can be permanently set in MICROMASTER using object 1010H.

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.2	Transmission Type	Setting parameter of the data transfer type	SDO	U8 / rw	0 / (value also corresponds to object 27F9*.01 bit 10) Caution! After power-up, under certain circumstances, the value of this CAN parameter does not correspond to the value in the converter. After powering-up, this parameter should always be set again to the required value.
1600H		Receive PDO Mapping Parameters	Object to enter the mapped objects into receive PDO1			
	.0	Number of mapped objects in PDO	No. of the mapped objects	SDO	U8 / ro	1 / Mapping cannot be changed
	.1	First mapped object	First mapped object. The object is permanently mapped in the PDO.	SDO	U32 / ro	60400010H / Control word permanently mapped.
1604H		Receive PDO Mapping Parameters	Object to enter the mapped objects into receive PDO5			
	.0	Number of mapped objects in PDO	No. of the mapped objects	SDO	U8 / rw	0 / If the mapping is to be changed via the CAN bus, subindex 0 according to CANopen profile DS 3.01 must first be set to 0.
	.1	First mapped object	First mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the low byte of parameter 27F9.03 H)
	.2	Second mapped object	Second mapped object	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the low byte of parameter 27F9.03 H)
	.3	Third mapped object	Third mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the low byte of parameter 27F9.03 H)
	4.	Fourth mapped object	Fourth mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the low byte of parameter 27F9.03 H)
1605H		Receive PDO Mapping Parameters	Object to enter the mapped objects into receive PDO6			
	.0	Number of mapped objects in PDO	No. of the mapped objects	SDO	U8 / rw	0 / If the mapping is to be changed via CAN bus, then the subindex 0 according to the CANopen profile DS 4.01 must first be set to 0.

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.1	First mapped object	First mapped object.	SDO	U32 / rw	0 / Mapping can be changed via CANopen (can also be set using the low byte of parameter 27F9.04 H)
	.2	Second mapped object	Second mapped object.	SDO	U32 / rw	0 / Mapping can be changed via CANopen (can also be set using the low byte of parameter 27F9.04 H)
	.3	Third mapped object	Third mapped object.	SDO	U32 / rw	0 / Mapping can be changed via CANopen (can also be set using the low byte of parameter 27F9.04 H)
	4.	Fourth mapped object	Fourth mapped object	SDO	U32 / rw	0 / Mapping can be changed via CANopen (can also be set using the low byte of parameter 27F9.04 H)
1800H		Transmit PDO Communication Parameters	Setting parameter for the data transfer properties of the transmit PDOs 1			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	2 / -
	.1	COB-ID PDO	Identifier of the PDOs	SDO	U32 / rw	180H + node-ID / Changes can be permanently set in MICROMASTER using object 1010H.
	.2	Transmission Type	Setting parameter for the data transfer type	SDO	U8 / rw	0 / (can also be set using the low byte of parameter 27F9.00 H)
	.3	Inhibit time	Delay time	SDO	U16 / rw	0 / Changes can be permanently set in MICROMASTER using object 1010H.
	.4	Reserved	Not used	SDO	U8 / ro	0 / -
	.5	Event timer	Timer after which the PDO is always sent even if the value has not changed	SDO	U16 / rw	0 / Changes can be permanently set in MICROMASTER using object 1010H.
1804H		Transmit PDO Communication Parameters	Setting parameter of the data transfer properties of transmit PDOs 5			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	2 / -
	.1	COB-ID PDO	Identifier of the PDOs	SDO	U32 / rw	280H + node-ID / Changes can be permanently set in MICROMASTER using object 1010H.

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.2	Transmission Type	Setting parameter for the data transfer type	SDO	U8 / rw	0 / (can also be set using the high byte of parameter 27F9.00 H) Caution! After power-up, under certain circumstances, the value of this CAN parameter does not correspond to the value in the converter. After powering-up, this parameter should always be set again to the required value.
	.3	Inhibit time	Delay time	SDO	U16 / rw	0 / Changes can be permanently set in MICROMASTER using object 1010H.
	.4	Reserved	Not used	SDO	U8 / ro	0 / -
	.5	Event timer	Timer after which the PDO is always sent even if the value has not changed	SDO	U16 / rw	0 / Changes can be permanently set in MICROMASTER using object 1010H.
1805H		Transmit PDO Communication Parameters	Setting parameter of the data transfer properties of transmit PDOs 6			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	2 / -
	.1	COB-ID PDO	Identifier of the PDOs	SDO	U32 / rw	380H + node-ID / Changes can be permanently set in MICROMASTER using object 1010H.
	.2	Transmission Type	Setting parameter for the data transfer type	SDO	U8 / rw	0 / (can also be set using the low byte of parameter 27F9.01 H) Caution! After power-up, under certain circumstances, the value of this CAN parameter does not correspond to the value in the converter. After powering-up, this parameter should always be set again to the required value.
	.3	Inhibit time	Delay time	SDO	U16 / rw	0 / Changes can be permanently set in MICROMASTER using object 1010H.
	.4	Reserved	Not used	SDO	U8 / ro	0 / -
	.5	Event timer	Timer after which the PDO is always sent even if the value has not changed	SDO	U16 / rw	0 / Changes can be permanently set in MICROMASTER using object 1010H.

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
1A00H		Transmit PDO Mapping Parameters	Object to enter mapped objects into transmit PDO1			
	.0	Number of mapped objects in PDO	No. of the mapped objects	SDO	U8 / ro	1 / Mapping cannot be changed
	.1	First mapped object	First mapped object. The object is permanently mapped in the PDO.	SDO	U32 / ro	60410010H / Status word, permanently mapped.
1A04 H		Transmit PDO Mapping Parameters	Object to enter mapped objects into transmit PDO5			
	.0	Number of mapped objects in PDO	No. of the mapped objects	SDO	U8 / rw	2 / If the mapping is changed via the CAN bus, then subindex 0 according to the CANopen profile DS 4.01 must be first be set to 0.
	.1	First mapped object	First mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the high byte of parameter 27F9.02 H)
	.2	Second mapped object	Second mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the high byte of parameter 27F9.02 H)
	.3	Third mapped object	Third mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the high byte of parameter 27F9.02 H)
	4.	Fourth mapped object	Fourth mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the high byte of parameter 27F9.02 H)
1A05 H		Transmit PDO Mapping Parameters	Object to enter mapped objects in transmit PDO6			
	.0	Number of mapped objects in PDO	No. of the mapped objects	SDO	U8 / rw	2 / If the mapping is changed via the CAN bus, then subindex 0 according to the CANopen profile DS 4.01 must be first be set to 0.
	.1	First mapped object	First mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the high byte of parameter 27F9.03 H)
	.2	Second mapped object	Second mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the high byte of parameter 27F9.03 H)

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.3	Third mapped object	Third mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the high byte of parameter 27F9.03 H)
	4.	Fourth mapped object	Fourth mapped object.	SDO	U32 / rw	0H / Mapping can be changed via CANopen (can also be set using the high byte of parameter 27F9.03 H)
2000H * - 2801H *		SDO->Parameter Channel	Objects of the SDO -> parameter channel	SDO		- / Using these objects, it is possible to access all of the parameters in the MICROMASTER according to the formula MICROMASTER parameter number in hex + 2000H = SDO number. In this case, it does not involve CANopen objects, as the numbering of the sub-indices differs with respect to CAN-open (subindex 0 does not contain the number of objects, but already the first object). However, the objects from the parameter channel can, just like CANopen objects, be read and written using SDO access operations - they are designated with a * in the following.
2396H *		Node ID	ID or address of the MICROMASTER	SDO	U16 / rw	These objects set the NodeID of the MICROMASTER. If this value is changed on the CAN bus it takes effect after a reset communication or reset node network command.
27DAH *		Baudrate	Baudrate	SDO	U16 / rw	10 / Baud rate in kbit/s
2802H *		Free Objects into drive	Free objects, with which non-defined process data can be sent from the CANopen master to the MICROMASTER			These objects directly write process data into parameters P2050.2, P2050.3 and P2050.5. The gaps in the sub-indices are due to the mirroring of the object directly to parameter P2050 (PZD from CB). All other receive channels of the P2050 are occupied with the objects of the device profile. An error message is output if an attempt is made to map other sub-indices or to read/write using an SDO.
	.3	Free Object into drive	Free object which can be used to send non-defined process data to the MICROMASTER P2050.2	SDO/ PDO	I16 / rw	0 / Values are directly written into P2050.2.
	.4	Free Object into drive	Free object which can be used to send non-defined process data to the MICROMASTER P2050.3	SDO/ PDO	I16 / rw	0 / Values are directly written into P2050.3.

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.6	Free Object into drive (only MICROMASTER 440)	Free object which can be used to send non-defined process data to the MICROMASTER P2050.5	SDO/ PDO	I16 / rw	0 / Values are directly written into P2050.5.
2803H *		Free Objects from drive	Free objects, with which non-defined process data can be sent from the CANopen master to the MICROMASTER.			-/ These objects read process data from the parameters which are connected with P2051.2 and P2051.3 via BICO and transmit these on the CAN bus. The gaps in the sub-indices are due to the mirroring of the object directly onto parameter P2051. All other send channels of the P2051 are occupied with the objects of the device profile. An error message is output if an attempt is made to map the other sub-indices. A zero is displayed when reading-out the other sub-indices.
	.3	Free Object from drive	Free object which can be used to send non-defined process data to the MICROMASTER P2051.2.	SDO/ PDO	I16 / rw	0 / Values are directly sent from parameter BICO connected to P2051.2.
	.4	Free Object from drive	Free object which can be used to send non-defined process data to the MICROMASTER P2051.3.	SDO/ PDO	I16 / rw	0 / Values are directly sent from parameter BICO connected to P2051.3.
2804H * - 5FFF H*		SDO->Parameter Channel	Objects of the SDO-> parameter channel	SDO		- / Using these objects, it is possible to access all of the parameters in the MICROMASTER according to the formula MICROMASTER parameter number in hex + 2000H = SDO number. In this case, it does not involve CANopen objects, as the numbering of the sub-indices differs with respect to CAN-open (subindex 0 does not contain the number of objects, but already the first object). However, the objects from the parameter channel can, just like CANopen objects, be read and written using SDO access operations - they are designated with a * in the following.
6007H		Abort connection option code	Behaviour of the MICROMASTER-status engine in case of communication errors	SDO	I16 / rw	0 / 0 no action 1 malfunction 2 Device control command “disable voltage” 3 Device control command “quick stop” 4 ramped stop
603FH		Error code	The MICROMASTER error from parameter P0947 is displayed in this object	SDO	U16 / ro	0 / -

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
6040H		Control word	CANopen control word	SDO/ PDO	U16 / wo	0 / This is converted and mapped to the MICROMASTER control word 1 (r0054) when being transferred to the MICROMASTER. Parameter 22BCH* must be set to 6 so that the CANopen control word can control the MICROMASTER.
6041H		Status word	CANopen status word	SDO/ PDO	U16 / ro	0 / MICROMASTER status word 1 (r0052) is converted when being transferred from MICROMASTER and is mapped onto the CANopen status word. Parameter 22BCH* must be set to 6 so that the CANopen status word can send MICROMASTER status onto the CAN bus.
6042H		Vl_target_velocity	Reference (target) velocity of the velocity mode	SDO/ PDO	I16 / rw	0 RPM / Parameter 23E8H* must first be set to 6 (via CAN), or a BICO connection must be established from P2050.1 to the speed setpoint. Up until now, this BICO connection can only be established with the BOP or using the start-up tool.
6043H		Vl_velocity_demand	Velocity, controlled quantity	SDO/ PDO	I16 / ro	0 RPM / Parameter 23E8H* must first be set to 6 (via CAN) or a BICO connection must be established from r0021 to P2051.1. Up until now, this BICO connection can only be established with the BOP or using the start-up tool.
6044H ❶		Vl_control_effort	Velocity actual value	SDO/ PDO	I16 / ro	0 RPM / A BICO connection must be established from r0063 to P2051. Up until now, this BICO connection can only be established with the BOP or using the start-up tool.
6046H		Vl_velocity_min_max_amount	Minimum and maximum absolute value for the velocity limiting			
	.0	Number of entries	No. of sub-indices	SDO	U8 / ro	2 / -
	.1	Vl_velocity_min_amount	Absolute value of the minimum velocity limiting	SDO	U32 / rw	0 / (P1080 · 60 / r0313) RPM / Acts on P1080 (2438H*). When this object changes, this effects the gradient (rate-of-rise) of the up ramp.

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
	.2	Vl_velocity_max_amount	Absolute value of the maximum velocity limiting	SDO	U32 / rw	1500 / (P1082 · 60 / r0313) RPM / Acts on P1082 (243AH*). When this object changes, this effects the gradient (rate-of-fall) of the down ramp.
6048H		Vl_velocity_acceleration	Acceleration			
	.1	Delta_speed	Velocity change which is used to calculate the acceleration	SDO	U32 / rw	15 000/ RPM in 100 s / When reading-out the object the value in subindex 1 is calculated using P1120, P1082 and the pole pair number according to the following formula: $\Delta n = \frac{P1082 \cdot 100\,s \cdot 60\,s}{P1120 \cdot r0313}$ After each power-up, this subindex is set to the speed at which the drive rotates in 100 s. If subindex 1 or 2 are changed, then this effects P1120. (2070H*)
	.2	Delta_time	Time change which is used to calculate the acceleration	SDO	U16 / rw	100 s / After every power-up, this subindex is set to the default value of 100 s.
6049H		Vl_velocity_deceleration	Deceleration			
	.1	Delta_speed	Velocity change which is used to calculate the deceleration	SDO	U32 / rw	15 000/ RPM in 100 s / When reading-out the object the value in subindex 1 is calculated using P1121, P1082 and the pole pair number according to the following formula: $\Delta n = \frac{P1082 \cdot 100\,s \cdot 60\,s}{P1121 \cdot r0313}$ After each power-up, this subindex is set to the speed at which the drive rotates in 100 s. If subindex 1 or 2 are changed, then this effects P1121 (2071H*)
	.2	Delta_time	Time change which is used to calculate the deceleration	SDO	U16 / rw	100 s / After every power-up, this subindex is set to the default value of 100 s.
604EH		Vl_velocity_reference	Reference velocity	SDO	U32 / rw	1500/ $\frac{P2000 \cdot 60}{r0313}$ / Using this object, the reference velocity to convert the process quantities is set. This object writes or reads from P2000 (27D0H*), converted to RPM.

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
6052H		Vl_nominal _percentage	Velocity setpoint as a percentage	SDO/ PDO	I16 / rw	0 / The velocity is entered as a percentage referred to 604EH (P2000). 4000H = 100 % Parameter 23E8H* must first be set to 6 (via CAN) or a BICO connection must be established from P2050.1 to the speed setpoint. Up until now, this BICO connection can only be established with the BOP or using the start-up tool.
6053H		Vl_percentag e _demand	Setpoint generator output as a percentage	SDO/ PDO	I16 / rw	0 / Velocity controlled variable as a % referred to 604EH (P2000) or: 4000H = 100 %. Must be connected from r0021 to P2051.1 through a BICO connection.
6054H ❶		Vl_actual _percentage	Actual velocity as a %	SDO/ PDO	I16 / rw	0 / Actual velocity as a % referred to 604EH (P2000). 4000H = 100 %. A connection must be made from r0063 to P2051.5 through a BICO connection. Up until now, this BICO connection can only be established with the BOP or using the start-up tool.
6060H		Modes_of_ operation	Object to select the operating mode.	SDO/ PDO	I8 / wo	2 / MICROMASTER 420/430 only supports the Velocity Mode. MICROMASTER 440 can support the Velocity Mode and the Profile Torque Mode. (It is possible to toggle, for MICROMASTER 440, between the Profile Torque Mode and Velocity Mode by setting or resetting bit 12 in r2091.) P1501 must be connected to r2091.12 through a BICO connection.
6061H		Modes_of _operation _display	Object to display the operating mode	SDO/ PDO	I8 / ro	2 / Displays the mode selected in object 6060H. (This is realized by reading-out bit 12 of parameter P2051.6 (r2091). Must be connected using a BICO connection from P2051.06 with r0055. Up until now, this BICO connection can only be established with the BOP or using the start-up tool.
6071H ❶		Target_torqu e	Torque setpoint	SDO/ PDO	I16 / rw	0 % / This is entered as a per mille of the rated motor torque. Must be connected from r2050.04 to the torque setpoint through a BICO connection. Must do BICO connection P2050.04 = P1503.

O-Ind	S-Ind	Object name	Description	Transf. using	D-type / access	Default value / comment
6076H		Motor rated torque	Used to define actual torque equivalent to 1000 in objects 6071H and 6077H	SDO	U32 / rw	?/ Corresponds to 27D3H*(P2003) / (1000 = 1 Nm) note that by default P2003 = 2 x nominal torque so therefore by default a target torque of 1000 in object 6071H will give 2 x nominal torque.
6077H ❶		Actual_torque	Actual torque	SDO/ PDO	I16 / ro	0 % / Displays the actual torque as per mille of the rated motor torque. r0080 must be connected to P2051.04 through a BICO connection. Up until now, this BICO connection can only be established with the BOP or using the start-up tool.
6087H ❷		Torque_slope	Torque change as a per mille of the rated torque per s	SDO	U32 / rw	- / This object is used to enter the change of the torque as a per mille of the rated torque per s. In order to use this object, several BICO connections must be set. Refer to Section 6.3.2.
6088H ❷		Torque_profile _type	Selecting the ramp shape of the torque profile	SDO	I16 / rw	- / This object is used to enter the ramp shape which is then used to transfer a new torque setpoint to the torque controller. In order to use this object, several BICO connections must be set. Refer to Section 6.3.2.

❶ Object not available in MICROMASTER 420/430.

❷ Additional BICO connections must be established in order to use the object.

NOTE

CANopen will always write to the active data set. If the drive data set should change, then relevant objects in the object dictionary will be updated from the new data set, and the stored powerup values of parameters will be updated from the new data set.

(If the contents of parameters r0050 or r0051 are changed, the objects are updated automatically.)

3.2 CANopen communication services

3.2.1 NMT (Network Management) services

The Network Management is structured according to nodes and follows a master-slave structure.

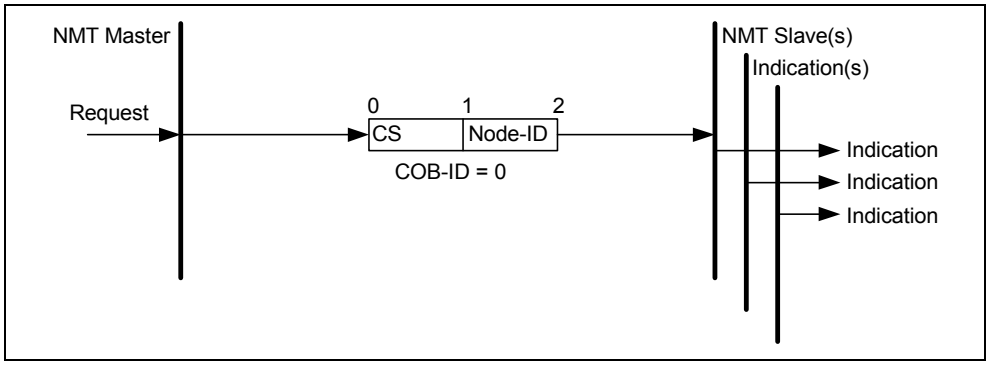


Fig. 3-1 NMT protocol

Nodes are initialized, started, monitored, reset or stopped using the NMT services. MICROMASTER is an NMT slave.
The following sketch shows the switched communications network of the option module CANopen.

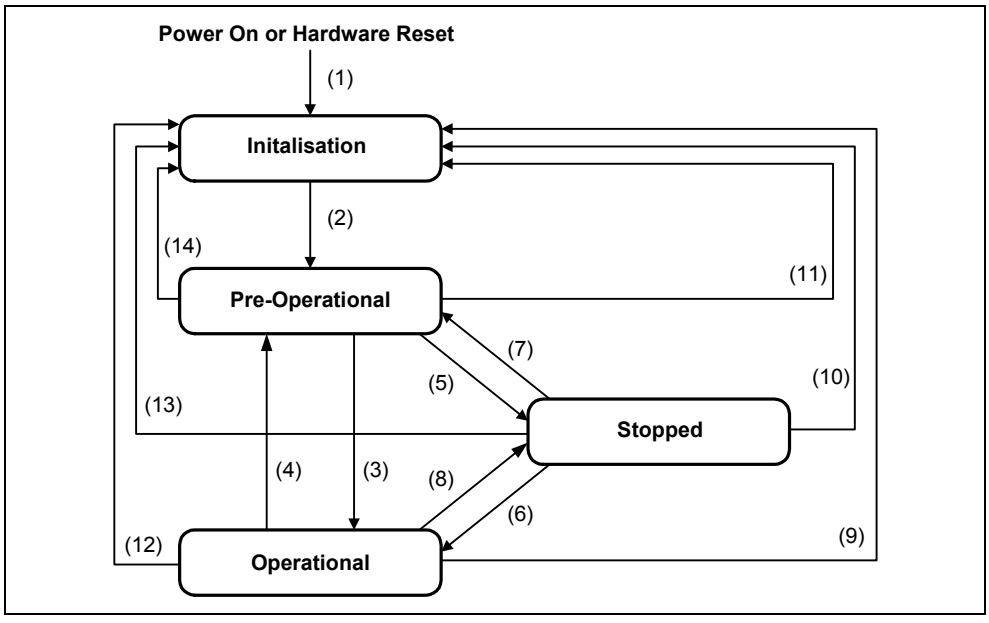


Fig. 3-2 CANopen switched communications network

Transition points of the switched communications network

Start Remote Node
This is an instruction to transition from the Pre-Operational to Operational communications state. The drive can only send and receive process data when it is in the Operational state.

Stop Remote Node
This is an instruction to transition from Pre-Operational into Stopped or Operational into Stopped. In the Stopped state, the nodes can only process NMT instructions.

Enter Pre-Operational
This is an instruction to transition from Operational or Stopped into Pre-Operational. In the Pre-Operational state, the node cannot process any PDOs. However, it can be parameterized or operated via SDOs.

Reset Node❶
This is an instruction to transition from Operational, Pre-Operational or Stopped to Initialization. The Reset Node instruction resets all objects (1000H - 9FFFH) into the state after power-up. This instruction/command is used, e.g. after changing-over the NodeID in order to make the new setting effective.

Reset Communication❷
This is an instruction to transition from Operational, Pre-Operational or Stopped to Initialization. After the Reset Communication instruction, all communication objects (1000H - 1FFFH) are reset into the state after power-on.

Table 3-1 Transitions of the switched communication network

(1)	After Power On, the MICROMASTER and the CANopen option module automatically go into the initialized state
(2)	After initialization, transition to PRE-OPERATIONAL
(3), (6)	Start_Remote_Node instruction (CS = 1)
(4), (7)	Enter_PRE-OPERATIONAL_State instruction (CS = 128)
(5), (8)	Stop_Remote_Node instruction (CS = 2)
(9), (10), (11)	Reset_Node instruction (CS = 129)
(12), (13), (14)	Reset_Communication instruction (CS = 130)

❶ With a Reset Node instruction, all objects of the node are reset to the state after power-up. For MICROMASTER 420/430/440, this function is only available with some restrictions. MICROMASTER only resets the objects from the communications profile (1000H - 1FFFH) as well as from the device profile (6000H - 9FFFH). The Reset Node instruction can only be executed if the object changes were only saved in the RAM (P0014 = 0) and no "save" instruction was sent to object 1010H. As soon as changes were saved in the EEPROM, as a result of the settings of parameter P0014 or by writing into the object 1010H, the values, saved in the EEPROM for a Reset Node instruction, are set. The Reset Node is, also, acknowledged by an emergency message in order to inform the user of this fact.
After a data set change, it is also no longer possible to reset values of the initial data set back to their Power On values. After a Reset Node, the actual data set is reset to the values it had, when it was first changed. The command "load"

sent to object 1011H is available in CANopen to reset to the original Power On values (refer to Page 95 DS301 V4.02).

❷ For a Reset Communication instruction, all of the communication objects (1000H - 1FFFH) are reset to the state after power-up. The Reset Communication instruction can only be executed if the object changes were only saved in the RAM (P0014=0) and no "save" instruction was sent to object 1010H. As soon as changes were saved in the EEPROM, either due to the settings of parameter P0014, or by writing into the object 1010H, it is no longer possible to reset the Power On values. The Reset Communication instruction is then acknowledged using an EMERGENCY message. The command "load" sent to object 1011H is available in CANopen to reset to the original Power On values (refer to Page 95 DS301 V4.02).

Properties in the various communication states

In the various communication states, nodes can only be accessed via CANopen using specific communication services. Further, the nodes in the various states only send specific telegrams. This is clearly shown in the following table.

Table 3-2 CANopen services which are available in the various states

	INITIALIZING	PRE-OPERATIONAL	OPERATIONAL	STOPPED
PDO			X	
SDO		X	X	
Synchronization Object		X	X	
Emergency Object		X	X	
Boot-Up Object	X			
Network Management Object		X	X	X

3.2.2 Communication monitoring services

There are three communication monitoring services in CANopen:

- Node Guarding / Life Guarding
- Heartbeat
- Sync loss detection

Node Guarding:

MICROMASTER 420/430/440 supports Node Guarding and Life Guarding. With Node Guarding, the CANopen master sends, to each slave, an RTR telegram with the COB-ID 700H + node-ID. The slave responds, with the same COB-ID, with its communications state. This means either Pre-Operational, Operational or Stopped. Further, the message that is sent contains a toggle bit - using the Object Guard Time, the time between two RTR messages, received at the slave, is set.

Life Guarding:

The CANopen slave monitors the incoming RTR messages from the master. The number of RTR telegrams which can fail as a maximum before the slave initiates a Life Guarding event is defined using the Life Time Factor object. The lifetime of the master is calculated from the product of the Guard Time (refer to Node Guarding) and Life Time Factor. This is the maximum time that the slave waits for an RTR telegram (message) before it initiates a Life Guarding Event.

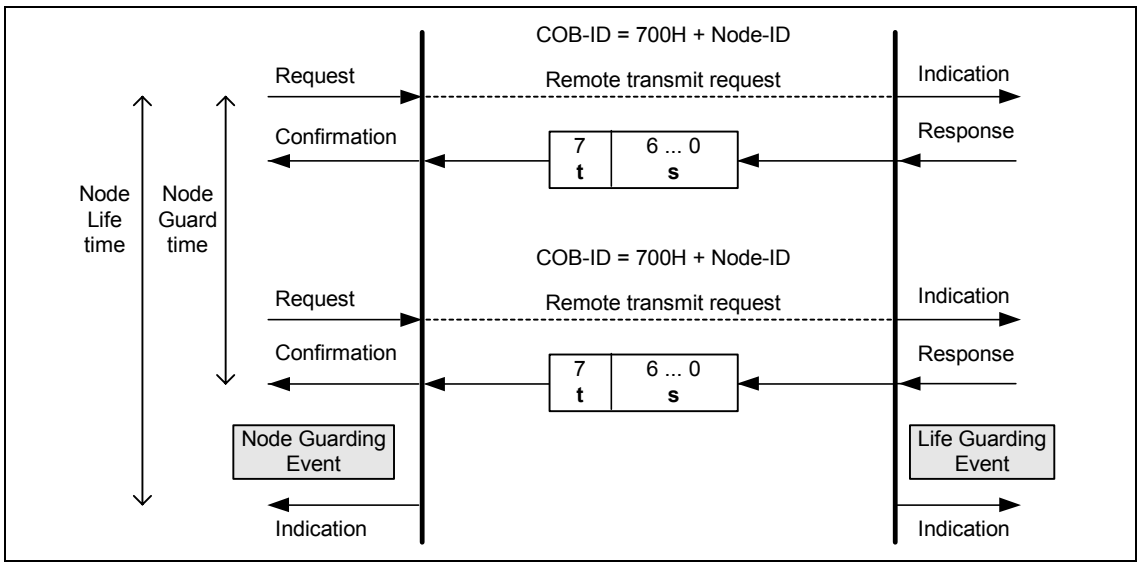


Fig. 3-3 Node Guarding/ Life Guarding protocols

Heartbeat:

The MICROMASTER 420/430/440 also supports the Heartbeat protocol as Heartbeat Producer. With the Heartbeat protocol, a Heartbeat Producer cyclically sends its NMT state to the CAN bus. The message does not contain a Toggle bit. The Heartbeat Producer object is used to define the time between two heartbeat messages. One or several Heartbeat Consumers expect the telegram of the Heartbeat Producer after the Heartbeat Consumer Time. If this message is not received, then the consumer initiates a Heartbeat event. The error message sent (EMCY) is the same as for Node Guarding/ Life Guarding.

NOTE

Only one communication monitoring service may be activated. This is either Node Guarding/Life Guarding or Heartbeat.

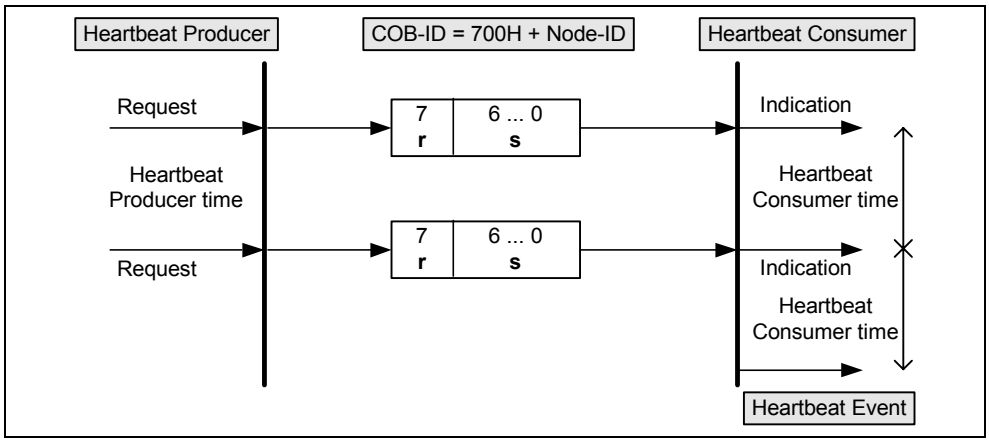


Fig. 3-4 Heartbeat protocol

SYNC Loss Detection:

The third communication monitoring is the SYNC loss detection.
The SYNC Producer sends cyclic SYNC telegram (this does not contain any data) with the SYNC producer time (object 1006H Communication cycle period) and the COB-ID 80H. The MICROMASTER 420/430/440, as SYNC consumer, monitors whether the SYNC telegram is sent in the SYNC Consumer time (object 1006H Communication cycle period multiplied by 1.5). If the SYNC producer doesn't send the SYNC telegram within the defined time (SYNC loss event), then MICROMASTER responds the same as for a Heartbeat- or a Life Guarding Event.

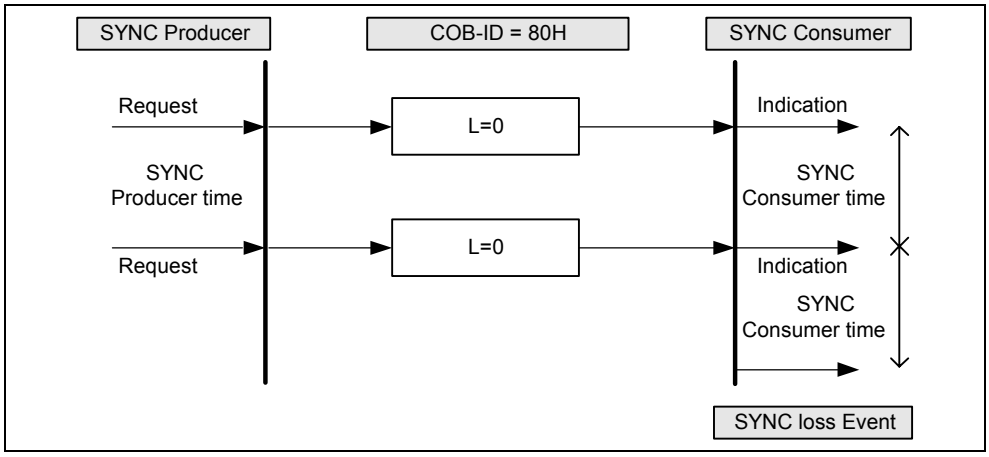


Fig. 3-5 SYNC monitoring protocol

NOTE

After powerup, Node Guarding and SYNC loss detection only become active after reception of the first Node-Guarding message and/or the first SYNC message.

The Node-Guarding or SYNC loss warning and the displayed LED error condition will be cleared immediately after the first Node-Guarding message or SYNC message was received. That means the first SYNC starts SYNC checking and the first Node-Guard message starts Node-Guarding.

If a Node-Guarding fault or a SYNC loss has caused the inverter to trip, then it is possible to reset the trip in the inverter via CAN. However, the inverter cannot go into run as long as a communications error is still present (Node-Guarding, SYNC, Busoff or Overrun).

3.2.3 Boot up telegram

After the initialization phase, a CANopen slave logs-on with a boot-up telegram. This telegrams contains the COB-ID 700 plus NODE-ID. The COB-ID can be taken from the node address of the slave. This allows a CANopen master to know which slaves are connected to the bus.

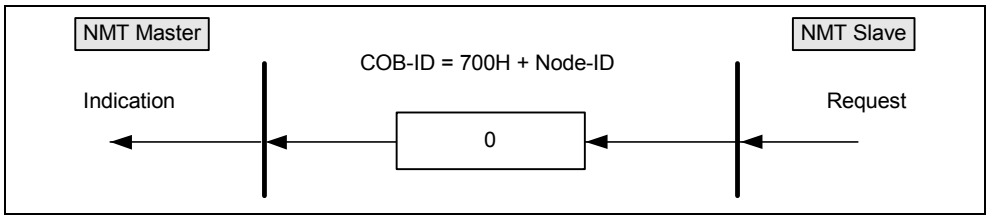


Fig. 3-6 Boot up protocol

3.2.4 SDO (Service Data Object) services

SDO services are used to access the object directory of the connected device. An SDO connection is a peer-to-peer coupling between an SDO client and a server. The MICROMASTER with its object directory is, in this sense, an SDO server. The identifiers of the Micromaster are defined according to CANopen. For communications between the client => server, then COB-ID 600H + node ID and for the server => client, COB-ID 580H + node ID apply.

Properties/features of SDOs

- Confirmed transfer of objects
- Data transfer/exchange is always asynchronous (non-synchronous)
- Values greater than 4 bytes can be transferred (normal transfer)
- Values not more than 4 bytes can be transferred (expedited transfer)

All variables of the drive can be addressed via SDO transfer.

Note

For MICROMASTER, SDO transfer is only possible by specifying the length of the data bytes contained. If another device uses the transfer mechanism without length data, then problems can occur.

Downloading SDO protocol

The download SDO protocol is used to write the values of the object directory into the drive. MICROMASTER 420/430/440 only supports the "expedited SDO download" for up to 4 bytes of data (e =1,s = 1).

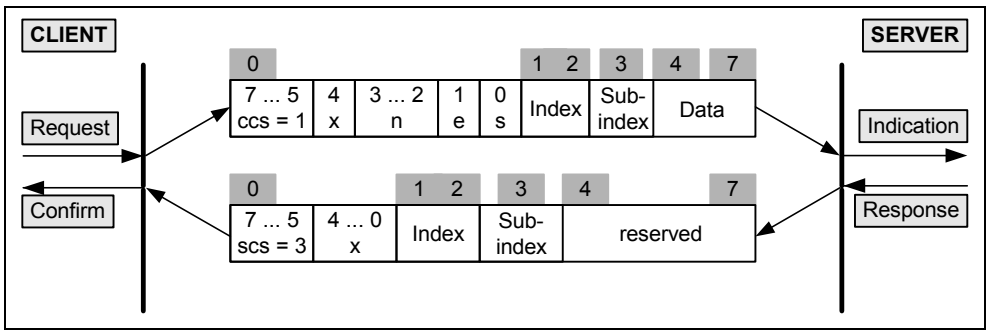


Fig. 3-7 Download SDO protocol

In order to be able to write parameters, a CANopen master must send a download protocol according to the following table

Table 3-3 Significance of the bytes in the SDO download protocol (CANopen master to MICROMASTER)

Byte 0:					**Bytes 1,2**	**Byte 3**	**Bytes 4,5**	**Bytes 6,7**
Bits 7-5 CCS = 1	Bit 4 = 0	Bits 3,2 n = 0 (Dword) n = 2 (Word)	e = 1	s = 1	Object Dictionary Index	Object Dictionary Subindex	Word data Dword data	Dword data

If the MICROMASTER 420/430/440 cannot correctly process the task, it acknowledges the download SDO protocol with a response telegram according to the following table:

Table 3-4 Significance of the bytes in the SDO download protocol (MICROMASTER to CANopen master)

Byte 0:		**Bytes 1,2**	**Byte 3**	**Bytes 4,5**	**Bytes 6,7**
Bits 7-5 SCS = 3	Bit 4 - 0 = 0	Object Dictionary Index	Object Dictionary Subindex	Reserved = 0	

The SDO download is either realized directly in the RAM or directly in the EEPROM. This can be controlled using parameter P0014 (object 200EH). If parameter P0014 (object 200EH) is set so that the download is saved in the RAM, data can be saved to object 1010H to save in the EEPROM using the "save" instruction.

Upload SDO protocol

The upload SDO protocol is used to read the values of the object directory of the drive. MICROMASTER 420/430/440 only supports the "expedited SDO upload" for up to 4 bytes of data (e =1,s = 1).

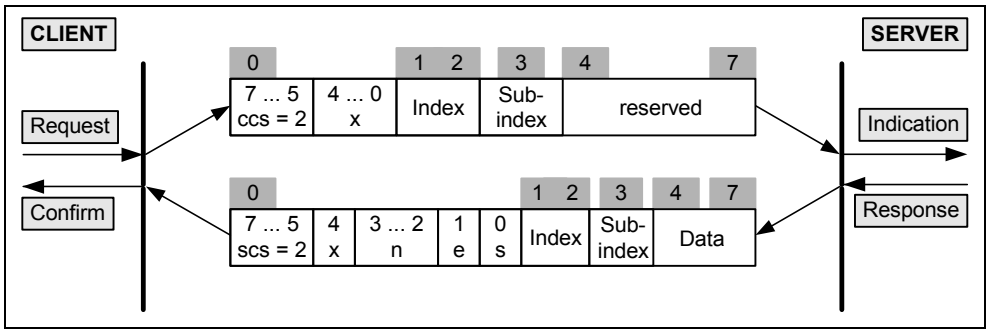


Fig. 3-8 Upload SDO protocol

In order to read objects from the MICROMASTER 420/430/440, a CANopen master must send an SDO upload protocol according to the following table.

Table 3-5 Significance of the bytes in the SDO upload protocol (CANopen master to MICROMASTER)

Byte 0: Control Word		**Bytes 1,2**	**Byte 3**	**Bytes 4,5**	**Bytes 6,7**
Bits 7-5 CCS = 2	Bit 4 - 0 = 0	Object Dictionary Index	Object Dictionary Subindex	Reserved = 0	

If the MICROMASTER 420/430/440 cannot correctly process the task, it acknowledges the upload SDO protocol using a response telegram according to the following table:

Table 3-6 Significance of the bytes in the SDO upload protocol (MICROMASTER to CANopen master)

Byte 0:					**Bytes 1,2**	**Byte 3**	**Bytes 4,5**	**Bytes 6,7**
Bits 7-5 SCS = 2	Bit 4 = 0	Bits 3,2 n = 0 (Dword) n = 2 (Word)	e = 1	s = 1	Object Dictionary Index	Object Dictionary Subindex	Word data Dword data	Dword data

Abort SDO transfer protocol

SDO tasks, which the MICROMASTER 420/430/440 cannot process are responded to using an abort SDO protocol. If the MICROMASTER does not respond in the expected time, the CANopen master sends an abort SDO protocol.

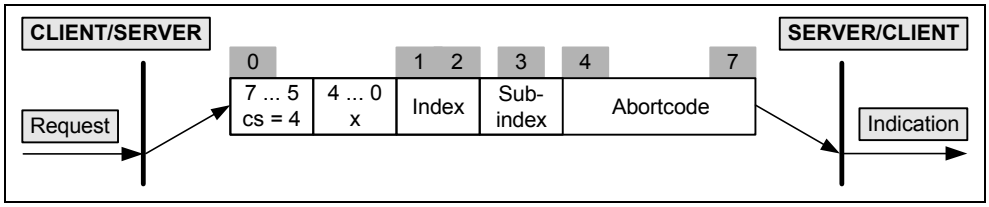


Fig. 3-9 Abort SDO transfer protocol

In this case, there are various abort codes. The abort codes, which occur in the MICROMASTER 420/430/440, are listed in the following table. Most of the faults occur due to communication errors between the CANopen option module and MICROMASTER. In order to be able to access the parameters in the MICROMASTER, SDO tasks must be re-coded into parameter tasks and then sent to the MICROMASTER via the SOL. If the MICROMASTER cannot correctly process the parameter task which it was issued, then it signals this in the form of a response to the option module. In turn, the option module transforms this response into the abort codes listed in the table. If the MICROMASTER sends an abort code, the parameter request issued must be re-checked and repeated.

Abort Name	Abort Code	Description
SDO protocol time overflow	05040000H	MICROMASTER had not responded within xxxs to a parameter task of the option module. The CANopen module cancels the request.
Illegal access	06010001H	This parameter can only be write accessed (write only) (parameter value cannot be read)
Illegal access	06010002H	Parameter is a read-only parameter (parameter value cannot be changed)
Object not available in the object directory	06020000H	Access to a non-existing object (parameter number does not exist)
General parameter incompatibility	06040043H	Illegal parameter value (the parameter only permits specific values)
Data type does not match the length of the SDO value, SDO value too high or too low	06070010H	Mixture between word and double word (incorrect data type)
Subindex does not exist	06090011H	SDO access to a non-existing subindex of an object (erroneous subindex, no array) For tasks from "indexed parameter to non-indexed parameter" type Example: Task: 'Change parameter value (word, array)' for non-indexed parameters
Parameter value exceeded	06090030H	Upper or lower limit value exceeded (write only)
General error	08000000H	Incorrect parameter status: Parameter, read or write buffer simultaneously used in both directions New parameter task is sent before the previous has been responded to.
Data transfer error	08000020H	Data cannot be transferred to the application or cannot be saved.
Data not able to be transferred or saved due to local control	08000021H	Change request without having status as higher-level control (refer to P0927, corresponds to object 0x239F*) Drive converter parameter: 'Access code' and/or 'Special parameter access' presently not set (the operator has no change authorization for the PKW interface)
Service error	08000022H	The drive converter does not permit the presently output task (the task cannot be executed due to the operating state)

Working with the SDO<->parameter channel

The CANopen option module offers an SDO->parameter channel, which can be used to read or write all parameters (PKW objects) of the MICROMASTER 420/430/440. In this case, it does not involve CANopen objects in the true sense. The reason for this is that the numbering of the sub-indices with respect to CANopen differs (subindex 0 does not contain the number of objects, but already the first object). However, the objects from the parameter channel can be read and written just like CANopen objects using SDO access operations. Their object number must be appropriately converted. They are designated by a * in this document.

You must proceed as follows to write into a MICROMASTER parameter via the SDO<->parameter channel:

Firstly, the parameter number of the PKW object, specified as a decimal number, must be converted into a hexadecimal value.

After this, 2000H must be added to the hexadecimal parameter value. This is then the object number that is used for the SDO access operation (reading or writing). The numerical sequence after the point in the specification of the PKW object generally corresponds to the subindex number of the CANopen object. It only has to be converted from a decimal number into a hexadecimal number. In order to be able to write into the parameter, in addition, the data type - e.g. unsigned 16 (U16) - must be taken from the MICROMASTER parameter list.

The following diagram again clearly shows how to work with the SDO parameter channel:

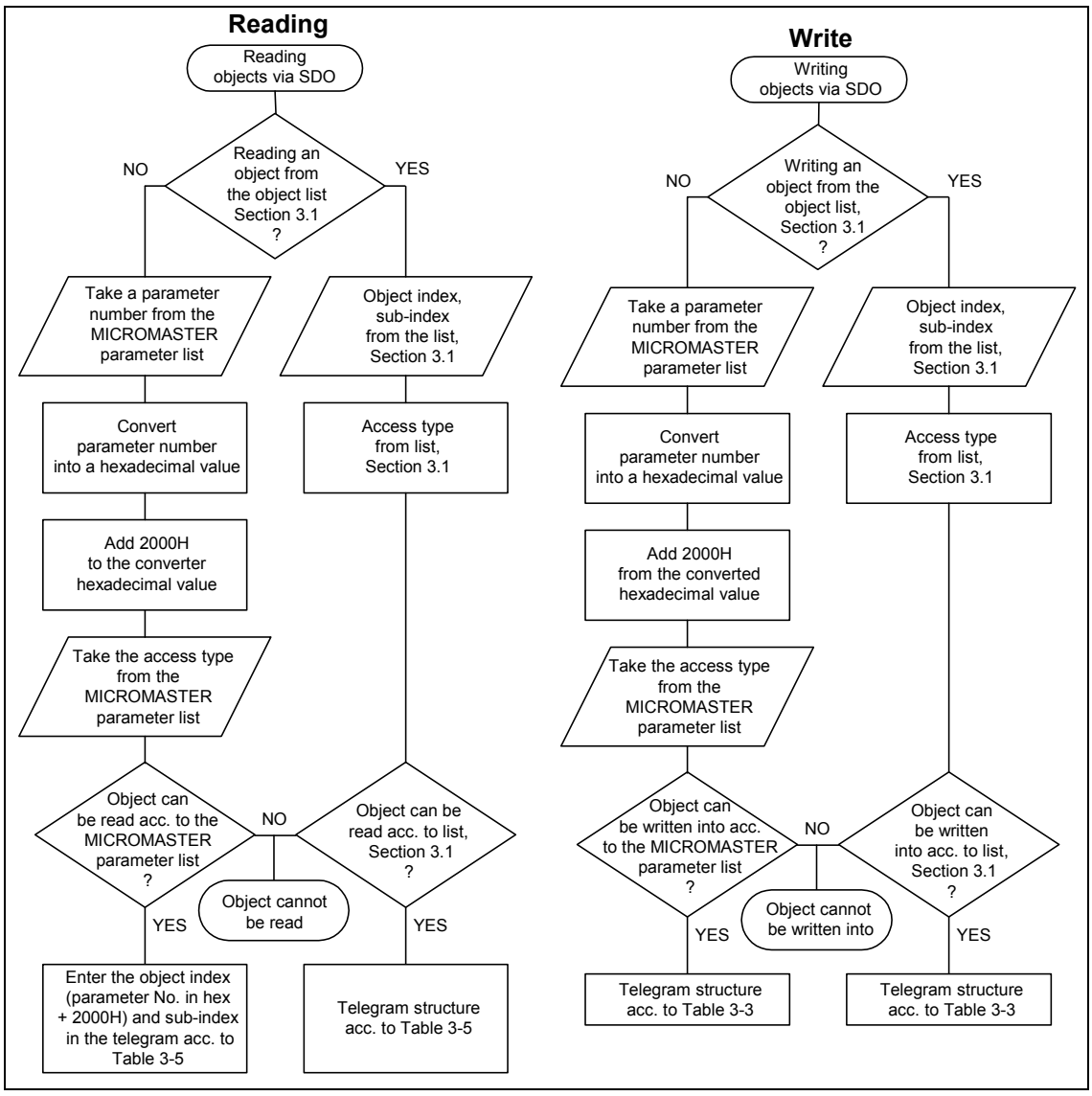


Fig. 3-10 Reading / writing PKW objects via SDO

Using two examples, the following two sections show the read and write access operations using SDO on two PKW objects.

Example 1: Reading parameter r0964.02

Parameter number (964) must be converted into a hexadecimal number in order to read the value of parameter r0964.02:

964dec = 3C4H.

After this, 2000H must be added:

03C4H + 2000H = 23C4H.

The numerical sequence after the point in the specification of the PKW object (in this case: 2 in r0964.02) corresponds to the subindex number of the CANopen object. This must be converted from a decimal number into a hexadecimal number:

02 dec =2H

Now, an SDO read task is defined for the object that was determined:

- Byte 0
  - Bit 5 - 7 (CCS) = 2H ->initiate upload request
  - Bit 0 - 4 = 0H ->always for initiate SDO-upload request
- Byte 1
  - Bit 0 - 7 = C4H ->index of the object, part 2
- Byte 2
  - Bit 0 - 7 = 23H ->index of the object, part 1
- Byte 3
  - Bit 0 - 7 = 02H ->subindex of the object
- Byte 4-7
  - Bit 0 - 7 = 00H ->always for initiate SDO-upload request

CANopen telegram of the master:

Table 3-7 SDO read request to parameter r0964.02 (object 0x23C4* sub2)

Byte 0:		**Bytes 1**	**Byte 2**	**Byte 3**	**Bytes 4,5**	**Bytes 6,7**
Bits 7-5 CCS = 2	Bit 4-0 = 0	C4H	23H	02H	reserved = 0	

CANopen MICROMASTER response telegram

Table 3-8 MICROMASTER response to the SDO read request

Byte 0:					**Bytes 1,2**	**Byte 3**	**Bytes 4,5**	**Bytes 6,7**
Bits 7-5 SCS = 2 (initiate upload response)	Bit 4 = 0	Bits 3,2 n= 2 (word) (number of bytes that do not contain any data)	e = 1 expedited transfer)	s = 1 (data size is indicated)	23C4H (index)	02H (subindex)	Word data (2 bytes data)	0H (two bytes, no data as specified in n)

Example 2: Writing into parameter P1203

In order to write the value 99 (decimal) into parameter P1203.00, the parameter number must first be converted into a hexadecimal number:

1203dec = 4B3H

Then, 2000H must be added:

04B3H + 2000H = 24B3H

The numerical sequence after the point in the specification of the PKW object (in this case: 0 in P1203.00) corresponds to the subindex number of the CANopen object. It must be converted from a decimal number into a hexadecimal number:

00 dec =0H

The data type of the parameter must now be taken from the MICROMASTER parameter list. This is for parameter P1203 unsigned 16 (U16).

The value to be written (in this case, 99 decimal) must be specified in the CANopen telegram as hexadecimal number:

99dec = 63H

Now, an SDO write request is defined for the object that was determined:

- Byte 0
  - Bit 5 - 7 (CCS) = 1H ->initiate download request
  - Bit 4 = 0H ->always for initiate download request
  - Bit 2 - 3 = 2H ->n=number of bytes that do not contain any data
  - Bit 1 = 1H ->e=1 expedited transfer
  - Bit 0 = 1H ->s=1 data size is indicated
- Byte 1
  - Bit 0 - 7 = B3H ->index of the object, part 2
- Byte 2
  - Bit 0 - 7 = 24H ->index of the object, part 1
- Byte 3
  - Bit 0 - 7 = 00H ->subindex of the object
- Byte 4-5
  - Bit 0 - 7 = 63H ->data that is to be written into
- Byte 6-7
  - Bit 0 - 7 = 00H ->data that is to be written into, in this case empty

CANopen telegram of the master

Table 3-9 SDO write request for parameter (object 0x24B3* sub0)

Byte 0:					**Bytes 1,2**	**Byte 3**	**Bytes 4,5**	**Bytes 6,7**
Bits 7-5 CCS = 1	Bit 4 = 0	Bits 3,2 n = 2 (Word)	e = 1	s = 1	24B3H	0H	63H	0H

CANopen MICROMASTER response telegram

Table 3-10 MICROMASTER response to the issued SDO write request

Byte 0:		**Bytes 1,2**	**Byte 3**	**Bytes 4,5**	**Bytes 6,7**
Bits 7-5 SCS = 3 (initiate download response)	Bit 4-0 = 0	24B3H (index)	0H (sub-index)	reserved = 0	

Additional comments regarding this Chapter:

- **A** request (SDO request) or **a** response (SDO Response) always refers to **one** parameter value.
- The slave only sends the response if data is available in the MICROMASTER 420/430/440. For specific objects, the appropriate BICO connections must first be generated. This takes approximately 50 ms under normal operating conditions.
  If the inverter cannot respond to the task that was issued, then it sends an abort telegram.
- The master may only issue a new SDO request if the response to the previous request was received.

3.2.5 PDO (Process Data Object) services

PDO services are used to transfer data which are critical from a time perspective. PDO connections follow the Producer/Consumer model. Whereby a normal PDO connection follows the Push model and an RTR connection, the Pull model. Several objects are mapped in a PDO. This mapping is an agreement between the sender and receiver as to which object is located at which position in the PDO. This means that the sender knows at which position in the PDO it should write data and the receiver knows to where it should transfer data which it received. A more detailed description of the PDO mapping is provided in Sections 2.2 and 5.4.

Properties and features of PDOs

- Generally unconfirmed data transfer (only confirmed for RTR queries)
- For fast data transfer of up to 8 bytes of net data without protocol overhead
- Data transfer types: Cyclic, synchronously, non-cyclically, synchronously, RTR synchronously, RTR non-synchronously, non-synchronously

PDO services

Write PDO service

The Write PDO service is unacknowledged. There is a PDO producer which sends its PDO to the PDO consumer. There can be 0 or more consumers in the network - but always only one Producer. This service is used for the normal process data transfer between MICROMASTER 420/430/440 and one of the other CANopen nodes (e.g. CANopen master). Regarding Receive PDOs, MICROMASTER is the consumer and regarding Transmit PDOs, is the Producer.

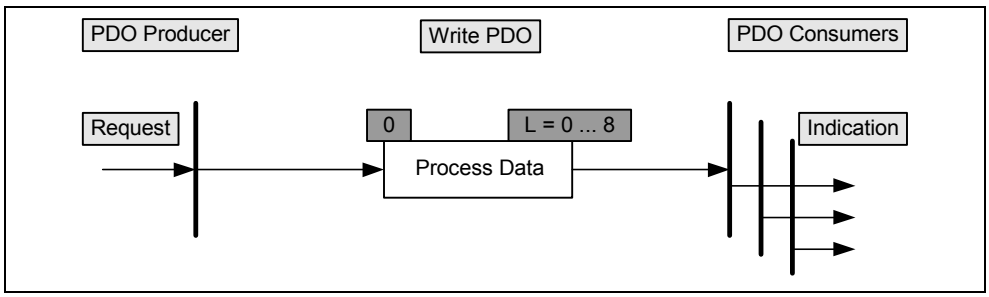


Fig. 3-11 Write PDO service

Read PDO service

The read PDO service is an acknowledged service. One or several PDO consumers send a Remote Transmission Request (RTR) message to the network. After it has received the RTR message, the PDO producer sends the requested PDO.

This service is used for RTR queries. Using this service, actual values can be interrogated independently of the selected cycle time.

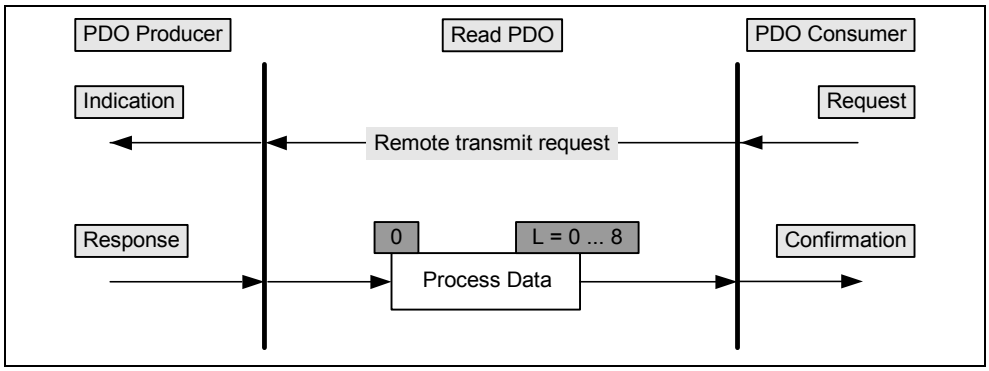


Fig. 3-12 Read PDO service

PDO identifier

It is recommended that an identifier is allocated to the PDOs in the CANopen profile for the first four transmit and receive PDOs. However, this can always be changed by writing into the appropriate communication objects. In Micromaster 440 the COBIDs are pre-set as follows:

Receive PDOs

1. 200H + Node ID
5. 300H + Node ID

   (corresponds to the pre-setting [default] for 2. after DS301, however, the second RPDO does not exist for the Micromaster, this is the reason that COBID was assigned here for the next existing PDO)
6. 400H + Node ID
   (corresponds to the pre-setting for 3. after DS301)

Transmit PDOs

1. 180H + Node ID
5. 280H + Node ID
   (corresponds to the pre-setting [default] for 2. after DS301. However, the second TPDO does not exist for the Micromaster; this is the reason that the COBID was assigned here for the next existing PDO)
6. 380H + Node ID
   (corresponds to the pre-setting [default] for 3. after DS301)

PDO data transfer types

CANopen makes a differentiation between various data transfer types (trigger events for the PDOs).
The various data transfer types for PDOs is shown in the following table:

Table 3-11 PDO data transfer types

Data transfer type	**PDO data transfer**				
	Cyclic	Non-cyclic	Synchronous	Non-synchronous	RTR
0		X	X		
1 - 240	X		X		
241 - 251	Reserved				
252			X		X
253				X	X
254				X	
255				X	

Significance of the various data transfer types:

Non-cyclic synchronous transfer ⇒ If a value has changed, it is sent after each SYNC telegram

Cyclic synchronous data transfer ⇒ A value is sent (n = 1 - 240) after each n^{th} SYNC telegram

RTR, synchronous data transfer ⇒ After an RTR query has been received, after the next received SYNC message, the value is sent.

RTR, non-synchronous data transfer ⇒ The value is immediately sent after an RTR query has been received.

Non-synchronous ⇒ Non-synchronous PDOs are always sent if an event has occurred (the value has changed). Data transfer types 254 and 255 differ as follows: For 254, the trigger has a manufacturer-specific origin. For 255, the trigger is received from an event defined in the profile. In order to receive the PDO in a specific cycle, in spite of the non-synchronous data transfer, even if the value did not change, in the communication objects event timers can be set which means that PDOs are sent after the selected time.

Synchronous PDOs are transferred using the standardized SYNC object. The SYNC Producer sends these periodically.

The Transmission Type (operating mode) specifies the data transfer type and the trigger initiation for a PDO. For synchronous Transmit PDOs, the Transmission Type also defines the data transfer rate in the form of a factor. For a factor of 0, the PDO is transferred after each SYNC object if an additional event has occurred (a value has changed). For a factor of 1, the PDO is transferred after every SYNC object. For a factor n (n = 1 - 240), the PDO is sent after every n^{th} SYNC signal. Non-synchronous PDOs are sent independently of the SYNC object. Data of a synchronous Receive PDO, that is received after a SYNC object, is processed by the application after the next received SYNC object - independently of the value of the factor for the Transmission Type. The data of an asynchronous PDO are always directly transferred to the application.

For receive PDOs, the MICROMASTER 420/430/440 only makes a differentiation between synchronous and non-synchronous PDOs. The factor is not saved in the MICROMASTER. This means that after power on, the MICROMASTER only displays in the communication objects as to whether it received the PDO synchronously or non-synchronously. The Power-On value of CANopen objects, that can define the data transfer type of the PDOs, can, under certain circumstances, deviate from the true values of the objects in the drive inverter. This is the reason that we urgently recommend that these objects are not saved using a "SAVE" command via object 0x1010. In fact, after every Power-On command they should be explicitly re-defined by CAN in order to avoid an incorrect function.

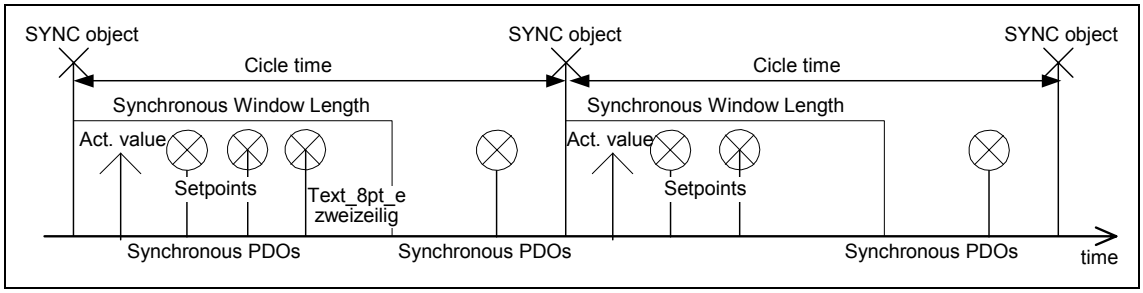


Fig. 3-13 Data transfer types

Fault: PDO data transfer type:

If these objects are at 255, and were saved with "Save", then they can no longer be set back to a synchronous data transfer type. Or, the CAN display remains at the value 255, even if the inverter is reset to 0 using the commissioning (start-up) tool and this is also displayed.

3.2.6 Synchronization service

A SYNC producer sends the synchronization object cyclically as broadcast telegram. The SYNC telegram defines the basic clock cycle of the network. The time between the SYNC telegrams is set using the object Communication Cycle Period (1006H). In order to obtain a precise (accurate) cycle between the SYNC signals, the SYNC telegram is sent with a high-priority identifier (generally 80H). This can be modified using the object 1005H. The SYNC transfer applies the Producer/Consumer Push model and is non-confirmed.

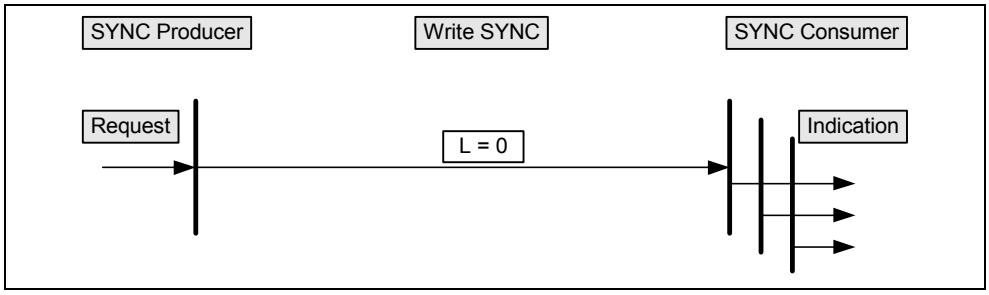


Fig. 3-14 Synchronization service

3.2.7 EMERGENCY object service

The EMERGENCY object is used to transfer an error message to the CANopen master, or also to another node which can process the error message.

The node fault which last occurred is indicated in the object 1003H Predefined Error Field. In addition, an 8 byte EMERGENCY telegram is sent non-synchronously.

The EMERGENCY telegram structure is shown in the following table:

Table 3-12 Structure of the EMERGENCY telegram

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
CANopen Error code	CANopen Error code	CANopen Error register	MICRO-MASTER Error number	MICRO-MASTER Error number	Reserved	Reserved	Reserved

Significance of the individual bytes

Byte 0/1 CANopen error code. This corresponds to an MICROMASTER fault. The interrelationship between the CANopen error number and the MICROMASTER error number is shown in Table 3-14.

Byte 2 Error register. In the error register it is indicated as to which error class the error belongs. This fault code is also used for display in object 1001H.

- Bit 0: General error
- Bit 1: Current error
- Bit 2: Voltage error
- Bit 3: Temperature error
- Bit 4: Communications error
- Bit 5: Device profile-specific
- Bit 6: Reserved (always 0)
- Bit 7: Manufacturer-specific

Byte 3/4 MICROMASTER fault number. Using this fault number it is possible to directly view the MICROMASTER documentation to identify the error involved and possibly initiate counter-measures.

Byte 5-7 Not used. These bytes are always zero.

NOTE

Warnings do not initiate EMERGENCY telegrams – they are indicated by the warning bit in the status word. The CANopen master can read-out the warning number from object 283EH.

Optimizing EMERGENCY telegrams

In order to better handle Emergency telegram that are sent non-synchronously when high bus load levels exist, there are various optimization methods.
Emergency messages can be suppressed by setting bit 31 in object 1014H (COB-ID EMCY). Then, faults that occur are only displayed in the status word and can be read-out of object 1003H.
Object 1014H also allows the COB-ID to be changed, which can then be used to modify the priority of the message. Using the object 1015H (Inhibit Time EMCY), it is possible to set a delay time for sending the EMCY message.

MICROMASTER 420/430/440 sends an Emergency message if an fault has occurred in either the option module or in the inverter.

Table 3-13 EMERGENCY messages that are not generated by the drive

Error Code (Bytes 1/ 2) (Hex)	Meaning	Additional code (Bytes 3/4)	Description
6300H	Dataset update failed	1	Failed to change to new drive dataset. Further information in diagnostic parameter P2054[0]. Note that this message does not have a corresponding reset-emergency message!!!
7600H	ResetCommunication or ResetNode failed	2	Tried to execute ResetNode, or ResetCommunication after data has been saved to EEPROM. Note that this message does not have a corresponding reset-emergency message!!!
7600H	ResetCommunication or ResetNode not possible	2	ResetNode, or resetCommunication failed. Further information in diagnostic parameter P2054[0]. Note that this message does not have a corresponding reset-emergency message!!!
8000H	CAN sync error	0	CAN sync object not received within communication cycle period 1006H x 1.5
8000H	Drive running	1	Tried to execute ResetNode, or ResetCommunication whilst drive running. Note that this message does not have a corresponding reset-emergency message!!!
8110H	CAN overrun	0	Data Overrun has occurred (unable to process data before next message arrived).
8130H	CAN lifeguarding failure	0	RTR frame was not received within the time: Node Guard time x Node Lifetime factor
8140H	Busoff cleared	0	Busoff condition has been cleared. Note that this message does not have a corresponding reset-emergency message!!!

The following table indicates all CANopen error codes and the associated MICROMASTER faults. A precise description is provided in the MICROMASTER documentation.

Table 3-14 Assignment of the CANopen error codes to MICROMASTER errors. Comment: Additional error numbers were introduced here that are not defined in CANopen in order to be able to more precisely specify the fault/error.

Error Code (Bytes 1/2) (Hex)	**Meaning**	**Fault number (Bytes 3/4) Dec/Hex**	**Sub Fault number (Bytes 5/6)**	**Description**
1000	Undecoded error	0000	0000	Read the error number from Byte 4
2120	Earth leakage	F0021/0015H	0000	Earth fault (output)
2300	Current on device output side	F0005/0005H	0000	Inverter I2T
2301	Current on device output side	F0011/000BH	0000	Motor Overload
2310	Current on Device output side	F0001/0001H	0000	Overcurrent
3120	Mains undervoltage	F0020/0014H	0000	Mains Phase Missing
3130	Phase failure	F0023/0017H	0000	Output phase disconnected
3210	DC link overvoltage	F0002/0002H	0000	Overvoltage (DC link)
3220	DC link under voltage	F0003/0003H	0000	Undervoltage (DC link)
4110	Excess ambient temp	F0004/0004H	0000	r0949/0 Inverter overtemp
4111	Excess ambient temp	F0004/0004H	0002/0002H	r0949/2 Ambient overtemp
4210	Excess temp device	F0004/0004H	0001/0001H	r0949/1 Rectifier overtemp
4211	Excess temp device	F0004/0004H	0003/0003H	r0949/3 Ebox overtemp
4212	Excess temp device	F0024/0018H	0000	Rectifier overtemp (input)
5220	Control/Computing circuit	F0060/003C H	0000	Asic Timeout
5400	Power section	F0022/0016H	0001/0001H	r0949/1 Powerstack fault IGBT short Chopper short Earth fault IO board not plugged in proper
5401	Power section	F0022/0016H	0012/000C H	r0949/12 UCE failure
5401	Power section	F0022/0016H	0013/000D H	r0949/13 UCE failure
5401	Power section	F0022/0016H	0014/000E H	r0949/14 UCE failure
5402	Power section	F0022/0016H	0021/0015H	r0949/21 I2C bus read error
5510	Data storage/RAM	F0101/0065H	0000	Stack overflow

Error Code (Bytes 1/2) (Hex)	Meaning	Fault number (Bytes 3/4) Dec/Hex	Sub Fault number (Bytes 5/6)	Description
5530	Data storage/EEPROM	F0051/0033H	0000	Parameter EEPROM fault
5531	Data storage/EEPROM	F0052/0034H	0000	Powerstack fault (I2C read / invalid data)
5532	Data storage/EEPROM	F0053/0035H	0000	IO EEPROM fault
7000	Additional modules	F0054/0036H	0000	Wrong IO board
7001	Additional modules	F0070/0046H	0000	Control board setpoint timeout
	Motor ID failure	F0041/0029H	0000	Motor ID failure See additional word for further information
7125			0000	Alarm value = 0: Load missing
7126			0001/0001H	Alarm value = 1: Current limit level reached during identification.
7127			0002/0002H	Alarm value = 2: Identified stator resistance less than 0.1 % or greater than 100 %.
7128			0003/0003H	Alarm value = 3: Identified rotor resistance less than 0.1 % or greater than 100 %.
7129			0004/0004H	Alarm value = 4: Identified stator reactance less than 50 % and greater than 500 %
7130			0005/0005H	Alarm value = 5: Identified main reactance less than 50 % and greater than 500 %
7131			0006/0006H	Alarm value = 6: Identified rotor time constant less than 10 ms or greater than 5 s
7132			0007/0007H	Alarm value = 7: Identified total leakage reactance less than 5 % and greater than 50 %
7133			0008/0008H	Alarm value = 8: Identified stator leakage reactance less than 25 % and greater than 250 %
7134			0009/0009H	Alarm value = 9: Identified rotor leakage inductance less than 25 % and greater than 250 %
7135			0020/0014H	Alarm value = 20: Identified IGBT on-voltage less than 0.5 or greater than 10 V
7136			0030/001EH	Alarm value = 30: Current controller at voltage limit
7137			0040/0028H	Alarm value = 40: Inconsistence of identified data set, at least one identification failed
7305	Sensor/incremental sensor 1 fault	F0090/005AH	0000	Encoder feedback loss
7330	Inverter temp signal lost	F0012/000CH	0000	Inverter temp signal lost
7331	Motor temp signal lost	F0015/000FH	0000	Motor temp signal lost

Error Code (Bytes 1/2) (Hex)	Meaning	Fault number (Bytes 3/4) Dec/Hex	Sub Fault number (Bytes 5/6)	Description
7332	FAN failure	F0030/0029H	0000	FAN failure
7510	Communication/serial interface No. 1	F0071/0047H	0000	USS/BOP link setpoint timeout
7520	Communication/serial interface No. 2	F0072/0048H	0000	USS/COMM link setpoint timeout
FF22	PID f/b above max value	F0222/00DEH	0000	PID f/b above max value
FF35	Restart after fault retries	F0035/0023H	0000	Restart after fault retries
FF40	Auto Cal failure	F0040/0028H	0000	Auto Cal failure
FF42	Speed optimization failure	F0042/002AH	0000	Speed optimisation failure
FF52	Belt failure detected	F0452/01C4H	0000	Belt failure detected
FF80	ADC lost input signal	F0080/0050H	0000	ADC lost input signal
FF85	External Fault	F0085/0055H	0000	External Fault

3.3 Controlling MICROMASTER drive converters via CANopen

The state machine for drives - as is supported by MICROMASTER - is shown in the diagram below. The state machine defines the various stati of the drives and the possible transitions between them. In order to speed-up the drive run-up, the drive does not require a second command to progress (advance) from Ready to power-up to operation enable. If the control has set the two relevant bits, that are required for these transitions, then the MICROMASTER only displays the powered-up (switched-on) state in the status word and then automatically changes into the Operation Enable state.

CANopen state machine

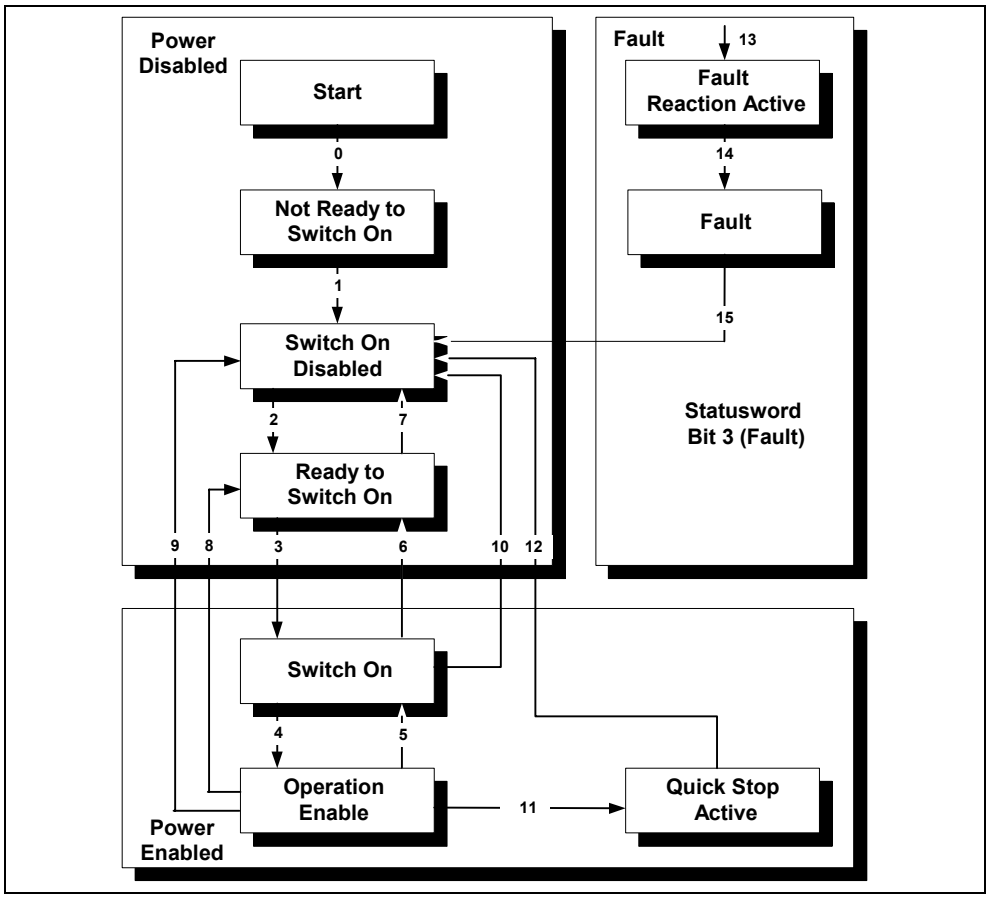


Fig. 3-15 State machine for drives

3.3.1 CANopen control word

The CANopen control word is used in order to execute the state transitions described in the previous Chapter.

Parameter P0700 (object:0x22BC*) must be set to 6 in order to transfer the CANopen control word to the appropriate internal parameters.

The CANopen control word is then written into the inverter at P2050.00 (object 0x2802*) and is then internally copied into parameter r2090. The CANopen control word is converted to the MICROMASTER control word in the option module. The assignment of the bits of the CANopen control word to the MICROMASTER control word are shown in the following table.

Table 3-15 Assignment of CANopen control word bits to MICROMASTER control word bits

Bit Pos.	**CANopen Control Word**		**Drive Control Word**	
00	Switch ON	0 = NO 1 = switch ON	ON/OFF1	0 = OFF1 1 = RUN
01	Disable Voltage (OFF2)	1 = NO 0 = disable voltage	Do OFF2	0 = YES 1 = NO
02	Quick Stop (OFF3)	1 = NO 0 = quick stop	Do OFF3	0 = YES 1 = NO
03	Enable Operation	0 = NO 1 = enable operation	Pulse Enable	0 = NO 1 = YES
04	Operation Mode Specific: RFG disable	0 = disable RFG 1 = enable RFG	RFG Enable	0 = NO 1 = YES
05	Operation Mode Specific RFG Stop	0 = stop RFG 1 = start RFG	RFG Start	0 = NO 1 = YES
06	Operation Mode Specific RFG Zero	0 = Zero RFG 1 = RFG not zeroed	Setpoint Enable	0 = NO 1 = YES
07	Reset Fault		Fault Acknowledge	0 = NO 1 = YES
08	Halt		Not used in CAN (forced to 0)	
09	Reserved (0)		No connection (forced to 0)	
10	Reserved (always 1)		Control from PLC (forced to 1)	0 = NO 1 = YES
11	Manufacturer Specific		Reverse	0 = NO 1 = YES
12	Manufacturer Specific		Not used	
13	Manufacturer Specific		Motor pot. UP	0 = NO 1 = YES
14	Manufacturer Specific		Motor pot. DOWN	0 = NO 1 = YES
15	Manufacturer Specific		CDS Bit 0 (Local / Remote) (needs manually setting of P0810.0 = 2090.15)	0 = NO 1 = YES

This means that it is not possible to stop the drive using the stop bit of the CANopen control word. Bits 4, 5 and 6 are specific to the operation mode and only exist in the velocity mode.

The bit combinations which result in the various state machine transitions are shown in the following table.

Table 3-16 Device control commands

Command/ Bit of the control word	Bit 7 Fault reset	Bit 3 Enable operation	Bit 2 Quickstop	Bit 1 Switch-on voltage (enable voltage)	Bit 0 Switch-on	Transitions
Shutdown	0	X	1	1	0	2,6,8
Switch on	0	X	1	1	1	3
Disable voltage	0	X	X	0	X	7,9,10,12
Quick stop	0	X	0	1	X	7,10,11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4
Fault reset	⇑	X	X	X	X	15

3.3.2 CANopen status word

The CANopen status word is used to define the drive status according to the state machine described above. Especially after a transition, by sending the CANopen control word from the master to the drive, it can be identified as to whether the change (transition) was correctly executed.

In order to transfer the CANopen status word to the appropriate internal parameters, parameter P0700 (object:0x22BC*) must be set to 6.

The MICROMASTER status word (r0052, corresponds to object 0x20034*) is sent to the CANopen option module via the BICO connection with P2051.00 (object 0x2803*). There, it is converted into the CANopen status word.

The assignment of the bits of the CANopen status word to the MICROMASTER status word are shown in the following table.

Table 3-17 Assignment of CANopen status word bits to MICROMASTER status word bits

Bit	CANopen Status Word		Bit	Drive Status Word	
00	Ready to Switch ON	0 = NO 1 = YES	00	Drive READY	0 = NO 1 = YES
01	Switched ON	0 = NO 1 = YES	01	Drive READY to RUN	0 = NO 1 = YES
02	Operation Enabled	0 = NO 1 = YES	02	Drive RUNNING	0 = NO 1 = YES
03	Fault	0 = NO 1 = YES	03	Drive FAULT ACTIVE	0 = NO 1 = YES
04	Voltage Disabled	0 = NO 1 = YES	04	OFF 2 active	0 = NO 1 = YES
05	Quick Stop: ❶	0 = YES 1 = NO	05	OFF3 Active: ❶	0 = NO 1 = YES
06	Switch on Disabled	0 = NO 1 = YES	06	ON INHIBIT active	0 = NO 1 = YES
07	Warning	0 = NO 1 = YES	07	Drive WARNING active	0 = NO 1 = YES
08	Manufacturer specific	0 = NO 1 = YES	10	Max Frequency Reached	0 = NO 1 = YES
09	Remote Control	0 = NO 1 = YES	09	PZD Control	0 = NO 1 = YES
10	Target Reached:	0 = NO 1 = YES	08	Deviation present: Deviation between Setpoint / Act. Value	 0 = YES 1 = NO
11	Internal limit active	0 = NO 1 = YES	11	Warning: Motor Current Limit	0 = NO 1 = YES
12	Operation Mode Specific ❷		12	Motor Holding Brake Active	0 = NO 1 = YES
13	Operation Mode Specific ❷		13	Motor Overload	0 = NO 1 = YES
14	Manufacturer specific		14	Motor runs Direction right	0 = NO 1 = YES
15	Manufacturer specific		15	Inverter Overload	0 = NO 1 = YES

❶ Note inversion between Drive status word and CANopen Status word.

❷ These bits are defined as “reserved” for velocity mode in the CANopen Spec so these definitions may not be possible

The status messages of the CANopen status word are listed in the following table.

Table 3-18 Status word bits in the various states

State	**Bit 6 Switch on disabled**	**Bit 5 Quick stop**	**Bit 4 Voltage disabled**	**Bit 3 Fault**	**Bit 2 Operation enabled**	**Bit1 Switched on**	**Bit 0 Ready to switch on**
Not ready to switch on	0	X	X	0	0	0	0
Switch on disabled	1	X	X	0	0	0	0
Ready to switch on	0	1	X	0	0	0	1
Switched on	0	1	X	0	0	1	1
Operation Enabled	0	1	X	0	1	1	1
Fault	0	X	X	1	0	0	0
Fault Reaction active	0	X	X	1	1	1	1
Quick stop active	0	0	X	0	1	1	1

3.3.3 MICROMASTER control and status word 2

In addition to the CANopen control and status word, for Micromaster, there are also the so-called control and status words 2. If so required, these can be used to enable additional Micromaster functions.

In order to be able to address control word 2 of the MICROMASTER via CANopen, parameter r2050 .3 (object 0x2802* subindex 4) must be connected via the appropriate BICO connection!

Status word 2 can also be connected as checkback signals to control word 2 via parameter P 2051.3 (object 0x2803* subindex 4).

This is possible using the BICO connection from r0053 to r2051.03.

Bit 12 of control word 2 (torque control) is additionally changed by the CANopen object 0x6060 "modes_of_operation", in order to change between closed-loop speed and torque control and/or between "velocity mode" and" profile torque mode" (refer to Section 3.1). Object 0x6060 only becomes available if this BICO connection has been established from bit 12 of control word 2. This can be realized either via CAN, BOP or the start-up tool.

Note that the free object 0x2802* subindex 4 in the drive corresponds to parameter r2050.3 and that parameter r2050.3 is internally connected to parameter r2091. The bits of r2091 can be connected appropriately via BICO to provide the functions defined below (see Table 3-19 "Manually setting").

The meaning of the bits in control word 2 and status word 2 are as follows:

Table 3-19 Significance of the bits in control word 2 and status word 2

Bit Pos.	Control Word 2		Manually setting	Status Word 2	
00	Fixed Frequency Bit 0	0 = NO 1 = YES	P1020.0 = 2091.0	DC Brake Active	0 = NO 1 = YES
01	Fixed Frequency Bit 1	0 = NO 1 = YES	P1021.0 = 2091.1	Act Freq > freq.off	0 = NO 1 = YES
02	Fixed Frequency Bit 2	0 = NO 1 = YES	P1022.0 = 2091.2	Act Freq > f_min	0 = NO 1 = YES
03	Fixed Frequency Bit 3 (not MICROMASTER 420)	0 = NO 1 = YES	P1023.0 = 2091.3	Act Curr. >= P2170	0 = NO 1 = YES
04	Drive data set (DDS) Bit 0	0 = NO 1 = YES	P0820.0 = 2091.4	Act Freq > P2155	0 = NO 1 = YES
05	Drive data set (DDS) Bit 1	0 = NO 1 = YES	P0821.0 = 2091.5	Act Freq <= P2155	0 = NO 1 = YES
06	Not used		Not Used	Act Freq >= setpoint	0 = NO 1 = YES
07	Not Used		Not Used	Act Vdc < P2172	0 = NO 1 = YES
08	Enable PID	0 = NO 1 = YES	P2200.0 = 2091.8	Act Vdc > P2172	0 = NO 1 = YES
09	Enable DC Brake	0 = NO 1 = YES	P1230.0 = 2091.9	Ramping finished (setpoint reached)	
10	Not Used		Not Used	PID output == PID min	0 = NO 1 = YES
11	Enable Droop	0 = NO 1 = YES	Not Used	PID output == PID max	0 = NO 1 = YES
12	Torque Control (only 440)	0 = Speed Control 1 = Torque Control	P1501.0 = 2091.12	Not Used	
13	External Fault	0 = NO 1 = External Fault	P2106.0 = 2091.13	Not Used	
14	Not Used		Not Used	Download Data set 0 from AOP	0 = NO 1 = YES
15	Command Data Set (CDS)	0 = NO 1 = YES	P0811.0 = 2091.15	Download Data set 1 from AOP	0 = NO 1 = YES

3.3.4 Modes of Operation and Modes of Operation display

The CANopen object 0x6060 "modes-of-operation" is used to instruct devices to change their modes.

Whether the mode of operation of the device was actually changed is indicated in the object 0x6061 "modes of operation display".

In order to be able to address the CANopen object 0x6060, beforehand, a BICO connection must be established between P1501 and r2091.12. In CANopen this means writing the value 282B00C into object 0x25DD*.

MICROMASTER 420/430 only supports the Velocity Mode; MICROMASTER 440 additionally supports the Profile Torque Mode. The definition and the interpretation of the option module for this object is as follows:

Table 3-20 Significance of the settings in the object Modes of Operation

Modes of operation	Meaning	MICROMASTER reaction
–1 to –128	Manufacturer specific	Ignored by MICROMASTER Return emergency Message
0	Reserved	Ignored by MICROMASTER Return emergency Message
1	Profile position mode	Ignored by MICROMASTER Return emergency Message
2	**Velocity Mode**	**Velocity Mode (MICROMASTER 420/430/440)**
3	Profile velocity mode	Ignored by MICROMASTER Return emergency Message
4	**Profile Torque Mode**	**Profile Torque Mode (MICROMASTER 440)**
5	Reserved	Ignored by MICROMASTER Return emergency Message
6	Homing mode	Ignored by MICROMASTER Return emergency Message
7	Interpolation position mode	Ignored by MICROMASTER Return emergency Message
8 – 127	Reserved	Ignored by MICROMASTER Return emergency Message

If "modes-of-operation" is set to the value 4, then, bit 12 (Torque Control) is automatically SET from control word 2 - and a transition made into the Torque Control Mode. If the value of "modes-of-operation" is set to 2, bit 12 (Torque Control) is RESET and a change/transition made to closed-loop speed control (Velocity mode). See Section 3.3.3 for the definition of control word 2. If the mode changeover is to be displayed in the CAN object 0x6061 "modes of operation display", then in addition, a BICO connection must be established from P2051.06 to r0055. Up until now, this connection can only be established using the BOP or the start-up (commissioning) tool.

CAUTION

Note that for mode selection to work control bit 12 of control word 2 must be BICO-linked to torque-control / speed-control selection bit via P1501 as follows:

P1501 = r2091.12

(i.e. 082B000C must be written into CAN object 0x25DD sub 0)

For the CAN object 0x6061 "modes of operation display", in addition, a BICO connection must be established from P2051.06 to r0055. Up until now, this connection can only be established using the BOP or the start-up (commissioning) tool.

4 Connecting to CANopen

4.1 Installing the CANopen module for Sizes A, B, C

WARNING

The drive converter/drive inverter must be powered-down (brought into a no-voltage condition) before installing or removing the CANopen communications module from a MICROMASTER 420/430/440.

Installing the module

Introduce the CANopen communications module at the lower end and engage the guide lugs in the unit and then move the top of the module towards the unit until it latches into place.

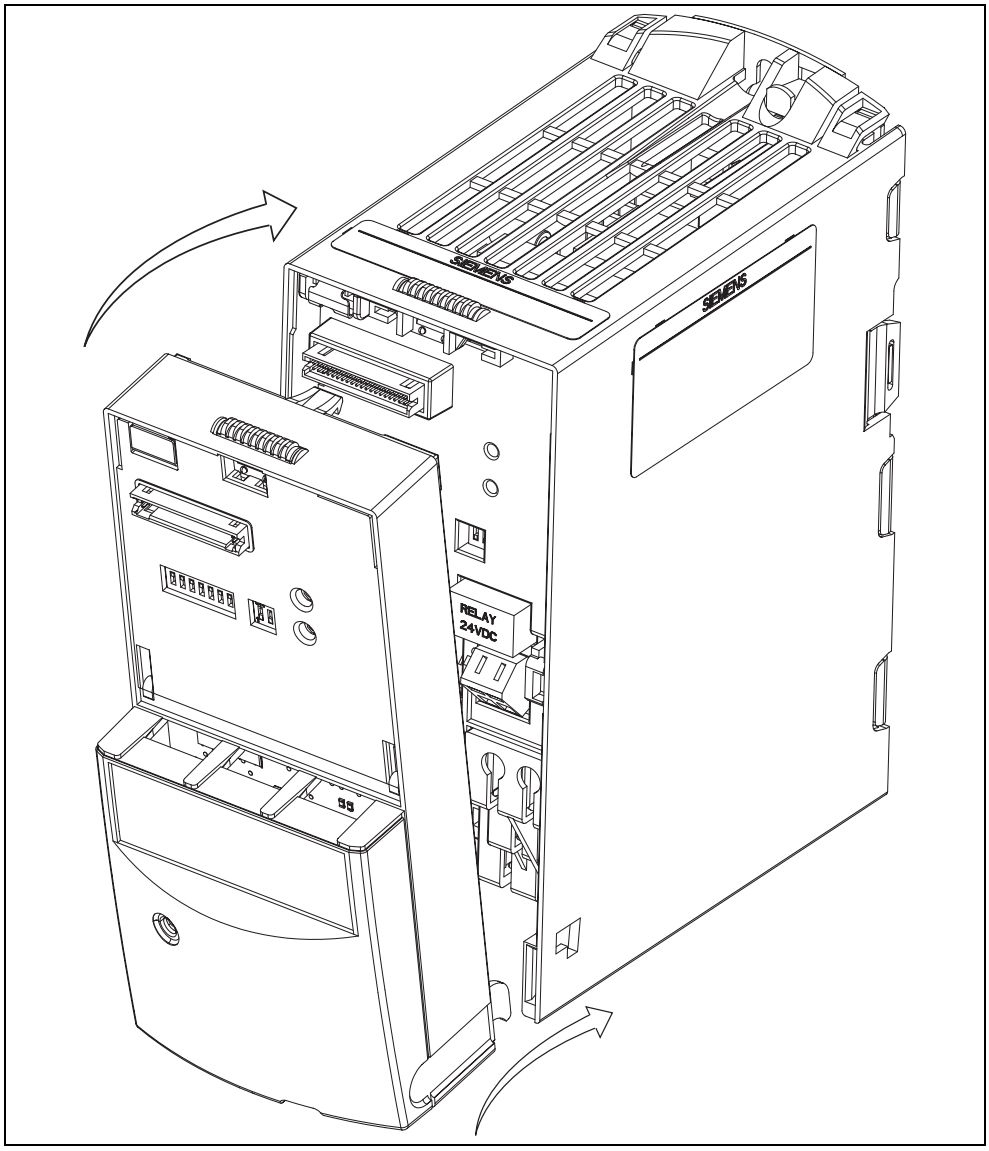


Fig. 4-1 Installing the communications module on MICROMASTER 4 Sizes A, B, C

4.2 Installing the CANopen module for Sizes D, E, F

WARNING

The drive converter/drive inverter must be powered-down (brought into a no-voltage condition) before the CANopen communications module is installed or withdrawn from a MICROMASTER 430/440.

Installing the communications module

For these Sizes, the CANopen communications module **is installed in the drive converter/drive inverter housing.**

To do this, the Standard Display Panel (SDP) and the front cover must be removed.

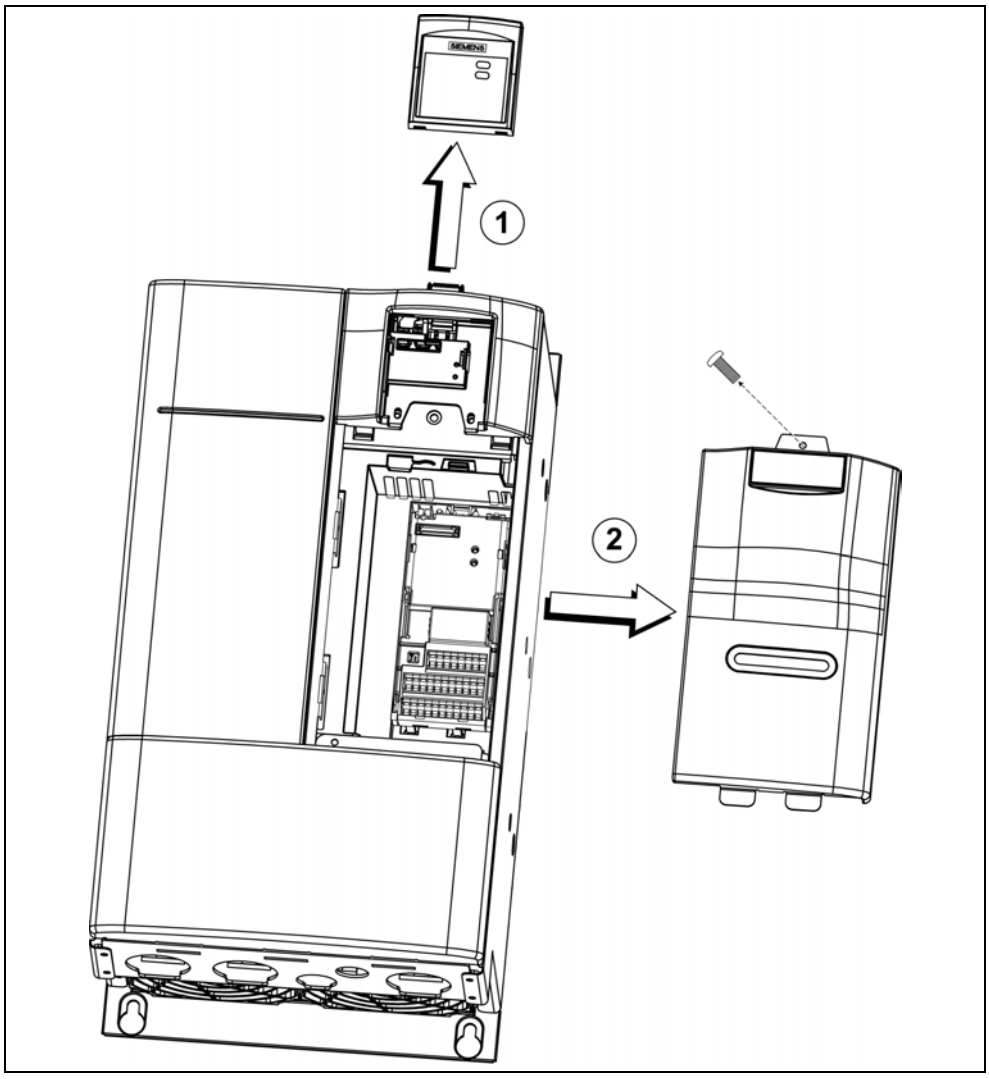


Fig. 4-2 Removing the covers for MICROMASTER 4, Sizes D, E, F

We recommend the following procedure when installing the communications module:

First insert the bus cable through the appropriate cable gland (the cable should not have a connector).

Then assemble the CANopen connector

Mount the connector on the communications module

Remove the Display Interface Module (DIM)

Snap the communications module onto the drive converter/drive inverter (into the appropriate recess)

Insert the Display Interface Module (DIM) into the installed communications module

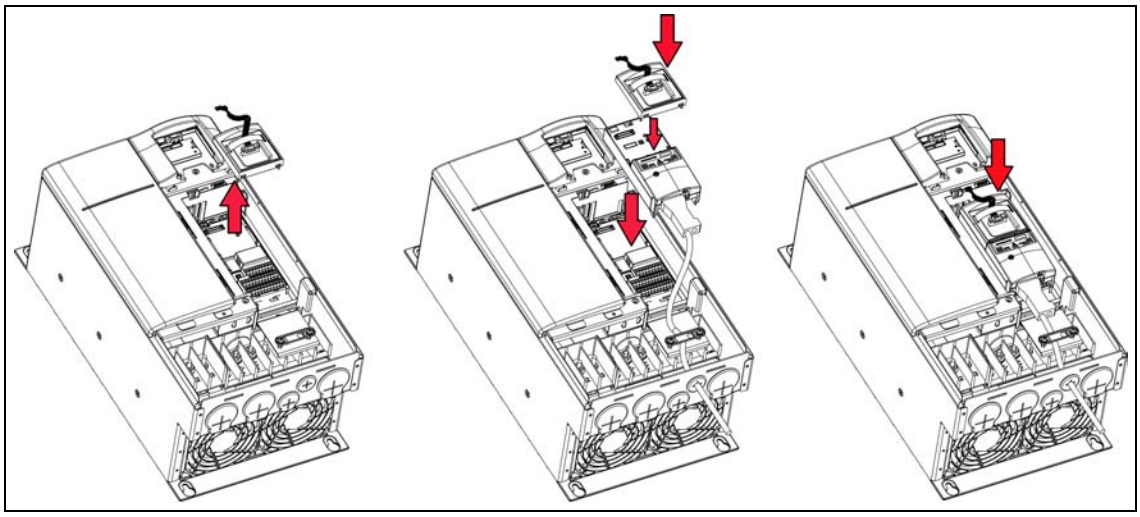


Fig. 4-3 Installing the communications module for MICROMASTER 4, Sizes D, E, F

Then re-locate the front cover and Standard Display Panel (SDP) on the drive.

4.3 Installing the CANopen module for Sizes FX, GX

WARNING

The drive converter/drive inverter must be powered-down (brought into a no-voltage condition) before the CANopen communications module is installed or withdrawn from a MICROMASTER 430/440.

Installing the communications module

For these Sizes, the CANopen communications module **is installed in the drive converter/drive inverter housing.**

To do this, the panel must be removed.

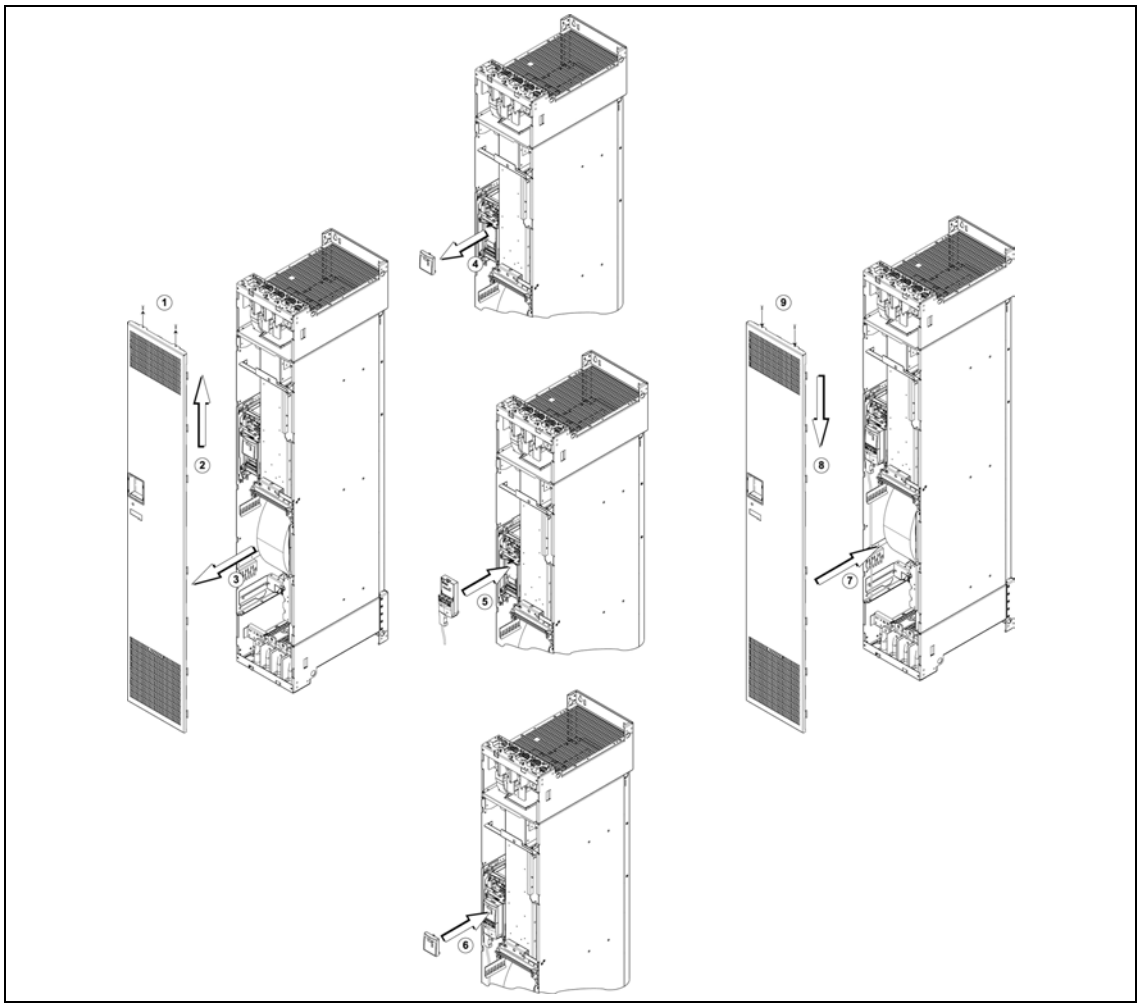


Fig. 4-4 Installing the communications module for MICROMASTER 4, Sizes FX, GX

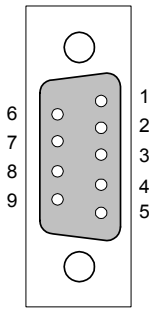
The communications module is installed essentially the same as for Size A (refer to Section 4.1). Then re-locate the covers on the drive converter/drive inverter.

4.4 Bus connection

Pin assignment of the bus connection

9-pin SUB-D plug connectors are used to connect the CANopen option module to the CAN bus.

Table 4-1 Assignment of CANopen sub-D9 connector



Pin	Function	Description
1	–	Reserved
2	CAN_L	CAN_L bus line (dominant low)
3	CAN_GND	CAN Ground
4	-	Reserved
5	CAN_SHLD	Optional CAN Shield
6	GND	Optional Ground
7	CAN_H	CAN_H bus line (dominant high)
8	-	Reserved
9	CAN_V+ (not MICROMASTER)	Optional CAN external positive supply

CAN-bus cables

The CANopen DRP 301-1 profile recommends cables with the following properties/features:

Table 4-2 Recommended bus cable according to CiA

Bus length [m]	Bus cable (1)		Termination resistance [Ω]	Baud rate [kbit/s]
	Length-related resistance [mΩ/m]	Cross-section [mm2]		
0 ... 40	70	0.25 ... 0.34	124	1000 at 40 m
40 ... 300	< 60	0.34 ... 0.6	150 ... 300	> 500 at 100 m
300 ... 600	< 40	0.5 ... 0.6	150 ... 300	> 100 at 500 m
600 ... 1000	< 26	0.75 ... 0.8	150 ... 300	> 50 at 1 km

(1) Recommended cable AC parameters: 120-Ω impedance and 5-ns/m specific line delay

Bus termination

In order to guarantee perfect operation of the CAN bus, bus terminating resistors must be provided at both ends of the bus cable (refer to Table 4-2).

Ground connection

Generally, the CAN ground can be connected. However, in completely electrically isolated CANopen networks the CAN connection does not have to be grounded.

Cable lengths

The lengths of the bus cables in a CANopen network depend on the baud rate and the delay times of the various nodes.

The maximum cable lengths for a CANopen network with a CANopen master and a MICROMASTER 420/430/440 are calculated in the following example. It is assumed that the CANopen MASTER has the same delay times as MICROMASTER 420/430/440.

When calculating the maximum cable lengths between the two nodes located the farthest away from each other, it is decisive that the nodes with the highest delay are also located at the start and at the end of the network. Another calculation must be made if a node with even higher delay times is to be located in a network.

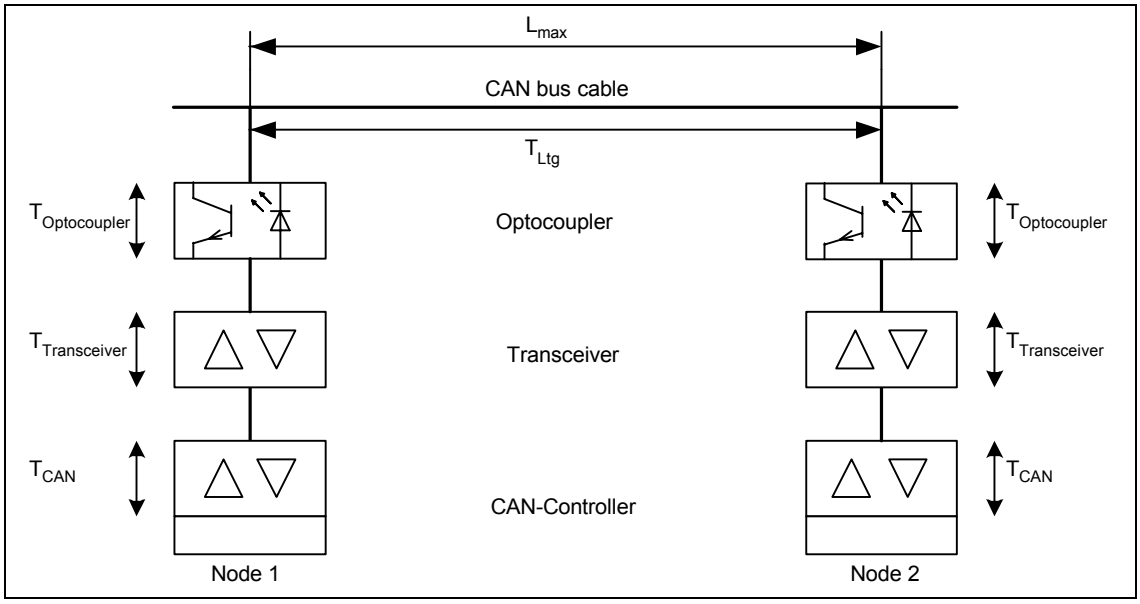


Fig. 4-5 CANopen network with delay times

T_{CAN}	Time lag of the CAN interface chip
$T_{Transceiver}$	Time lag of the transceiver element
$T_{Optocoupler}$	Time lag of the opto-coupler
T_{Ltg}	Time lag of the cable
L_{max}	Maximum length of cable

The maximum length of cable between two nodes is calculated using the formula below:

$$L_{max} = \frac{\frac{0,85}{2 \cdot Baudrate} - T_{el}}{T_{Ltg}} =$$

$$\frac{\frac{0,85}{2 \cdot Baudrate} - ([T_{CAN}\ In+out]+[T_{Transceiver}\ Tx+Rx]+[2 \cdot T_{Optocoupler}])}{T_{Ltg}}$$

The following values must be taken into account for the maximum cable length between a MICROMASTER and a CANopen-Master with the same values:

CAN interface chip = T_{CAN} = In+out 26 ns

Line driver = $T_{Transceiver}$ (Tx) = 80 ns

Line driver = $T_{Transceiver}$ (Rx) = 150 ns

Opto-coupler = $T_{Optocouler}$ = 40 ns

Cable time lag = T_{Ltg} = 5 ns/m

Signal propagation time = $0{,}85 \cdot T_{Bit} = \frac{0{,}85}{\text{Baudrate}}$

(The value 0.85 is permanently set in the MICROMASTER)

This produces the following equation:

$$\frac{\frac{0{,}85}{2 \cdot \text{Baudrate}} - ([T_{CAN}\ \text{In+out}] + [T_{Transceiver}\ \text{Tx+Rx}] + [2 \cdot T_{Optocoupler}])}{T_{Ltg}} =$$

$$\frac{\frac{0{,}85}{2 \cdot \text{Baudrate}} - ([26\,\text{ns}] + [80\,\text{ns} + 150\,\text{ns}] + [2 \cdot 40\,\text{ns}])}{5\,\frac{\text{ns}}{\text{m}}}$$

The following approximate cable lengths are obtained from this formula:

Table 4-3 Maximum cable lengths between two bus nodes with the delay times of MICROMASTER 420/430/440

Baud rate (kbit/s)	**Maximum network range (m)**
1000	18
800	39
500	103
250	273
125	613
50	1633
20	4183
10	8433

5 Commissioning with CANopen

When commissioning a system for the first time via CANopen, it is important for the master that it knows the default values of the Node ID (in this case: 3) and the baud rate (in this case: 10kbit/s) of a new node. This then allows the master to configure it corresponding to the specified requirements. Further, there must be an EDS file, to provide information about its default, highest and lowest values as well as its data type via existing CANopen objects. Practical examples for commissioning a system for the first time are provided in the Chapters: "Commissioning in the velocity mode" and "Commissioning in the profile torque mode".

5.1 Electronic Data Sheet (EDS)

The information required for the existing object of a node are saved in the EDS file. This EDS file is provided on the CD that is included in the scope of supply of the CANopen module.

5.2 Important parameters when commissioning the system for the first time

NOTE

The CANopen option module can only be connected if the inverter was previously disconnected/isolated from the power supply.

When commissioning the CANopen option module for the first time, the settings of the following parameters must be checked:

PKW object	CANopen object	Description
P0918	0x2396*	CANopen node-ID (network node address of the inverter) Default value: 3
P2010	0x27DA*	CANopen baud rate (baud rate in kbit/s) Default value: 10
P2040	0x27F8*	Process data telegram failure time, refer to Section 5.3 Default value > 0, should be set to 0 when commissioning for the first time
P2041	0x27F9*	Communication-module functions (to set other CANopen features), refer to Section 5.6
P0700	0x22BC*	Selects the command source (for complex applications) Must be set to 6 for CANopen
P1000	0x23E8*	Selects the frequency setpoint (for complex applications) Must be set to 6 for CANopen
P0719	0x22CF*	Must be set to 0 in order that P700 and P1000 can be used
r2050	0x2802*	Selects the process data setpoint (BICO) (for complex applications)
P2051	0x2803*	Selects the process data actual values (BICO). Corresponding to the CAN parameter, is "read-only". The BICO connection can only be established using the BOP or commissioning tool
P0927	0x239F*	Modification source for parameters
r2053	0x2805*	Software release and other software information of the CANopen option module. (refer to Section 7.4)
r2054	0x2806*	Communication module diagnostics (refer to Section 7.3)

NOTE

If the CAN parameters - baud rate and NodeID - were changed, then it is either necessary to run-up again or send the "reset node" command of the CAN master in order to make the changes in the drive effective. The run-up must be initiated by powering-OFF/ON the inverter (powering-down and powering-up again).

5.3 Parameter "P2040" (object 0x27F8*), telegram failure time

Parameter "P2040" (object 0x27F8*) is used to activate the monitoring of the telegram failure time. This parameter is used to set the time for cyclic monitoring of the master (life guarding and SYNC). Fault F0070 is activated if the CANopen option module does not receive a new telegram within the telegram failure time (this depends on the setting of P2041.04, (or object 0x27F9* sub 4) and then specifies the response to this fault).
If one of two telegrams fails, the particular alarm (refer to Section 7.2) and a fault telegram are activated.

The monitoring function is activated as standard (default setting). If the master does not immediately start with node guarding, then this results in a fault state of the drive..

"P2040" = 0 no monitoring

"P2040" > 0 telegram failure time in milliseconds (as standard > 0 !)

Caution!

When commissioning the system for the first time, we recommend that parameter "P2040" (object 0x27F8*) is set to zero in order to prevent the drive going into a fault state.

5.4 Setting-up the PDOs

The PDOs can either be configured using a CANopen master or using a commissioning (start-up) tool. There are several restrictions when it comes to using and configuring the PDOs due to the fact that the configuration parameters of the CAN module are not available..

Section 5.6 discusses how parameters 2041.02 and P2041.03 are used to define the mapping of PDOs 5 and 6 using a commissioning (start-up) tool.

It should be ensured, that at power-up of the PDOs, the default COB_Ids - as they are described in the CANopen specification DS301, are set. It is not possible to modify the 4 most significant bits of these COB_IDs. However, the drive can change the 7 least significant bits of the COB_IDs as these comprise the CANopen node ID.

The power-on values of the CANopen objects, that define the data transfer type of the PDOs, can, under certain circumstances, deviate from the true values of the objects in the inverter. This is the reason that we urgently recommend that these objects are not saved using a "SAVE" command via object 0x1010. In fact, after every power-on, they should be explicitly re-defined using CAN in order to avoid an incorrect function.

The supported PDOs with their default values are as follows:

PDO number	**COB_ID**	**Default mapping**	**Comment**
R_PDO_1	200H + node ID	Control_word Fixed mapping	Cannot be mapped
T_PDO_1	180H + node ID	Status_word Fixed mapping	Cannot be mapped
R_PDO_5	300H + node ID	**MICROMASTER 420/430:** Default: No mapping	Mapping via CANopen master or with the low byte from parameter P2041.02 Refer to Section 5.6
T_PDO_5	280H + node ID	**MICROMASTER 420/430:** Default: No mapping	Mapping via CANopen master or with the low byte from parameter P2041.02 Refer to Section 5.6
R_PDO_6	400H + node ID	**MICROMASTER 420/430/440:** Default: No mapping	Mapping via CANopen master or with the low byte from parameter P2041.02 Refer to Section 5.6
T_PDO_6	380H + node ID	**MICROMASTER 420/430/440:** Default: No mapping	Mapping via CANopen master or with the low byte from parameter P2041.02 Refer to Section 5.6

In order to set the mapping for PDO 5 or PDO 6 via the CANopen master, the usual sequence must be maintained (also refer to DS 301):

1/ Set the number of mapping entries in the subindex 0 to 0

2/ Write new mapping entries - one after the other - into sub-indices 1 to 4

3/ Set the number of mapping entries in subindex 0 to the number of the now newly mapped objects (2 for 2 mapping entries, 4 for 4 mapping entries...)

NOTE

The internal MICROMASTER parameters only use a single bit in order to specify a mapping parameter. This is the reason that in CANopen, a specific sequence must be maintained when writing the mapping values. The connection between drive parameters and mapping entry is specified in the following table:

Direction	Mapping	Saved mapping value	Mapping sequence	Bit number in P2041.02 or P2041.03
R_PDO	Control word	60400010	1 (lowest mapping index)	Bit 0
R_PDO	Setpoint (reference) speed	60420010H or 60520010H	2	Bit 1
R_PDO	Free object 2802 subindex 3	28020310H	3	Bit 2
R_PDO	Free object 2802 subindex 3	28020410H	4	Bit 3
R_PDO	Setpoint (reference) torque ❶	60710010H	5	Bit 4
R_PDO	Free object 2802 subindex 3 ❶	28020610H	6	Bit 5
R_PDO	Modes of operation ❶	60600008H	7 (highest mapping index)	Bit 6
R_PDO	Select RPM or per unit	-	-	Bit 7
T_PDO	Status word	60410010H	1 (lowest mapping index)	Bit 8
T_PDO	Speed control quantity	60430010H or 60530010H	2	Bit 9
T_PDO	Free object 2803 index 3	28030310H	3	Bit 10
T_PDO	Free object 2803 index 4	28030410H	4	Bit 11
T_PDO	Actual torque ❶	60770010H	5	Bit 12
T_PDO	Actual speed ❶	60440010H or 60540010H	6	Bit 13
T_PDO	Displays the operating mode ❶	60610008H	7 (highest mapping index)	Bit 14
T_PDO	Select RPM or per unit	-	-	Bit 15

❶ Objects can only be mapped in MICROMASTER 440

Initially, values that correspond to the lowest bit numbers must be mapped.
As example, here, the following mapping should be set for RPDO_5:

Control word, free object 2802.03, setpoint (reference) torque and modes of operation.

The following entries are written into the objects of RPDO5:

Set the number of mapping entries to 0 (write 0 to 1604H subindex zero).

1604H index 01 = control word = 60400010H Mapping sequence from list = 1

1604H index 02 = free object 2802.03 = 28020310H Mapping sequence from list = 3

1604H index 03 = setpoint (reference) torque = 60710010H Mapping sequence from list = 5

1604H index 04 = modes of operation = 60600008H Mapping sequence from list = 7

Set the number of mapping entries to 4 (write 4 to 1604H subindex 0) in order to enable the PDO

This would be correct and the mapping entry would be used in RPDO5.

However, if the entries are written as follows:

Set the number of mapping entries to 0 (write 0 to 1604H subindex zero).

1604H index 01 = control word = 60400010H Mapping sequence from list = 1

1604H index 02 = setpoint (reference) torque = 60710010H Mapping sequence from list = 5

1604H index 03 = free object 2802.03 = 28020310H Mapping sequence from list = 3

1604H index 04 = modes of operation = 60600008H Mapping sequence from list = 7

Set the number of mapping entries to 4 (write 4 to 1601H subindex 0), in order to enable PDO.

This mapping sequence would result in an error!

The free object 2802.03 must be mapped in a lower object directory subindex than the setpoint (reference) torque. This error would then have been sent as EMCY message to the CANopen master - the mapping would not have been used.

5.5 Timing of the PDOs

The process data of the CANopen option module are updated approx. every 8 ms. This means, on one hand, that the CANopen master should only send a new setpoint (reference) value telegram every 8 ms - but, on the other hand, for a non-synchronous PDO transfer type, actual values are only updated and sent every 8 ms. Setpoints, that are sent in a shorter cycle than the 8ms, can be overwritten. Only the last setpoint telegram before the data transfer between the CANopen option module and MICROMASTER 420/430/440 is valid.

5.6 PDO properties in parameter "P2041"

Instead of using the usual CANopen mechanisms, the properties of the PDOs can be set and checked using the indexed parameter P2041(object 27F9*).This includes the PDO mapping and the PDO data transfer type. Further, in this parameter, the behavior of the bus node for a Live Guarding Event, the baud rate, the response of the CAN state machine to loss of communications (using object 1029H) and the response of the drive status machine to communications loss (using object 6007H) are defined.

The possible settings are listed in the following tables.

Table 5-1 Parameterizing the communications module using parameter (object 27F9*)

Para-meter	Content	Comment	MICRO-MASTER	
			420/430	440
2041.00	Low Byte = Transmission Type for T_PDO_1 High Byte = Transmission Type for T_PDO_5	Transmission Type for T_PDO_1 and T_PDO_5 See Table 5-2 for definition of Transmission Type. **Example**. T_PDO_1 must be transmitted asynchronously. T_PDO_5 must be transmitted cyclically and synchronously every sync pulse. High byte = 1H, Low byte = FFH Convert 1FFH to decimal = 511 Enter 511 into P2041.00 Example setup : 2041.00 = 65535 T_PDO_1 = 255 T_PDO_5 = 255	X	X
2041.01	Low Byte = Transmission Type for T_PDO_6 High Byte = Transmission Type for R_PDO_1, R_PDO_5 and R_PDO_6	Transmission Type for T_PDO_6 and R_PDO_1, R_PDO_5 and R_PDO_6 See Table 5-2 for definition of Transmission Type. Transmission Type for T_PDO_6 has the same definition as the Transmission Types described in 2041.00. Transmission Type for R_PDO_1, R_PDO_5 and R_PDO_6: Note that for a node receiving a PDO, it is only necessary to know whether the received PDO data is transferred immediately to the Drive (asynchronous transfer), or whether the PDO is transferred to the Drive on the next sync pulse (synchronous transfer). For this reason for most efficient parameter storage the Transmission Types for R_PDO_1, R_PDO_5 and R_PDO_6 are stored as single bits: Bit 8 = 0 means R_PDO_1 is asynchronous and 255 is stored as the Transmission Type for R_PDO_1 Bit 8 = 1 means R_PDO_1 is synchronous and 0 is stored as the Transmission Type for R_PDO_1 Bit 9 = 0 means R_PDO_5 is asynchronous and 255 is stored as the Transmission Type for R_PDO_5 Bit 9 = 1 means R_PDO_5 is synchronous and 0 is stored as the Transmission Type for R_PDO_5 Bit 10 = 0 means R_PDO_6 is asynchronous and 255 is Stored as the Transmission Type for R_PDO_6 Bit 10 = 1 means R_PDO_6 is asynchronous and 255 is Stored as the Transmission Type for R_PDO_6 Note that a CANopen configurator can change the values for the R_PDO_1, R_PDO_5 and R_PDO_6 Transmission Types to anything within the allowable range, but only one bit will be used within the Drive to store this information for each R_PDO. After a power cycle, because only one bit is used for each R_PDO Transmission Type, the R_PDO Transmission Types will be set to either 255 (asynchronous) or 0 (synchronous) and the configurator can write in the exact values if required. However, note that as far as a CANopen slave is concerned, it is only necessary to know whether the R_PDO is synchronous or asynchronous and overwriting by the CANopen configurator with exact values of Transmission Type is not really necessary. Example setup: 2041.01 = 2047 T_PDO_6 = 255 R_PDO_1 = asynchronous = 255 R_PDO_5 = asynchronous = 255 R_PDO_6 = asynchronous = 255	X	X

Para-meter	Content	Comment	MICRO-MASTER	
			420/430	440
2041.02	Least Significant Byte = Mapping for R_PDO_5 Most Significant Byte = Mapping for T_PDO_5	Individual bits in these bytes are used select objects to be mapped. The mapping is outlined below in the mapping Table 5-3 (MICROMASTER 420/430) and Table 5-4 (MICROMASTER 440).	X	X
		As an example, it is necessary to map PDO 5 so that it : • receives the control word and target torque • transmits the actual torque and the modes of operation display. From the Table 5-4 we see that R_PDO_5 = 11H T_PDO_5 = 50H P2041.02 = 5011H = 20497 decimal **Example setup for MICROMASTER 440:** P2041.02 = 1111H = 4369: R_PDO_5 = control_word (6040H), vl_target_torque (6071H) T_PDO_5 = Status_word (6041H), Actual_torque (6077H)		X
		Example setup for MICROMASTER 420/430: P2041.02 = 0303H = 771: R_PDO_5 = control_word (6040H), vl_target_velocity (6042H) T_PDO_5 = Status_word (6041H), Vl_velocity_demand (6043H)	X	
2041.03	Least Significant Byte = Mapping for R_PDO_6 Most Significant Byte = Mapping for T_PDO_6	Individual bits in these bytes are used select objects to be mapped. The mapping is outlined below in the mapping Table 5-3 and Table 5-4 (MICROMASTER 440).	X	X
		As an example, it is necessary to map PDO 6 so that it : • receives the control word, speed reference (in per-unit), target torque, and modes of operation • transmits the status word, actual speed (in RPM), actual torque, and modes of operation display From the Table 5-4 we see that R_PDO_6 = 11010011 = D3H T_PDO_6 = 01110001 = 71H P2041.02 = 71D3H = 29139 decimal		X
		Example setup MICROMASTER 420/430/440 P2041.02 = 0303H = 771: R_PDO_6 = control_word (6040H), vl_target_velocity (6042H) T_PDO_6 = Status_word (6041H), Vl_velocity_demand (6043H)	X	X

Para-meter	Content	Comment	MICRO-MASTER	
			420/430	440
2041.04	Least Significant Digit (ones) = CAN Statemachine Reaction to Comm Error Middle Digit: (tens) = Drive Statemachine to Comm Error	**Reaction to Comm Error**: If CAN Overflow, bus-off, node guarding error or Sync loss go to **the following state:** 0 = Pre-operational 1 = No state change 2 = Pre-operational with an OFF2 3 to 9 = reserved / not permissible Also the following action will be taken: – Warning will be generated in Drive. – Emcy message will be generated. – Error LED will be updated **Response to Lifeguarding Event** If CAN Overflow, bus-off, nodeguarding error or Sync loss go to **the following state:** 0 = remains in current state – Warning will be generated in Drive. – No Emcy message will be generated unless 1029H = 0 or 2. – Error LED will be updated 1 = malfunction – **TRIP the Drive**. This is achieved by forcing a comms timeout which will generate a trip F0070. If P2040 = 0, then time out is not possible and an OFF2 is generated instead. – Emcy message will be sent. – Error LED will be updated – Warning will be generated in Drive. 2 = disable voltage to switch on disabled – OFF 2 – Emcy message will be sent. – Error LED will be updated – Warning will be generated in Drive. 3 = quick stop to switch on disabled – OFF 3 – Emcy message will be sent. – Error LED will be updated – Warning will be generated in Drive. 4 = quick stop to switch on disabled – OFF 1 – Emcy message will be sent. – Error LED will be updated – Warning will be generated in Drive. 5 - 9 = reserved / not permissible	X	X

Para-meter	Content	Comment	MICRO-MASTER	
			420/430	440
2041.04	Most Significant Digit (hundreds): Baudrate	**Baud Rate** : 0 = 10 kbit/s 1 = 20 kbit/s 2 = 50 kbit/s 3 = 125 kbit/s 4 = 250 kbit/s 5 = 500 kbit/s 6 = 800 kbit/s 7 = 1 Mbit/s 8 – 9 = reserved, not permissable **Example**: required baudrate = 250 kbit/s CAN state machine reaction to comm error = No state change Drive state machine reaction to comm error = disable pulses (OFF2) Least significant digit = 2 Middle digit = 3 Most Significant Digit = 4 P2041.04 = 432 decimal	X	X

The subsequent table shows all of the possible data transfer types for Transmit PDOs. For Receive PDOs it is only possible to set as to whether the PDO can receive in synchronism or out of synchronism with the SYNC telegram. This can either be set using the appropriate CANopen objects or using bits 8, 9 and 10 of parameter 2041.01 (object 27F9*sub 1).
The power-on values of the CANopen objects, that define the data transfer type of the PDOs, can, under certain circumstances, deviate from the true values of the objects in the inverter. This is the reason that we urgently recommend that these objects are not saved using a "SAVE" command via object 0x1010. In fact, after every power-on, they should be explicitly re-defined using CAN in order to avoid an incorrect function.

Table 5-2 Table showing the possible data transfer types

Transmission Type	PDO Transmission				
	Cyclic	Acyclic	Synchronized	Asynchronous	RTR only
0		X	X		
1 - 240	X		X		
241 - 251	reserved				
252			X		X
253				X	X
254				X	
255				X	

The following tables indicate the significance of the various bits in parameters P2041.02 (object 27F9*sub 2) and P2041.03 (object 27F9*sub 3).

Table 5-3 Significance of the bits for MICROMASTER 420/430 in parameters P2041.02 and P2041.03

MICROMASTER 420/430				
SET Bit Number	**Direction**	**Object Dictionary Index**	**Drive Table**	**Description**
Data into Drive : Transferred from CANopen bus to CANopen objects to Drive parameters 2050.x				
Bit 0	R_PDO	6040H	2050.0	Map to CANopen control word (6040H). Note that this object is mapped internally in the Drive to Parameter P2090 which is bit-addressable.
Bit 1	R_PDO	6042H or 6052H	2050.1	Map to CANopen vl_target_velocity (RPM, 6042H) or vl_nominal_percentage (per-unit, 6052H) Depending on Bit 7.
Bit 2	R_PDO	2802.3H	2050.2	Map to Free Object (2802H subindex 3).
Bit 3	R_PDO	2802.4H	2050.3	Map to Free Object (2802H subindex 4). Note that this object is mapped internally in the Drive to Parameter P2091 which is bit-addressable. This therefore allows CANopen to change control word 2 or other control words (through appropriate BICO connections) if required.
Bit 4	Not used			
Bit 5	Not used			
Bit 6	Not used			
Bit 7	R_PDO			Choose mapping from from RPM or per-unit values. CLEAR: If Bit 7 clear, map to vl_target_velocity (RPM, 6042H) SET: If Bit 7 set, map to vl_nominal_percentage (Per-unit, 6052H)
Data out of Drive : Transferred from Drive parameters 2051.x to CANopen objects to CANopen bus				
Bit 8	T_PDO	6041H	2051.0	Map to CANopen status word (6041H)
Bit 9	T_PDO	6043H or 6053H	2051.1	Map to CANopen …. vl_velocity_demand (RPM, 6043H) or vl_percentage_demand (per-unit, 6053H) depending on bit 15
Bit 10	T_PDO	2803.3H	2051.2	Map to Free Object (2803H subindex 3)
Bit 11	T_PDO	2803.4H	2051.3	Map to Free Object (2803H subindex 4)
Bit 12	Not used			
Bit 13	Not used			
Bit 14	Not used			
Bit 15	T_PDO			Choose mapping from RPM or per-unit values. CLEAR: if Bit 15 clear, map to vl_velocity_demand (RPM, 6043H) SET: if bit 15 set, map to vl_nominal_percentage (Per-unit, 6053H)

Table 5-4 Significance of the bits for MICROMASTER 440 in parameters P2041.02 and P2041.03

MICROMASTER 440				
SET Bit Number	**Direction**	**Object Dictionary Index**	**Drive Table**	**Description**
Data into Drive : Transferred from CANopen bus to CANopen objects to Drive parameters 2050.x				
Bit 0	R_PDO	6040H	2050.0	Map to CANopen control word (6040H). Note that this object is mapped internally in the Drive to Parameter P2090 which is bit-addressable.
Bit 1	R_PDO	6042H or 6052H	2050.1	Map to CANopen…. vl_velocity_reference (RPM, 6042H) or vl_nominal_percentage (per-unit, 6052H) depending on Bit 7.
Bit 2	R_PDO	2802.3H	2050.2	Map to Free Object (2802H subindex 3)
Bit 3	R_PDO	2802.4H	2050.3	Map to Free Object (2802H subindex 4). Note that this object is mapped internally in the Drive to Parameter P2091 which is bit-addressable. This therefore allows CANopen to change control word 2 or other control words if required. Note that if modes of operation is mapped, then bit 12 of this object will have no effect because bit 12 of 2050.3 will be set or cleared depending on the value of modes of operation.
Bit 4	R_PDO	6071H	2050.4	Map to CANopen Target_Torque (6071H)
Bit 5	R_PDO	2802.6H	2050.5	Map to Free Object (2802H subindex 6)
Bit 6	R_PDO	6060H	2050.3 bit 12	Map to CANopen Modes_of_operation (6060H)
Bit 7	R_PDO			Choose mapping from RPM or per-unit values CLEAR: if Bit 7 clear, map to vl_velocity_reference (RPM, 6042H) SET: if bit 7 set, map to vl_nominal_percentage (per-unit, 6052H)
Data out of Drive : Transferred from Drive parameters 2051.x to CANopen objects to CANopen bus				
Bit 8	T_PDO	6041H	2051.0	Map to CANopen status word (6041H)
Bit 9	T_PDO	6043H or 6053H	2051.1	Map to CANopen vl_velocity_demand (RPM, 6043H) or vl_percentage_demand (per-unit, 6053H) depending on Bit 15
Bit 10	T_PDO	2803.3H	2051.2	Map to Free Object (2803H subindex 3)
Bit 11	T_PDO	2803.4H	2051.3	Map to Free Object (2803H subindex 4). Note that if modes of operation display is mapped, then bit 12 of this object will reflect the value of P2051.06 bit 12.
Bit 12	T_PDO	6077H	2051.4	Map to CANopen actual torque (6077H)
Bit 13	T_PDO	6044H or 6054H	2051.5	Map to CANopen vl_control_effort (RPM, 6044H) or vl_actual_percentage (per-unit, 6054H) depending on Bit 15
Bit 14	T_PDO	6061H	2051.6	Map to CANopen modes of operation display (6061H)

MICROMASTER 440				
SET Bit Number	**Direction**	**Object Dictionary Index**	**Drive Table**	**Description**
Bit 15	T_PDO			Choose mapping to RPM or per-unit values CLEAR: if bit 9 set, map to vl_velocity_demand (RPM, 6043H) if bit 14 set, map to vl_control_effort (RPM, 6044H) SET: if bit 9 set, map to vl_percentage_demand (per-unit, 6053H) if bit 14 set, map to vl_actual_percentage (per-unit, 6054H)

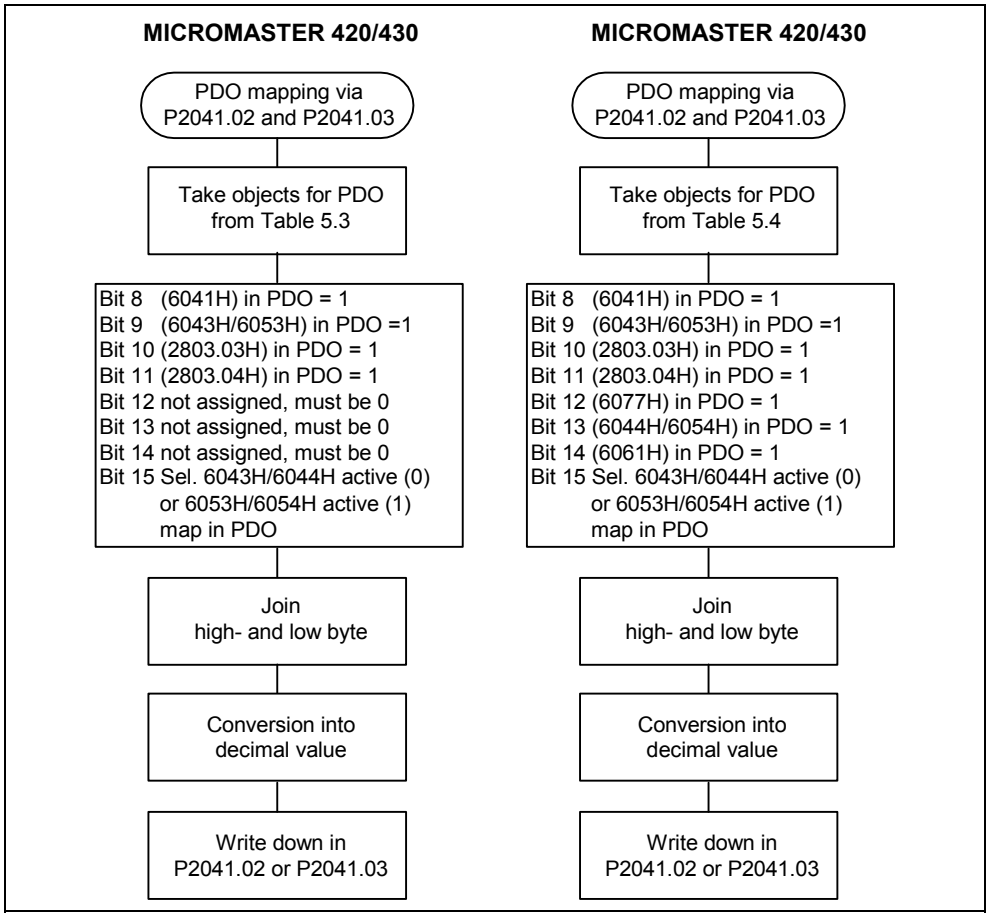


Fig. 5-1 Transmitting a PDO mapping via parameter P2041.02 (object 27F9*sub 2) and P2041.03 (object 27F9*sub 3)

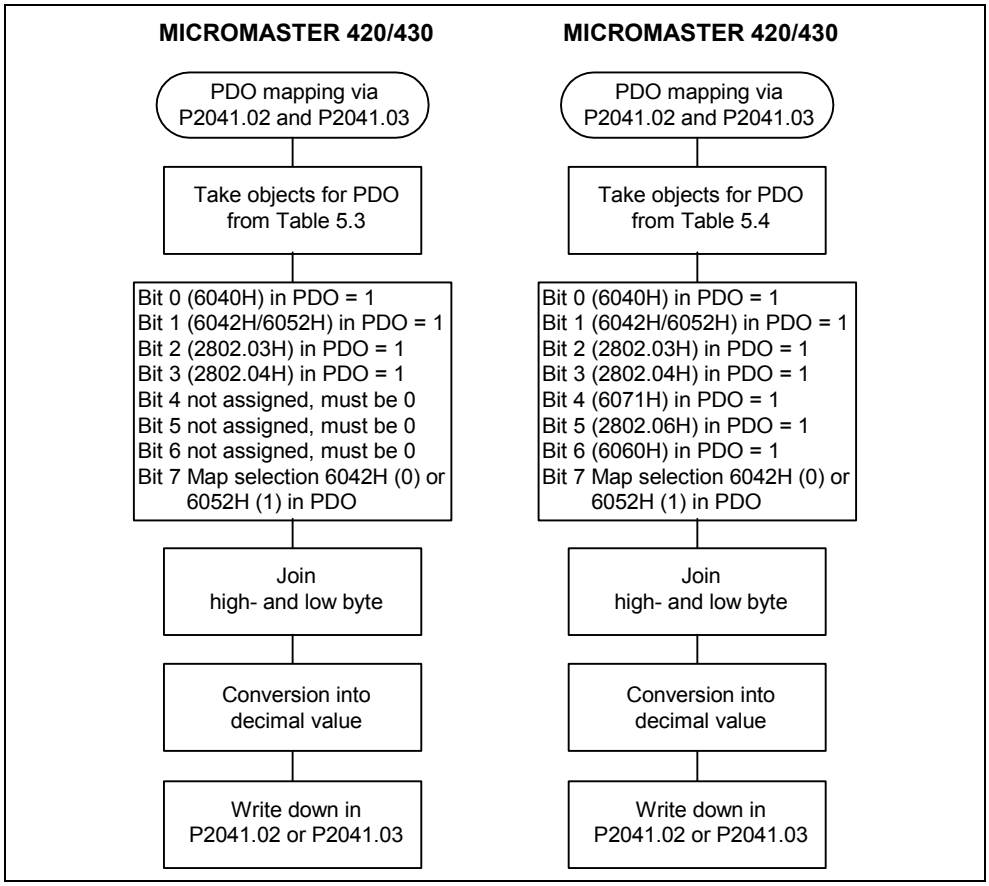


Fig. 5-2 Reading the set PDO mapping via parameters P2041.02 and P2041.03

Example of a PDO configuration via parameter P2041

The TPDO 5 is to be configured for the MICROMASTER 440. To do this, the PDO mapping must be set, the data transfer type selected and the COB-ID set with which the PDO is to be transferred.

The default value of the TPDO 5 looks like this:

COB-ID: 280H + node ID

PDO mapping: 0

Data transfer type: 0, i.e. non-cyclic synchronous - this means that if data has changed, it is sent after the next SYNC telegram.

These settings should now be changed to the following settings:

COB-ID should be 570H + node ID

PDO mapping: free_object 2803.03, torque_actual_value (6077H), vl_actual_percentage (6054H), modes_of_operation_display (6061H)

DData transfer type: Cyclic synchronous, i.e. data is, independently of a change, sent after each tenth SYNC telegram. This is realized as follows:

PDO mapping:

The appropriate parameters are taken from Table 5-1. In this case: Parameter P2041.02.

The significance of the individual bits that must be set in this parameter for the PDO mapping are shown in Table 5-4: Bit 10 for object 2803.03, bit 12 for object 6077H, bit 13 for object 6054H and bit 14 for object 6061H.

Bit 15 must be additionally set so that object 6054H is mapped and not object 6044H.

Data transfer type:

From Table 5-1 it can be seen that this is specified in the high byte of parameter P2041.00. Here, the value 10 must be entered for synchronous data transfer after each tenth value. The low byte of parameter P2041.00 specifies the data transfer type of TPDO 1. This means that the value for parameter P2041.00 is obtained as follows:

Data transfer type TPDO 1 in hex + data transfer type TPDO 5 in hex * 100H = parameter value in hex.

Then, the value must be converted into a decimal number and entered into parameter P2041.00.

COB-ID:

COB-ID can only be changed via the CAN bus (SDO access to object: 0x1404 sub 1). This value can be saved in a non-volatile fashion in EEPROM using CANopen object 0x1010.

5.7 First commissioning in the velocity mode

The first commissioning in the velocity mode is described in this Chapter. All of the necessary settings and CAN messages are documented here in order that the drive can be moved:

Firstly, the following BICO connections must be set using the BOP link or commissioning (start-up) tool:

Parameter P2051.6 = 55 - connection to display the operating modes

CAN object modes of operation display 0x6061 is active

Parameter P2051.5 = 63 - the actual speed is repeatedly read

CAN object 0x6044 is active

Caution! It is not possible to set this connection via CAN, as object 0x2803* (P2051) is a read-only object.

5.7.1 Additional commissioning via CANopen

Set the process data failure time to zero:

Write 0 into object 0x27F8 sub 0

```
T0 00:01:43,9766: 0x603  8-Data: 2b f8 27 00 00 00 00 00
R0 00:01:44,0105: 0x583  8-Data: 60 f8 27 00 00 00 00 00
```

Select the command source, P700=6

Write 6 into object 0x22BC* sub 0

```
T0 00:01:44,0716: 0x603  8-Data: 2b bc 22 00 06 00 00 00
R0 00:01:44,1494: 0x583  8-Data: 60 bc 22 00 00 00 00 00
```

Select the frequency setpoint, P1000=6

Write 6 into object 0x23E8* sub 0

```
T0 00:01:44,2375: 0x603  8-Data: 2b e8 23 00 06 00 00 00
R0 00:01:44,2642: 0x583  8-Data: 60 e8 23 00 00 00 00 00
```

Parameter P719 =0 so that P700 and P1000 can be used

Write 0 into object 0x22CF* sub 0

```
T0 00:01:44,3815: 0x603  8-Data: 2b cf 22 00 00 00 00 00
R0 00:01:44,4114: 0x583  8-Data: 60 cf 22 00 00 00 00 00
```

Parameter P1501.0 = P2091.12 (12th bit from parameter 2091)

BICO connection allows the operating modes to be changed-over (selected)

CANopen object 0x6060 is activated

Write 082B000C into object 0x25DD* sub 0

```
T0 00:01:44,4977: 0x603  8-Data: 22 dd 25 00 0c 00 2b 08
R0 00:01:44,5263: 0x583  8-Data: 60 dd 25 00 00 00 00 00
```

Set parameter P1300 (control type) to 0 for V/f operation (closed-loop speed control)

Write 0 into object 0x2514* sub 0

```
T0 00:01:44,6277: 0x603  8-Data: 2b 14 25 00 00 00 00 00
R0 00:01:44,6573: 0x583  8-Data: 60 14 25 00 00 00 00 00
```

Set the data transfer type of all PDOs to non-synchronous data transfer:

Write FF into object 0x1400 sub 2

```
T0 00:01:44,7392: 0x603  8-Data: 2f 00 14 02 ff 00 00 00
R0 00:01:46,4598: 0x583  8-Data: 60 00 14 02 00 00 00 00
```

Write FF into object 0x1404 sub 2

```
T0 00:01:46,5702: 0x603  8-Data: 2f 04 14 02 ff 00 00 00
R0 00:01:46,5988: 0x583  8-Data: 60 04 14 02 00 00 00 00
```

Write FF into object 0x1405 sub 2

```
T0 00:01:46,6895: 0x603  8-Data: 2f 05 14 02 ff 00 00 00
R0 00:01:46,7217: 0x583  8-Data: 60 05 14 02 00 00 00 00
```

Write FF into object 0x1800 sub 2

```
T0 00:01:46,7989: 0x603  8-Data: 2f 00 18 02 ff 00 00 00
R0 00:01:48,4259: 0x583  8-Data: 60 00 18 02 00 00 00 00
```

Write FF into object 0x1804 sub 2

```
T0 00:01:48,5405: 0x603  8-Data: 2f 04 18 02 ff 00 00 00
R0 00:01:50,2609: 0x583  8-Data: 60 04 18 02 00 00 00 00
```

Write FF into object 0x1805 sub 2

```
T0 00:01:50,3706: 0x603  8-Data: 2f 05 18 02 ff 00 00 00
R0 00:01:50,4000: 0x583  8-Data: 60 05 18 02 00 00 00 00
```

5.7.1.1 Execute the mapping for RPDO 4

Set the number of mapped objects to 0

Write 0 into object 0x1604 sub 0

```
T0 00:01:50,4900: 0x603  8-Data: 2f 04 16 00 00 00 00 00
R0 00:01:50,5034: 0x583  8-Data: 60 04 16 00 00 00 00 00
```

The control word is the first mapped object:

Write 0x60400010 into object 0x1604 sub 1

```
T0 00:01:50,6068: 0x603  8-Data: 23 04 16 01 10 00 40 60
R0 00:01:50,6205: 0x583  8-Data: 60 04 16 01 00 00 00 00
```

Target velocity is the second mapped object:

Write 0x60420010 into object 0x1604 sub 2

```
T0 00:01:50,7267: 0x603  8-Data: 23 04 16 02 10 00 42 60
R0 00:01:50,7394: 0x583  8-Data: 60 04 16 02 00 00 00 00
```

Set the number of mapped objects to 2

Write 2 into object 0x1604 sub 0

```
T0 00:01:50,8290: 0x603  8-Data: 2f 04 16 00 02 00 00 00
R0 00:01:52,4893: 0x583  8-Data: 60 04 16 00 00 00 00 00
```

5.7.1.2 Execute the mapping for TPDO 4

Set the number of mapped objects to 0

Write 0 into object 0x1A04 sub 0

```
T0 00:01:52,5891: 0x603  8-Data: 2f 04 1a 00 00 00 00 00
R0 00:01:52,6020: 0x583  8-Data: 60 04 1a 00 00 00 00 00
```

The status word is the first mapped object:

Write 0x60410010 into object 0x1A04 sub 1

```
T0 00:01:52,6976: 0x603  8-Data: 23 04 1a 01 10 00 41 60
R0 00:01:52,7110: 0x583  8-Data: 60 04 1a 01 00 00 00 00
```

The velocity-control effort is the second mapped object:

Write 0x60440010 into object 0x1A04 sub 2

```
T0 00:01:52,7920: 0x603  8-Data: 23 04 1a 02 10 00 44 60
R0 00:01:52,8058: 0x583  8-Data: 60 04 1a 02 00 00 00 00
```

Set the number of mapped objects to 2

Write 2 into object 0x1A04 sub 0

```
T0 00:01:52,9377: 0x603  8-Data: 2f 04 1a 00 02 00 00 00
R0 00:01:54,5864: 0x583  8-Data: 60 04 1a 00 00 00 00 00
```

5.7.1.3 Execute the mapping for RPDO 5

Set the number of mapped objects to 0

Write 0 into object 0x1605 sub 0

```
T0 00:01:54,6741: 0x603  8-Data: 2f 05 16 00 00 00 00 00
R0 00:01:54,6864: 0x583  8-Data: 60 05 16 00 00 00 00 00
```

The control word is the first mapped object:

Write 0x60400010 into object 0x1605 sub 1

```
T0 00:01:54,7770: 0x603  8-Data: 23 05 16 01 10 00 40 60
R0 00:01:54,7894: 0x583  8-Data: 60 05 16 01 00 00 00 00
```

Modes of operation is the second mapped object:

Write 0x60600008 into object 0x1605 sub 2

```
T0 00:01:54,8523: 0x603  8-Data: 23 05 16 02 08 00 60 60
R0 00:01:54,8659: 0x583  8-Data: 60 05 16 02 00 00 00 00
```

Set the number of mapped objects to 2

Write 2 into object 0x1605 sub 0

```
T0 00:01:54,9624: 0x603  8-Data: 2f 05 16 00 02 00 00 00
R0 00:01:56,6833: 0x583  8-Data: 60 05 16 00 00 00 00 00
```

5.7.1.4 Execute the mapping for TPDO 5

Set the number of mapped objects to 0

Write 0 into object 0x1A05 sub 0

```
T0 00:01:56,7626: 0x603  8-Data: 2f 05 1a 00 00 00 00 00
R0 00:01:56,7748: 0x583  8-Data: 60 05 1a 00 00 00 00 00
```

The status word is the first mapped object:

Write 0x60410010 into object 0x1A05 sub 1

```
T0 00:01:56,8695: 0x603  8-Data: 23 05 1a 01 10 00 41 60
R0 00:01:56,8819: 0x583  8-Data: 60 05 1a 01 00 00 00 00
```

Modes of operation display is the second mapped object:

Write 0x60610008 into object 0x1605 sub 2

```
T0 00:01:56,9977: 0x603  8-Data: 23 05 1a 02 08 00 61 60
R0 00:01:57,0108: 0x583  8-Data: 60 05 1a 02 00 00 00 00
```

Set the number of mapped objects to 2

Write 2 into object 0x1A05 sub 0

```
T0 00:01:57,1004: 0x603  8-Data: 2f 05 1a 00 02 00 00 00
R0 00:01:58,7806: 0x583  8-Data: 60 05 1a 00 00 00 00 00
```

Set the nodes into the operational mode:

```
T0 00:01:58,8546: 0x000  2-Data: 01 03
```

Receive the three PDOs:

```
R0 00:01:58,8655: 0x183  2-Data: 31 fe
R0 00:01:58,8741: 0x283  4-Data: 31 fe 00 00
R0 00:01:58,8816: 0x383  3-Data: 31 fe 02
```

Run-through the state machine of the drive:

Send 6 using PDO 1

```
T0 00:01:58,9832: 0x203  2-Data: 06 00
```

Send 7 using PDO 1

```
T0 00:01:59,0533: 0x203  2-Data: 07 00
```

Receive the three PDOs:

```
R0 00:01:59,0953: 0x183  2-Data: 33 fe
R0 00:01:59,1038: 0x283  4-Data: 33 fe 00 00
R0 00:01:59,1113: 0x383  3-Data: 33 fe 02
```

Send F using PDO 1

```
T0 00:01:59,1569: 0x203  2-Data: 0f 00
```

Receive the three PDOs:

```
R0 00:01:59,1942: 0x183  2-Data: 37 fe
R0 00:01:59,2027: 0x283  4-Data: 37 fe 00 00
R0 00:01:59,2103: 0x383  3-Data: 37 fe 02
```

The drive is now in the "operational“ state

Enter velocity 1000 and control word 7F using PDO4

-> the drive rotates

```
T0 00:01:59,3084: 0x303  4-Data: 7f 00 e8 03
```

Receive the three PDOs:

```
R0 00:01:59,3574: 0x183  2-Data: 37 fa
R0 00:01:59,3658: 0x283  4-Data: 37 fa 00 00
R0 00:01:59,3732: 0x383  3-Data: 37 fa 02
```

Receive the following three PDOs if the target velocity is reached and the "Target reached bit" is set by the drive:

```
R0 00:02:05,4193: 0x183  2-Data: 37 fe
R0 00:02:05,4278: 0x283  4-Data: 37 fe 00 00
R0 00:02:05,4354: 0x383  3-Data: 37 fe 02
```

5.7.2 Additional helpful commands for the velocity mode

Setting the baud rate to 500 kbit/s:

Write 500 into object 0x27DA sub 0:

```
T0 00:00:36,4500: 0x603  8-Data: 2b da 27 00 f4 01 00 00
R0 00:00:38,2083: 0x583  8-Data: 60 da 27 00 00 00 00 00
```

The new baud rate becomes active after the drive is re-started

Changing the NodeID (CANopen node name):

Write 4 for NodeID=4 into object 0x2396 sub 0

```
T0 00:00:19,0341: 0x603  8-Data: 2f 96 23 00 04 00 00 00
R0 00:00:19,0540: 0x583  8-Data: 60 96 23 00 00 00 00 00
```

After the drive re-starts, the new NodeID becomes active

Save the CANopen parameter with "SAVE", object 0x1010:

Write 0x65766173 into object 0x1010 sub 1

```
T0 00:00:19,1443: 0x603  8-Data: 23 10 10 01 73 61 76 65
R0 00:00:38,1658: 0x583  8-Data: 60 10 10 01 00 00 00 00
```

Check the PDO mapping using MICROMASTER parameter P2041 (0x27F9*):

The data apply to the mapping set above:

Read FF FF from object 0x27F9 sub 0:

```
T0 00:00:54,8671: 0x603  8-Data: 40 f9 27 00 00 00 00 00
R0 00:00:54,8856: 0x583  8-Data: 4b f9 27 00 ff ff 00 00
```

Read 00 FF from object 0x27F9 sub 1:

```
T0 00:00:54,9398: 0x603  8-Data: 40 f9 27 01 00 00 00 00
R0 00:00:54,9593: 0x583  8-Data: 4b f9 27 01 ff 00 00 00
```

Read 2103 from object 0x27F9 sub 2:

```
T0 00:00:55,0574: 0x603  8-Data: 40 f9 27 02 00 00 00 00
R0 00:00:55,0740: 0x583  8-Data: 4b f9 27 02 03 21 00 00
```

Read 41 41 from object 0x27F9 sub 3:

```
T0 00:00:55,1647: 0x603  8-Data: 40 f9 27 03 00 00 00 00
R0 00:00:55,1805: 0x583  8-Data: 4b f9 27 03 41 41 00 00
```

5.8 First commissioning in the Profile Torque Mode

A first commissioning in the velocity mode is described in this Chapter. All of the necessary settings and CAN messages are documented here in order to move the drive:

When commissioning in the Profile Torque Mode, the baud rate for CANbus communications must be set to 500kbit/s:

Write 500 into object 0x27DA sub 0

```
T0 00:00:36,4500: 0x603  8-Data: 2b da 27 00 f4 01 00 00
R0 00:00:38,2083: 0x583  8-Data: 60 da 27 00 00 00 00 00
```

The new baud rate becomes active after the drive is re-started

Initially, the following BICO connections must be set using the BOP or commissioning (start-up) tool:

Parameter P2051.6 = 55 - connecting to display the operating modes

CAN object modes of operation display 0x6061 is active

Parameter P2051.5 = 63 - the actual speed is repeatedly read

CAN object 0x6044 is active

Parameter P2051.4 = 80 - torque

CAN object 0x6077 is active

Caution! It is not possible to set these connections via CAN as object 0x2803* (P2051) is a read-only object.

5.8.1 Additional commissioning via CANopen

Set the process data failure time to zero:

Write 0 into object 0x27F8 sub 0

```
T0 00:01:43,9766: 0x603  8-Data: 2b f8 27 00 00 00 00 00
R0 00:01:44,0105: 0x583  8-Data: 60 f8 27 00 00 00 00 00
```

Select the command source, P700=6

Write 6 into object 0x22BC* sub 0

```
T0 00:01:44,0716: 0x603  8-Data: 2b bc 22 00 06 00 00 00
R0 00:01:44,1494: 0x583  8-Data: 60 bc 22 00 00 00 00 00
```

Select the frequency setpoint, P1000=6

Write 6 into object 0x23E8* sub 0

```
T0 00:01:44,2375: 0x603  8-Data: 2b e8 23 00 06 00 00 00
R0 00:01:44,2642: 0x583  8-Data: 60 e8 23 00 00 00 00 00
```

Parameter P719 =0 so that P700 and P1000 are used

Write 0 into object 0x22CF* sub 0

```
T0 00:01:44,3815: 0x603  8-Data: 2b cf 22 00 00 00 00 00
R0 00:01:44,4114: 0x583  8-Data: 60 cf 22 00 00 00 00 00
```

Parameter P1501.0 = P2091.12 (12th bit of parameter 2091)

BICO connection allows the operating modes to be changed-over (selected)

CANopen object 0x6060 is activated

Write 082B000C into object 0x25DD* sub 0

```
T0 00:01:44,4977: 0x603  8-Data: 22 dd 25 00 0c 00 2b 08
R0 00:01:44,5263: 0x583  8-Data: 60 dd 25 00 00 00 00 00
```

Set parameter P1300 (control type) to 22 (closed-loop control without encoder)

Write 22 into object 0x2514* sub 0

```
T0 00:02:06,2779: 0x603  8-Data: 2b 14 25 00 16 00 00 00
R0 00:02:06,2977: 0x583  8-Data: 60 14 25 00 00 00 00 00
```

Parameter P2050.4 = P1503.0 BICO connection to the torque setpoint source

CANopen object 0x6071 is activated

Write 08020004 into object 0x25DF* sub 0

```
T0 00:02:06,3982: 0x603  8-Data: 22 df 25 00 04 00 02 08 R0
00:02:06,4125: 0x583  8-Data: 60 df 25 00 00 00 00 00
```

Changeover into the Profile Torque Mode:

Write 4 into object 0x6060 sub 0

```
T0 00:02:06,4915: 0x603  8-Data: 2f 60 60 00 04 00 00 00
R0 00:02:06,4925: 0x583  8-Data: 60 60 60 00 00 00 00 00
```

Check whether the changeover was successful...

Read 4 from object 0x6061 sub 0

```
T0 00:02:06,5915: 0x603  8-Data: 40 61 60 00 00 00 00 00
R0 00:02:06,5932: 0x583  8-Data: 4f 61 60 00 04 00 bf 84
```

Set the data transfer type of all PDOs to non-synchronous data transfer:

Write FF into object 0x1400 sub 2

```
T0 00:01:44,7392: 0x603  8-Data: 2f 00 14 02 ff 00 00 00
R0 00:01:46,4598: 0x583  8-Data: 60 00 14 02 00 00 00 00
```

Write FF into object 0x1404 sub 2

```
T0 00:01:46,5702: 0x603  8-Data: 2f 04 14 02 ff 00 00 00
R0 00:01:46,5988: 0x583  8-Data: 60 04 14 02 00 00 00 00
```

Write FF into object 0x1405 sub 2

```
T0 00:01:46,6895: 0x603  8-Data: 2f 05 14 02 ff 00 00 00
R0 00:01:46,7217: 0x583  8-Data: 60 05 14 02 00 00 00 00
```

Write FF into object 0x1800 sub 2

```
T0 00:01:46,7989: 0x603  8-Data: 2f 00 18 02 ff 00 00 00
R0 00:01:48,4259: 0x583  8-Data: 60 00 18 02 00 00 00 00
```

Write FF into object 0x1804 sub 2

```
T0 00:01:48,5405: 0x603  8-Data: 2f 04 18 02 ff 00 00 00
R0 00:01:50,2609: 0x583  8-Data: 60 04 18 02 00 00 00 00
```

Write FF into object 0x1805 sub 2

```
T0 00:01:50,3706: 0x603  8-Data: 2f 05 18 02 ff 00 00 00
R0 00:01:50,4000: 0x583  8-Data: 60 05 18 02 00 00 00 00
```

Execute the mapping for RPDO 4 - set the number of mapped objects to 0

Write 0 into object 0x1604 sub 0

```
T0 00:01:50,4900: 0x603  8-Data: 2f 04 16 00 00 00 00 00
R0 00:01:50,5034: 0x583  8-Data: 60 04 16 00 00 00 00 00
```

The control word is the first mapped object:

Write 0x60400010 into object 0x1604 sub 1

```
T0 00:01:50,6068: 0x603  8-Data: 23 04 16 01 10 00 40 60
R0 00:01:50,6205: 0x583  8-Data: 60 04 16 01 00 00 00 00
```

The target torque is the second mapped object:

Write 0x60710010 into object 0x1604 sub 2

```
T0 00:02:12,4968: 0x603  8-Data: 23 04 16 02 10 00 71 60
R0 00:02:12,4982: 0x583  8-Data: 60 04 16 02 00 00 00 00
```

Set the number of mapped objects to 2

Write 2 into object 0x1604 sub 0

```
T0 00:01:50,8290: 0x603  8-Data: 2f 04 16 00 02 00 00 00
R0 00:01:52,4893: 0x583  8-Data: 60 04 16 00 00 00 00 00
```

5.8.1.1 Execute the mapping for TPDO 4

Set the number of mapped objects to 0

Write 0 into object 0x1A04 sub 0

```
T0 00:01:52,5891: 0x603  8-Data: 2f 04 1a 00 00 00 00 00
R0 00:01:52,6020: 0x583  8-Data: 60 04 1a 00 00 00 00 00
```

The status word is the first mapped object:

Write 0x60410010 into object 0x1A04 sub 1

```
T0 00:01:52,6976: 0x603  8-Data: 23 04 1a 01 10 00 41 60
R0 00:01:52,7110: 0x583  8-Data: 60 04 1a 01 00 00 00 00
```

The actual torque is the second mapped object:

Write 0x60770010 into object 0x1A04 sub 2

```
T0 00:02:14,5797: 0x603  8-Data: 23 04 1a 02 10 00 77 60
R0 00:02:14,5805: 0x583  8-Data: 60 04 1a 02 00 00 00 00
```

Set the number of mapped objects to 2

Write 2 into object 0x1A04 sub 0

```
T0 00:01:52,9377: 0x603  8-Data: 2f 04 1a 00 02 00 00 00
R0 00:01:54,5864: 0x583  8-Data: 60 04 1a 00 00 00 00 00
```

5.8.1.2 Execute the mapping for RPDO 5

Set the number of mapped objects to 0

Write 0 into object 0x1605 sub 0

```
T0 00:01:54,6741: 0x603  8-Data: 2f 05 16 00 00 00 00 00
R0 00:01:54,6864: 0x583  8-Data: 60 05 16 00 00 00 00 00
```

The control word is the first mapped object:

Write 0x60400010 into object 0x1605 sub 1

```
T0 00:01:54,7770: 0x603  8-Data: 23 05 16 01 10 00 40 60
R0 00:01:54,7894: 0x583  8-Data: 60 05 16 01 00 00 00 00
```

Modes of operation is the second mapped object:

Write 0x60600008 into object 0x1605 sub 2

```
T0 00:01:54,8523: 0x603  8-Data: 23 05 16 02 08 00 60 60
R0 00:01:54,8659: 0x583  8-Data: 60 05 16 02 00 00 00 00
```

Set the number of mapped objects to 2

Write 2 into object 0x1605 sub 0

```
T0 00:01:54,9624: 0x603  8-Data: 2f 05 16 00 02 00 00 00
R0 00:01:56,6833: 0x583  8-Data: 60 05 16 00 00 00 00 00
```

5.8.1.3 Execute the mapping for TPDO 5

Set the number of mapped objects to 0

Write 0 into object 0x1A05 sub 0

```
T0 00:01:56,7626: 0x603  8-Data: 2f 05 1a 00 00 00 00 00
R0 00:01:56,7748: 0x583  8-Data: 60 05 1a 00 00 00 00 00
```

The status word is the first mapped object:

Write 0x60410010 into object 0x1A05 sub 1

```
T0 00:01:56,8695: 0x603  8-Data: 23 05 1a 01 10 00 41 60
R0 00:01:56,8819: 0x583  8-Data: 60 05 1a 01 00 00 00 00
```

Modes of operation display is the second mapped object:

Write 0x60610008 into object 0x1605 sub 2

```
T0 00:01:56,9977: 0x603  8-Data: 23 05 1a 02 08 00 61 60
R0 00:01:57,0108: 0x583  8-Data: 60 05 1a 02 00 00 00 00
```

Set the number of mapped objects to 2

Write 2 into object 0x1A05 sub 0

```
T0 00:01:57,1004: 0x603  8-Data: 2f 05 1a 00 02 00 00 00
R0 00:01:58,7806: 0x583  8-Data: 60 05 1a 00 00 00 00 00
```

Set the nodes into the operational mode:

```
T0 00:01:58,8546: 0x000  2-Data: 01 03
```

Receive the three PDOs:

```
R0 00:01:58,8655: 0x183  2-Data: 31 fe
R0 00:01:58,8741: 0x283  4-Data: 31 fe 00 00
R0 00:01:58,8816: 0x383  3-Data: 31 fe 04
```

Run-through the drive state machine:

Send 6 using PDO 1

```
T0 00:01:58,9832: 0x203  2-Data: 06 00
```

Send 7 using PDO 1

```
T0 00:01:59,0533: 0x203  2-Data: 07 00
```

Receive the three PDOs:

```
R0 00:01:59,0953: 0x183  2-Data: 33 fe
R0 00:01:59,1038: 0x283  4-Data: 33 fe 00 00
R0 00:01:59,1113: 0x383  3-Data: 33 fe 04
```

Send F using PDO 1

```
T0 00:01:59,1569: 0x203  2-Data: 0f 00
```

Receive the three PDOs:

```
R0 00:01:59,1942: 0x183  2-Data: 37 fe
R0 00:01:59,2027: 0x283  4-Data: 37 fe 00 00
R0 00:01:59,2103: 0x383  3-Data: 37 fe 04
```

The drive is now in the "operational" state

Enter torque 200 and control word 7F using PDO4

```
-> the drive rotates T0 00:02:21,0924: 0x303  4-Data: 7f 00
c8 00
Receive e.g. the following PDOs:  R0 00:02:21,0954: 0x283  4-
Data: 37 fe 00 00
R0 00:02:21,1035: 0x283  4-Data: 37 fe 00 00
R0 00:02:21,1115: 0x283  4-Data: 37 fe 00 00
R0 00:02:21,1196: 0x283  4-Data: 37 fe ff ff
R0 00:02:21,1277: 0x283  4-Data: 37 fe b5 00
R0 00:02:21,1357: 0x283  4-Data: 37 fe fb 00 R0
00:02:21,1438: 0x283  4-Data: 37 fe 08 01
R0 00:02:21,1518: 0x183  2-Data: 37 fa
R0 00:02:21,1520: 0x283  4-Data: 37 fa 10 01
R0 00:02:21,1523: 0x383  3-Data: 37 fa 04
```

5.8.2 Additional helpful commands for the profile torque mode

Setting the baud rate:

Write 500 into object 0x27DA sub 0

```
T0 00:00:36,4500: 0x603  8-Data: 2b da 27 00 f4 01 00 00
R0 00:00:38,2083: 0x583  8-Data: 60 da 27 00 00 00 00 00
```

The new baud rate becomes active after the drive has restarted

Setting the NodeID (CANopen node name):

Write 4 for NodeID=4 into object 0x2396 sub 0

```
T0 00:00:19,0341: 0x603  8-Data: 2f 96 23 00 04 00 00 00
R0 00:00:19,0540: 0x583  8-Data: 60 96 23 00 00 00 00 00
```

The new node name becomes active after the drive has restarted

Save the CANopen parameters with "SAVE", object 0x1010:

Write 0x65766173 into object 0x1010 sub 1

```
T0 00:00:19,1443: 0x603  8-Data: 23 10 10 01 73 61 76 65
R0 00:00:38,1658: 0x583  8-Data: 60 10 10 01 00 00 00 00
```

Check the PDO mapping using MICROMASTER parameter P2041 (0x27F9*):

The data apply to the mapping set above:

Read FF FF from object 0x27F9 sub 0

```
T0 00:02:52,5010: 0x604  8-Data: 40 f9 27 00 00 00 00 00
R0 00:02:52,5160: 0x584  8-Data: 4b f9 27 00 ff ff 00 00
```

Read 00 FF from object 0x27F9 sub 1:

```
T0 00:02:52,5946: 0x604  8-Data: 40 f9 27 01 00 00 00 00
R0 00:02:52,6144: 0x584  8-Data: 4b f9 27 01 ff 00 00 00
```

Read 11 11 from object 0x27F9 sub 2:

```
T0 00:02:52,6746: 0x604  8-Data: 40 f9 27 02 00 00 00 00
R0 00:02:52,6962: 0x584  8-Data: 4b f9 27 02 11 11 00 00
```

Read 41 41 from object 0x27F9 sub 3:

```
T0 00:02:52,7545: 0x604  8-Data: 40 f9 27 03 00 00 00 00
R0 00:02:52,7699: 0x584  8-Data: 4b f9 27 03 41 41 00 00
```

6 Commissioning using a commissioning (start-up) tool

6.1 Assigning process data

The PZD data received (consumed) from the CANopen master is located in a specific data area in the MICROMASTER 420/430/440. For reasons of flexibility, this incoming PZD data must be allocated a purpose of use – for instance, the second word can be used as speed setpoint. Using parameters P0700 and P1000 to allocate, it is set so that the control word is received via P2050.00 and the main setpoint via P2050.01. The required BICO connections are also made for this purpose.

The PZD data, transferred back to the CANopen master (produced), come from another data area which is specifically reserved in MICROMASTER 420/430/440 for outgoing data. This is also done for reasons of flexibility. Every outgoing PZD data word (16 bit) must be allocated internal status words and actual value words, which are already available in MICROMASTER 420/430/440. This means that the second word, sent back to the CANopen master, can, for example, include the speed actual value. The indexed parameter P2051 is used to make this selection.

NOTE

Only 4 PZD words are shown for the MICROMASTER 420 in Table 6-1. This corresponds to the maximum capacity of the MICROMASTER 420. MICROMASTER 430/440 have higher capacities. The CANopen option module can process up to 6 PZD words.

Parameters "P0700" and "P1000" (selected via BICO)

As described above, control word and setpoint source can be quickly selected using parameter P0700 (this selects the command source) and P1000 (this selects the frequency setpoint).

P0719 must be set to 0 if the BICO technology is used with "P0700" and "P1000".

Parameters "r2050" and "P2051" (BICO)

An extremely high degree of flexibility is achieved by interconnecting the process data using binectors/connectors. More detailed information on this is provided in the Section "Binectors and connectors" in the Operating Instructions for MICROMASTER 420/430/440.

The connections of the various setpoints and actual values to and from the CANopen master via the CANopen option module are defined in "r2050" and "P2051".

The following tables contain the specific parameters which are used to connect process data for the CANopen option module:

Table 6-1 BICO connections for P2050 (PZD from CB)

Parameter	Data	Comment	BICO connection
2050.00	Control word 1	From CANopen control word in object 6040H	Is copied to r2090 in the MICROMASTER. Parameter P0700 to 6 sets all BICO connections.
2050.01	Vl_target_velocity/ vl_nominal _percentage	From CANopen vl_target_velocity in object 6042H or from CANopen vl_nominal_percentage in 6052H (depends on mapping selection, see 5.6)	BICO connection to P1070. Parameter 1000 to 6 sets the required BICO connection
2050.02	User defined	From CANopen free object 2802H subindex 3	BICO connection can be made user-specific
2050.03	User defined	From CANopen free object 2802H subindex 4 or from CANopen object Modes of Operation (MICROMASTER 440) NOTE If Modes of Operation is mapped, only Bit12 of 2050.03 is affected. All the other bits can be freely used. User can map free object 2802H subindex 4 AND Modes of Operation. In this case, user can freely change bits 0 to 11 and bits 13 to 15 in the free object but bit 12 will only be affected by Modes of Operation.	BICO connection can be made user-specific. Is copied to P2091 in the MICROMASTER. In the case of the MICROMASTER 440 and use of the Modes_of _operation object, bit 12 must be BICO connected with P1501. All other bits can be connected freely in BICO.
Next entries (only for MICROMASTER 440)			
2050.04	Target torque	From CANopen Target_Torque (6071H)	BICO connection to the momentary target value source (see 0)
2050.05	User defined	Map to Free Object (2802H subindex 6)	BICO connection can be made user-specific
2050.03	Modes of operation	From CANopen object modes of operation (6060H)	When using the object Modes_of_operation, bit 12 must be BICO connected with P1501.

NOTE

P2050.00 (Control word 1) is internally copied to parameter r2090 and that P2050.03 (Free Object) is internally copied into parameter r2091. This is to enable the BICO connections to be made to the individual bits within these control words. Note also that modes of operation operates on bit 12 of r2050.03.

Process data is sent from the drive to the network in parameter P2051 as follows:

Table 6-2 BICO connections for P2051 (PZD to CB)

Parameter	Data	Comment	BICO connection
2051.00	Status word 1	To CANopen status word object 6041H	BICO connection from r0052. Parameter 700 to 6 sets the required BICO connection
2051.01	Vl_velocity_demand	To CANopen vl_velocity_demand object 6043H and to CANopen vl_percentage_demand object 6053H (depends on mapping selection, see 5.6)	BICO connection from r0021. Parameter 1000 to 6 sets the required BICO connection
2051.02	User defined	To CANopen free object 2803H subindex 3	BICO connection can be made user-specific
2051.03	User defined	To CANopen free object 2803H subindex 4	BICO connection can be made user-specific
Next entries (only for MICROMASTER 440)			
2051.04	Actual_torque	To CANopen object torque actual value 6077H	BICO connection from r0080.
2051.05	Actual speed	To CANopen vl_control_effort in object 6044H and to vl_actual_percentage in object 6054H (depends on mapping selection, see 5.6)	Read from r0063 in VC and SLVC modes. Read from r0021 in VF modes. Default = 0
2051.06	Modes of operation display	To CANopen modes of operation display 6061H	BICO connection from r0055

NOTE

r2050 also acts as a display parameter with which the setpoints, received from the CANopen master, can be checked. These displays always have a decimal format – if, e.g. r2050.00 = 1150, then this would be 047EH in the hexadecimal format.

6.2 Parameter "P0927" – change source for parameters

This parameter can be set in order to define the sources of the parameter changes.

Bit 0	CANopen master	0: No 1: Yes
Bit 1	BOP	0: No 1: Yes
Bit 2	Mounting set PC/drive converter (USS at the BOP interface)	0: No 1: Yes
Bit 3	Local RS485 interface (terminal 14/15 (MICROMASTER 420) and 29/30 (MICROMASTER 440) with USS)	0: No 1: Yes

The pre-setting (default setting) for all bits is 1; this means that the parameters of all of the sources can be changed.

6.3 Parameter settings for the modes

Fig. 6-1 to Fig. 6-4 give an overview to the process data connections and show appropriate BICO connections for velocity mode and torque profile mode.

6.3.1 Velocity Mode

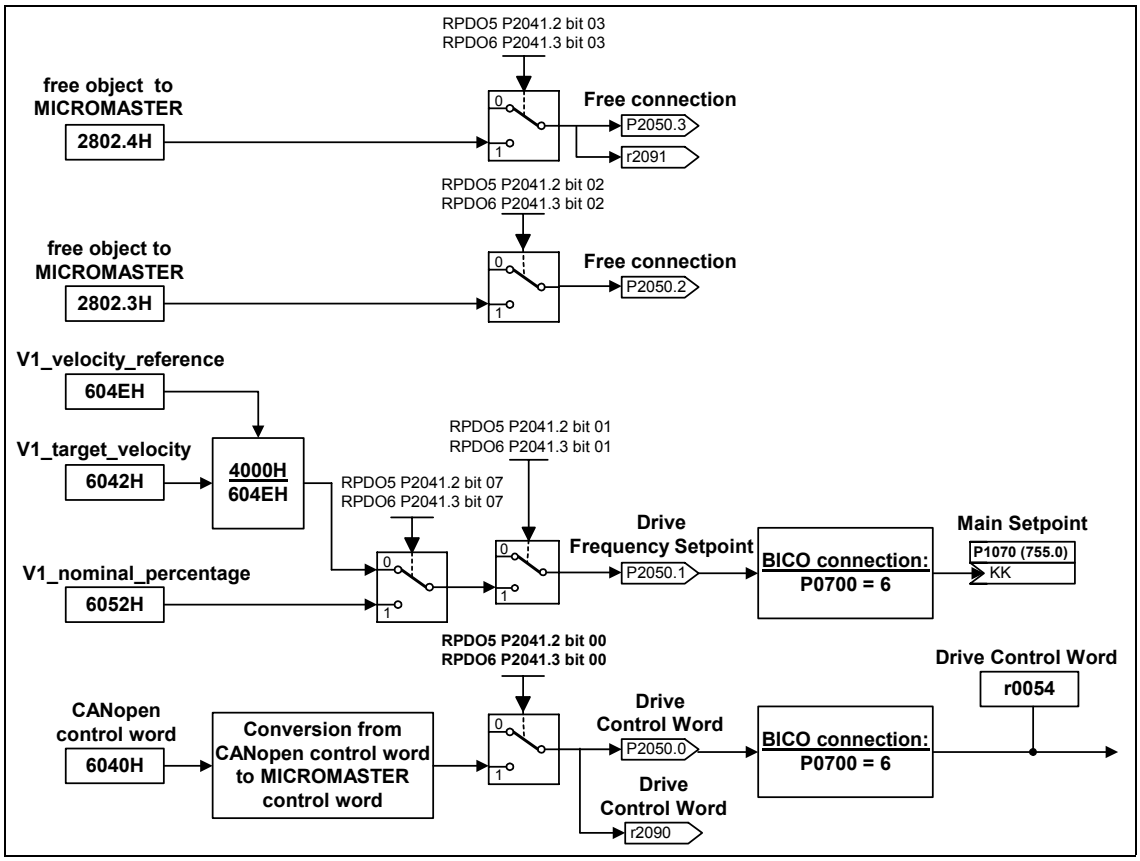


Fig. 6-1 Process Data transfer from CAN object dictionary to MICROMASTER 420/430/440 using RPDO5 or RPDO6

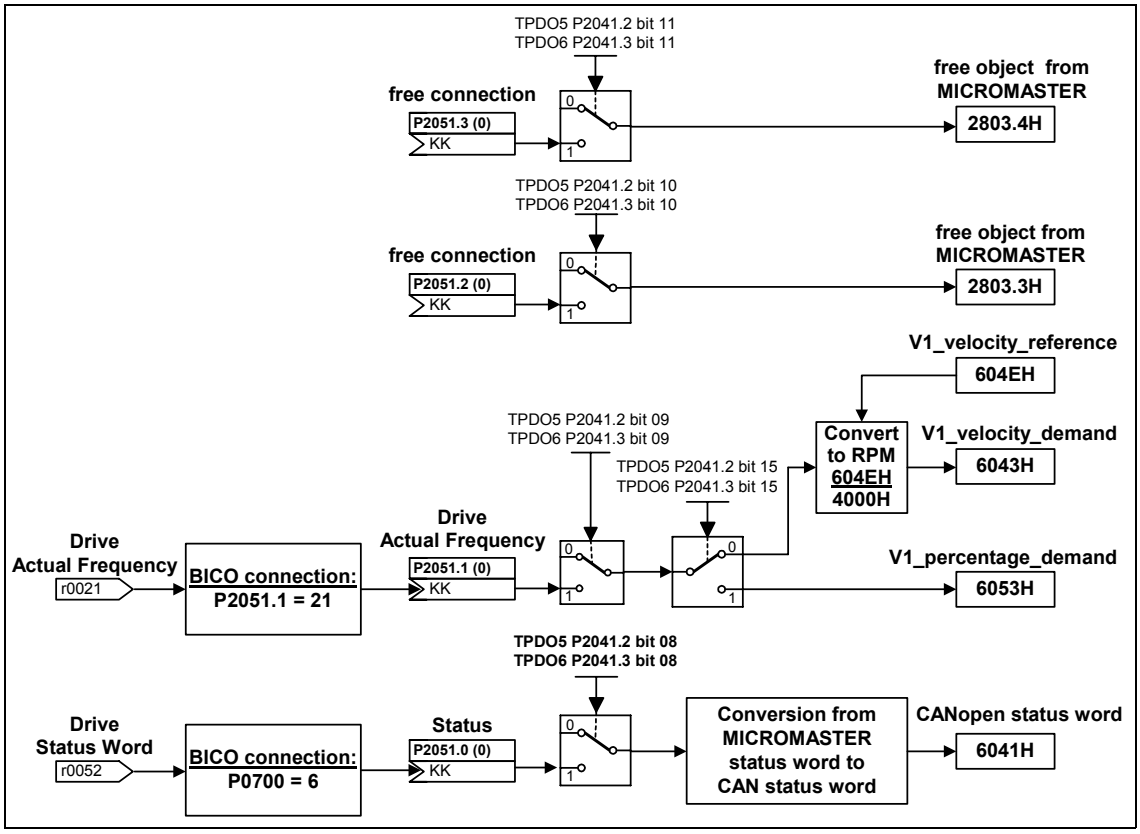


Fig. 6-2 Process Data transfer from MICROMASTER 420/430/440 to CAN object dictionary using TPDO5 or TPDO6

Typical parameter edits for the Velocity Mode

Parameter	Value	Comment
2041.0	65535	PDO1, PDO5 Transmit if data changes
2041.1	0 * no change required by user	0 = default value on powerup, so user does not need to change this. RPDO1, 5 and 6 are asynchronous TPDO6 = Acyclic, synchronised, but not mapped (not needed for minimal system)
2041.2	0x0303H = 771 decimal	RPDO5: Control Word Setpoint in RPM TPDO5: Status Word Actual Speed in RPM
2041.3	0 * no change required by user	0 = default value on powerup, so user does not need to change this. No mapping for pdo6 (not needed for minimum system)
2041.4	110 decimal	Baudrate 20 kb/s On comm loss, trip drive if possible On comm loss, switch to pre-operational state.
P0918	Set to drive address Default = 3	Each drive on the network must have a unique address
BICO Connections		
P0700	6	Control from CAN
P1000	6	Setpoint from CAN
P2051.1	21	Read back actual speed
P2051.6	55	Connection for modes of operation display

NOTE

Changes to CAN configuration parameters using BOP, AOP, Start-up Tool (STARTER, DriveMonitor) will only take effect after a powercycle.

Changes to CAN configuration parameters made over the CAN network via the object dictionary will take immediate effect (powercycle not necessary).

6.3.2 Profile Torque Mode

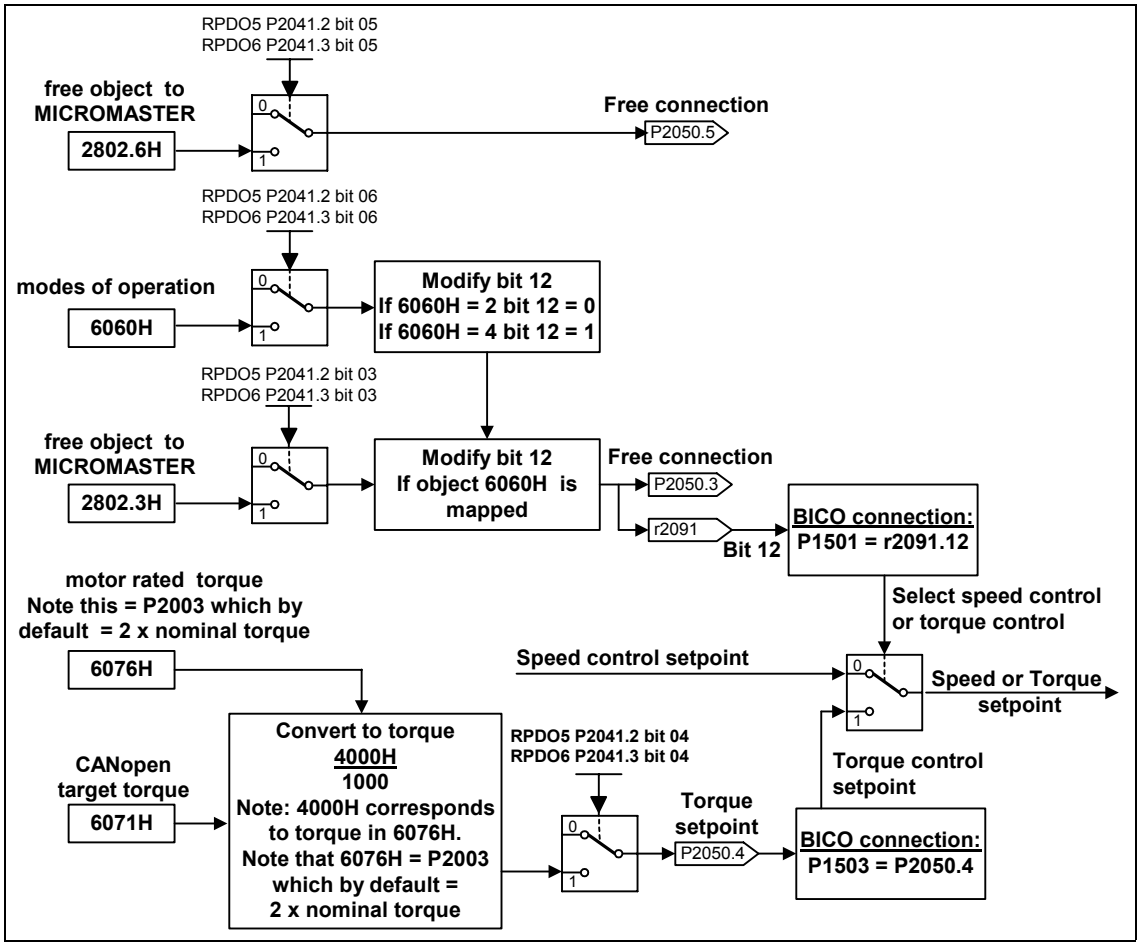


Fig. 6-3 Process Data transfer from CAN object dictionary to MICROMASTER 440 using RPDO5 or RPDO6

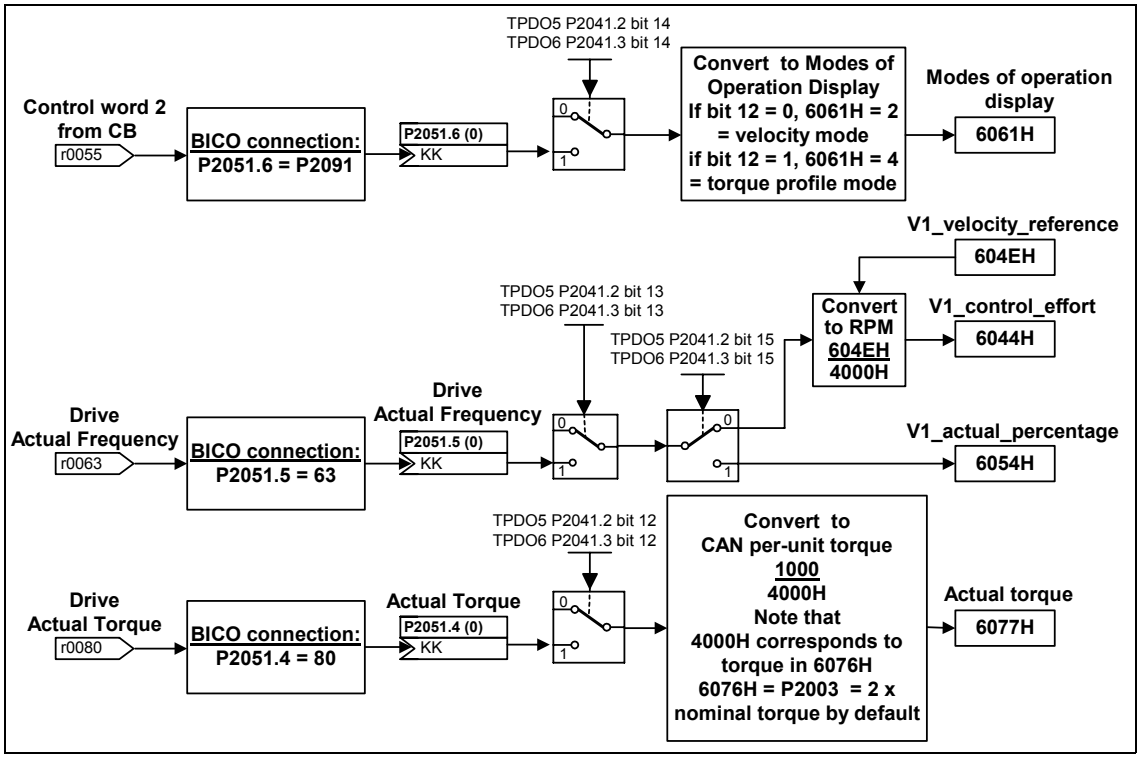


Fig. 6-4 Process Data transfer from MICROMASTER 440 to CAN object dictionary using TPDO5 or TPDO6

Typical parameter edits for the Profile Torque Mode allowing switchover between speed and torque control

Parameter	Value	Comment	
2041.0	65535	PDO1, PDO5 Transmit if data changes	
2041.1	0 * no change required by user	0 = default value on powerup, so user does not need to change this. RPDO1, 5 and 6 are asynchronous TPDO6 = Acyclic, synchronised, but not mapped (not needed for minimal system)	
2041.2	01110010 01010010 hex = 7252H = 29266 decimal	RPDO5 : Setpoint in RPM Torque setpoint Modes of operation	TDO5: Actual speed in RPM Actual torque Actual speed in RPM Modes of operation display
P1503	2050.4	Connect torque setpoint from CAN	
P1501	2091.12	Connect modes of operation to allow switching between speed control and torque control	
P2051.4	80	Connect actual torque	
P2051.5	63	Connect actual drive speed. This is not necessary for torque profile, but it does give the actual motor speed as shown by an (optional) attached encoder	
P2051.6	2091	Connection for modes of operation display	

NOTE

Changes to CAN configuration parameters using BOP, AOP, start-up tool (STARTER, DriveMonitor) will only take effect after a powercycle.

Changes to CAN configuration parameters made over the CAN network via the object dictionary will take immediate effect (powercycle not necessary).

7 Diagnostics and troubleshooting

A standard combi LED is provided for troubleshooting. This combi LED displays the module/network status and, alarms generated specifically from the CANopen module and a diagnostics display parameter.

7.1 LED display

A two-color LED is provided on the front panel of the CANopen module. This displays the operating status of the module.

An LED test is carried-out when the module is powered-up. The test proceeds as follows: The LED is lit green for approximately 1 second and then it is red for 1 second and then it goes dark.

The following table explains the various states of the combi LED for the module/network status.

Table 7-1 LEDs display types

LED	Action
On	LED colour constantly ON
Off	LED colour constantly OFF
Flickering	iso phase on and off with a frequency of about 10 Hz (approximately 50 ms on / 50 ms off)
Blinking	iso phase on and off with a frequency of about 2.5 Hz (approximately 200 ms on / 200 ms off)
Single flash	One short 200 ms flash followed by off for 1 s
Double Flash	Two short 200 ms flashes followed by off for 1 s
Triple Flash	Three short 200 ms flashes followed by off for 1 s

In the CANopen specification the RED part of the LED indicates the status of the CAN physical layer and indicates errors due to missing CAN messages (SYNC, NODE-GUARDING, etc).

The MICROMASTER 420/430/440 will support the following **mandatory** states of the RED part of the LED and an MICROMASTER 420/430/440 specific option as follows:

Table 7-2 Status of the red LED display

No.	RED part of LED	State	Description
1	Off	No error	The device is working normally
2	Single flash	Warning limit reached	At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames).
3	Double flash	Error control event	A node-guard event
4	Triple flash	Sync Error	The sync message has not been received within the configured communication cycle period timeout (see object dictionary entry 1006H).
5	On	Bus Off	The CAN controller is Bus-Off Or Fatal Error on option card
6	Blinking	Fatal error	MICROMASTER specific: Fatal error on Card.

In the CANopen specification the GREEN part of the LED indicates the status of the CANopen network state machine.

The MICROMASTER 420/430/440 will support the following **mandatory** states of the GREEN part of the LED as follows:

Table 7-3 Status of the green LED display

No.	GREEN part of LED	State	Description
1	Off	Executing a reset / Fatal error	The device is executing a RESET Or (MICROMASTER specific) option module fatal error
2	Single flash	Stopped	The device is in the STOPPED state
3	Blinking	Pre-operational	The device is in the PRE-OPERATIONAL state
4	On	Operational	The device is in the OPERATIONAL state

Note that the above effects of the two colours in the LED are combined to give a total status. For example ‘CAN warning limit reached’ during the ‘pre-operational’ state would give a 200 ms flash of the RED LED every second interspersed with a 2.5 Hz signal on the GREEN LED.

NOTE

Orange colour LED is not critical.

7.2 Alarms (warnings and faults)

If faults/errors occur in the CANopen communications, the appropriate alarm or fault number is output at the drive converter/drive inverter.

Warnings

Table 7-4 Warnings displayed at the drive converter

Alarm No.	Description
A0700	**Can Chip Passive Warning** Erroneous CAN messages are received or sent and the internal fault counter has exceeded 128. **Extra Information** The erroneous CAN messages are not used. Data which was last sent remains valid. If the erroneous data is process data, depending on the telegram failure monitoring setting (P2040) , fault F0070 can be initiated. There is no response in the Drive if the parameter access messages are erroneous. The red error LED of the CANopen module is single flashing. This warning means that there are problems with the communication through electrical noise or other problems. Note that the health of the link can be determined by monitoring the number of message errors in the message error counter in r2054.06. See description of r2054.06 for further information. The message error counter should only increment occasionally. This warning will clear automatically when the number of error messages on the bus reduces sufficiently. **Remedy:** • Check and if required correct most significant digit of P2041.04 (baud rate) for every bus node • check cable connection between bus nodes • check cable shielding • replace the CANopen module
A0701	**CAN chip or CAN buffer overflow** Data has been received too quickly to be processed. **Extra Information** If the CAN chip buffer is full, or the software message buffers are full, then any further received messages will be lost until the option module and drive have had time to process messages from the buffers. The user has the option to ignore this warning, stop the drive, or to force a drive trip in response to this condition. See note 2 for further information. **Remedy:** The cause of this error is that the bus loading is too high. Reduce the rate at which messages are sent to the MICROMASTER 420/430/440. Allow at least 1ms between 2 consecutive messages to the MICROMASTER 420/430/440. Note that within the drive CANopen module PDO message data is transferred to the drive every 8 ms, so writing to the same location (i.e. writing the speed set point) more often than every 8 ms will generally cause the earlier data to be overwritten. Note that PDO that write to different objects may be sent more often than 8 ms.

Alarm No.	Description
A0702	**Can Chip Bus-Off.** Erroneous CAN messages are received or sent and the internal fault counter has exceeded the error limit. **Extra Information** The erroneous CAN messages are not used. Data which was last sent remains valid. If the erroneous data is process data, depending on the telegram failure monitoring setting (P2040) and other configuration settings, fault F0070 can be initiated (see note 1). There is no response in the Drive if the parameter access messages are erroneous. The red error LED is continued on (see section 7.1). This warning means that there are problems with the communication through electrical noise or other problems. Note that the health of the link can be determined by monitoring the number of message errors in the message error counter in r2054.06. See description of r2054.06 for further information. The message error counter should only increment occasionally. The chip will automatically return to the non-busoff state as soon as bus-conditions permit, and will send a recovered from bus-off emcy message to the CAN network. If the drive has tripped due to the bus-off condition (see note 1) then the drive will need to be reset before it can control the motor again. **Remedy:** • Check and if required correct most significant digit of P2041.04 (baud rate) for every bus node • check cable connection between bus nodes • check cable shielding • replace the CANopen module
A0703	**NodeGuarding failure** MICROMASTER 420/430/440 has not been guarded from the CAN Master within the guard time x the lifetime factor. **Further Information:** If P2040 > 0, then node guarding is enabled. By default the guard time (object 100CH) will be set to P2040 / 2 ms and the lifetime factor (object 100DH) will be set to 2, although these values can be changed from CAN using SDO messages. In response to a node-guarding error the user has options to ignore the error, trip the Drive, stop the drive, and/or send an emcy message. See note 1. The red error LED is double flashing (see section 7.1). **Remedy:** • Check health of the bus (is error counter in r2054.06 incrementing frequently?). If bus is unhealthy, then make checks described for A0700. • Increase guard time in object 100CH or increase lifetime factor in 100DH • Increase value in P2040 • Set P2040 = 0 (turn off nodeguarding)
A0704	**Sync Error** MICROMASTER 420/430/440 has not received a sync signal within 1.5 times the communication cycle period. **Further Information:** If object 1006H > 0, then sync checking is enabled on the MICROMASTER 420/430/440. If a sync is not received within 1.5 time the communication cycle period then this warning will occur. In response to a sync loss the user has options to ignore the error, trip the Drive, stop the drive, and/or send an emcy message. See note 1. This warning will clear itself as soon as the software sees 2 sync signals within a communication cycle period. If the drive has tripped due to the sync-loss condition (see note 1) then the drive will need to be reset before it can control the motor again. The red error LED is triple flashing (see section 7.1). **Remedy:** • Check health of the bus (is error counter in r2054.06 incrementing frequently?). If bus is unhealthy, then make checks described for A0700. • Increase communication time in object 1006H • Set 1006H = 0 (turn off sync-loss check)
A0705	**Cause:** The status and actual value data of the drive converter is lost **Remedy:** None (fault in the drive converter)

Alarm No.	Description
A0706	**Cause:** Software error in the CANopen module **Remedy:** None (fault on the CANopen module – more detailed information is provided in Section 7.3, Diagnostic parameters)
A0707	**Cause:** Data set changed, but failed to update parameter accordingly. See also 2054[0]. **Remedy:** Repeat change of dataset
A0710	**Cause:** The drive converter detects an error in the communications with the CANopen module **Remedy:** None (the CANopen module must be replaced)
A0711	**Cause:** The parameters of the communications module, which define the CANopen connection were incorrectly set. The fault was detected while the CANopen module was being initialized. **Supplementary information:** Refer to the diagnostic parameters r2054.00 (Section 7.3). **Remedy:** Correct P0918 (CANopen node ID) and/or P2041 (CB parameter), indices 00 to 02.

Note 1: Objects 1029H and 6007H determine CANopen module response to most of the alarms described above.

1029H is set up from the least significant digit of P2041.04

6007H is set up from the middle digit of P2041.04

Note 2:

CAN Chip: The CAN chip is read on interrupt, and interrupt routine takes approx. 20 to 50 µs. Therefore if 3 messages received within 40 to 100 µs will get CAN overrun.

CAN software: The CAN software has 10 buffers. These buffers are read every 2 ms. Therefore if receive > 10 messages in 2 ms may get a CAN overrun.

Drive Cycle Time: This is 8 ms. If the MICROMASTER 420/430/440 receives the data for the same variable within 8 ms the previous data will be overwritten.

Faults

Table 7-5 Fault messages displayed at the drive converter

Fault No.	Description
F0070	**Cause:** The communications failure monitoring time, set in parameter P2040 has expired. The drive monitors this time. **Supplementary information:** Also refer to alarms A0700, A0701, A0702. **Remedy:** ➢ Check whether the CANopen master has stopped operation. ➢ Check the cable connections between the bus nodes. ➢ Check the cable shield. Ensure that the specifications in the CAN specification have been observed. ➢ Check whether the communications monitoring time was set too short in parameter P2040. ➢ Replace the CANopen master.

7.3 Diagnostic parameters

Parameter r2054

The indexed parameter r2054 provides detailed information if the MICROMASTER displays a warning (alarm) generated by the CANopen option module.

Index	Content
0	**Configuration or other errors** 0: no error 1: Invalid slave address (low byte of P0918) 2: Invalid PDO Transmission Type (P2041.00 - P2041.01) 3: Invalid T_PDO Transmission Type (P2041.00 - P2041.01) 4: Invalid R_PDO Transmission Type (P2041.01 high byte) 5: Invalid baud rate (P2041.04) 6: Invalid comm error response (P2041.04) 7: Invalid lifeguarding event response (P2041.04) 8 - 10: Unable to read configuration parameters : problem on comm link to Drive 12: Configuration timed out 13: Other configuration error 14 - 15: Configuration Mapping error (P2041.02 – P2041.03) 16: illegal drive type 32: wrong index passed by SDO 33: SDO wrong profile 35 - 36: Failure in writing Parameters to Drive on reset-node 64 - 66: Execution of resetCommunication command failed 67 - 69 Execution of resetNode command failed 96 - 101 Execution of “load” command failed 112 - 113 Execution of “save” failed 114 Tried to do resetcommunication or resetnode whilst drive running 128 - 129 Data set changed, but failed to update parametes accordingly 146 Tried to do resetnode or resetcommunication when data saved to eeprom or when already executed a “load” command **CANopen errors** 401: Out of RAM 402: Object does not exist 404: Operation not allowed in this state 405: No matching type 406: Inhibit time active 407: No initiate service executed 408: Service is already running 409: Datatype does not fit in telegram 410: Errortype does not fit in telegram 412: Invalid range 415: No COB database available 416: Object disabled 420: PDO Mapping error 421: No access to object dictionary 422: Object does not exist 423: Subindex doesn't exist 424: No read permission 425: No write permission 426: Value greater than upper limit 427: Value less than lower limit 428: Object has wrong size 429: Wrong trans type

	430: Hardware fault 431: Parameter incompatible 432: Unknown SDO error 433: SDO command specifier invalid 434: Invalid SDO block size 435: Invalid SDO block crc sum 436: No resources available for SDO connection 437: Bad requested error control mechanism 438: SDO timed out 439: SDO invalid togglebit 440: SDO invalid transmode 441: Bad device state 442: Bad CRC
1	Status of CAN physical layer: Bits representing CAN errors (because there may be multiple errors) 0: no errors Bit 0: not used Bit 1: CAN error passive (CAN error counter(s) >=127, too many error frames) Bit 2: CAN overflow (too many can messages in a too short time) Bit 3: error control (Node-guarding or Heartbeat error event) Bit 4: Sync error (Sync object not received within configured communication cycle) Bit 5: Bus-Off (CAN controller in Bus-Off state or fatal error on option card)
2	Status of the CANopen network state-machine: 0: CANopen in initialization state 1: Executing a Reset or Fatal Error 2: CANopen in pre-operational state 3: CANopen in stopped state 4: CANopen in operational state
3	counter for PDO telegrams received error-free since power on.
4	counter for telegrams transmitted since power on.
5	counter for all telegrams received error-free since power-on. this counter includes all CanBus messages, including those not directed to this module.
6	CAN Error counter and Fatal Code Value > 32767: Fatal Error Code. 0 – 32767 error counter. Counter wraps around back to 1 when count > 32767. Increments on any CAN passive error, CAN busoff error or CAN overflow error. If this value is changing rapidly and warning = A0701 then the CAN bus is overloaded. • Reduce Bus loading If this value is changing rapidly and warning = A0700 or A0702 then the CAN bus is not healthy. Check and if required correct most significant digit of P2041.04 (baud rate) for every bus node • check cable connection between bus nodes • check cable shielding • Reduce Bus loading • replace the CANopen option module

7.4 Software release and information

The software release and other software information on the communications module are displayed in the indexed parameter r2053.

Parameter	Description
r2053.00	Module type (3 = CANopen)
r2053.01	Version (11 = Version 1.1); this is the same version which is output at CANopen
r2053.02	Firmware details
r2053.03	Year of the software
r2053.04	Day / month (2404 = 24. April) of the software

8 Attachment

8.1 Technical data

Table 8-1 Technical data

Order No.	6SE6400-1CB00-0AA0
Size (Height x Width x Depth)	161 mm x 73 mm x 43.5 mm
Degree of pollution	Degree of pollution 2 according to IEC 60 664-1 (DIN VDE 0110/T1), moisture condensation during operation is not permissible
Mechanical strength Stationary - deflection - acceleration During transport - deflection - acceleration	Acc. to DIN IEC 60 068-2-6 (with a correctly installed module) 0.15 mm frequency range between 10 Hz and 58 Hz 19.6 m/s^2 frequency range from > 58 Hz to 500 Hz 3.5 mm frequency range between 5 Hz and 9 Hz 9.8 m/s^2 frequency range from > 9 Hz to 500 Hz
Climatic Class (in operation)	Class 3K3 according to DIN IEC 60 721-3-3
Cooling type	Air self-cooling
Permissible ambient or cooling-medium temperature - operation - storage - transport	 –10 °C to +50 °C (14 °F to 122 °F) –40 °C to +70 °C (–40 °F to 158 °F) –25 °C to +70 °C (–13 °F to 158 °F)
Relative air humidity (perm. humidity class) In operation During transport and storage	 <= 85 % rF (moisture condensation not permissible) <= 95 % rF
Power supply	Power is supplied to the CAN bus from the MICROMASTER's power supply
Output voltage	None
Data transfer rate	10, 20, 50, 125, 250, 500, 800 kbaud and 1Mbaud

8.2 EMC information

The module fulfills the following Standards regarding noise emission and noise immunity:

Noise emission according to EN55011 (1991) Class A

Noise immunity according to IEC 60 801-3 and EN61000-4-3

8.3 List of Abbriviations

AC	Alternating current
AD	Analog digital converter
ADC	Analog digital converter
ADR	Address
AFM	Additional frequency modification
AIN	Analog input
AOP	Advanced operator panel
AOUT	Analog output
ASP	Analog setpoint
ASVM	Asymmetric space vector modulation
BCC	Block check character
BCD	Binary-coded decimal code
BI	Binector input
BICO	Binector / connector
BO	Binector output
BOP	Basic operator panel
C	Commissioning
CAL	CAN Application Layer
CAN	Controller Area Network
CB	Communication board
CCS	Client Command Specifier
CCW	Counter-clockwise
CDS	Command data set
CI	Connector input
CiA	CAN in Automation
CM	Configuration management
CMD	Commando
CMM	Combimaster
CO	Connector output
CO/BO	Connector output / Binector output
COB-ID	Communication Object Identifier
COM	Common (terminal that is connected to NO or NC)
COM-Link	Communication link
CT	Commissining, ready to run
CT	Constant torque
CUT	Commissining, run, ready to run

CW	Clockwise
DA	Digital analog converter
DAC	Digital analog converter
DC	Direct current
DDS	Drive data set
DIN	Digital input
DIP	DIP switch
DOUT	Digital output
DS	Drive state
EEC	European Economic Community
EEPROM	Electrical erasable programmable read-only
ELCB	Earth leakage circuit breaker
EMC	Electro-magnetic compatibility
EMF	Electromotive force
EMI	Electro-magnetic interference
FAQ	Frequently asked questions
FCC	Flux current control
FCL	Fast current limit
FF	Fixed freqeuncy
FFB	Free function block
FOC	Field orientated control
FSA	Frame size A
GSG	Getting started guide
GUI ID	Global unique identifier
HIW	Main actual value
HSW	Main setpoint
HTL	High-threshold logic
I/O	Input and output
IBN	Commissioning
IGBT	Insulated gate bipolar transistor
IND	Subindex
JOG	Jog
KIB	Kinetic buffering
KTY	
LCD	Liguid crystal display
LED	Light emitting diode
LGE	Length
MHB	Motor holding brake
MM4	MICROMASTER 4th. Generation

MOP	Motor potentiometer
NC	Normally closed
NO	Normally open
Node-ID	Node address
NPN	
OPI	Operating instructions
PDO	Process Data Object
PDS	Power drive system
PID	PID controller (proportional, integral, derivative)
PKE	Parameter ID
PKW	Parameter ID value
PLC	Programmable logic controller
PLI	Parameter list
PNP	
PPO	Parameter process data object
PTC	Positive temperature coefficient
PWE	Parameter value
PWM	Pulse-width modulation
PNU	Parameter number
PX	Power extension
PZD	Process data
QC	Quick commissioning
RAM	Random-access memory
RCCB	Residual current circuit breaker
RCD	Residual current device
RFG	Ramp function generator
RFI	Radio-frequency interference
ro	read only
RPM	Revolutions per minute
RTR	Remote Transmission Request
rw	read/write
SCL	Scaling
SCS	Server Command Specifier
SDO	Service Data Object
SDP	Status display panel
SLVC	Sensorless vector control
STW	Control word
STX	Start of text
SVM	Space vector modulation

TTL	Transistor-transistor logic
USS	Universal serial interface
VC	Vector control
VT	Variable torque
wo	write only
ZSW	Status word

Recommendations and/or corrections

To Siemens AG Automation & Drives Group SD SM 5 P.O. Box 3269 D-91050 Erlangen Federal Republic of Germany Email: documentation.sd@siemens.com	**Recommendations** **Corrections** For the document/Manual: MICROMASTER 420/430/440 CANopen Option Module User Documentation
From Name: Company/Service Department Address: _____ _____ Telephone: _____ / _____ Fax: _____ / _____	Order No.: 6SE6400-5BC00-0BP0 Date published: 01/05 If, when reading this document, you come across printing mistakes, then please let us know using this form. We would also be grateful for any suggestions and recommendations for improvement.

Siemens AG
Automation and Drives Group (A&D)
Standard Drives (SD) Division
Postfach 3269, D-91050 Erlangen
Germany

Siemens Aktiengesellschaft

Order No.: 6SE6400-5BC00-0BP0
Printed in Germany