CANopen Fieldbus manual
for Lexium MDrive
V1.00, 09.2014

CANopen

CE RoHS REACH

Schneider Electric

CANopen Fieldbus Manual		
Date	**Revision**	**Changes**
07/03/2013	V1.00, 07.2013	Initial release
08/18/2014	V1.00, 08.2014	Updated to match firmware version 7.15. Notable changes include: hMT defaulted to an "off state, added a fourth RxPDO and TxPDO. Minor updates to indexes impacted by Firmware update.
09/22/2014	V1.00, 09.2014	Updated to match firmware version 7.16. Added Index 6077h: Demand Torque Actual Value.

Important information

This manual is part of the product.

Carefully read this manual and observe all instructions.

Keep this manual for future reference.

Hand this manual and all other pertinent product documentation over to all users of the product.

Carefully read and observe all safety instructions and the chapter "Before you begin - safety information".

Some products are not available in all countries.

For information on the availability of products, please consult the catalog.

Subject to technical modifications without notice.

All details provided are technical data which do not constitute warranted qualities.

Most of the product designations are registered trademarks of their respective owners, even if this is not explicitly indicated.

This page intentionally left blank

Table of Contents

iv

List of Figures

List of Tables

About this manual

The information provided in this manual supplements the product hardware manual.

Source manuals

The latest versions of the manuals can be downloaded from the Internet at:

http://motion.schneider-electric.com/

Source EDS files

For easier engineering, Electronic Datasheet Files and product master data are available for download from the Internet at:

http://motion.schneider-electric.com/

Graphic User Interface software

For easier configuration, a Graphic User Interface (GUI) is available for use with Lexium MDrive and Lexium MDrive product in conjunction with the optional MD-CC501-000 USB to CANopen interface cable kit. This software is available for download from the Internet at:

http://motion.schneider-electric.com/

Further reading

Recommended literature for further reading.

CAN users and manufacturers organization

CiA - CAN in Automation
Am Weichselgarten 26
D-91058 Erlangen

http://www.can-cia.org/

CANopen standards

- CiA Standard 301 (DS301)
  CANopen application layer and communication profile
- CiA Standard 402 (DSP402)
  Device profile for drives and motion control
- ISO 11898: Controller Area Network (CAN) for high speed communication
- EN 50325-4: Industrial communications subsystem based on ISO 11898 for controller device interfaces (CANopen)

V1.00, 09.2014

Page intentionally left blank

V1.00, 09.2014

1 Introduction

1.1 CAN bus

The CAN bus (Controller Area Network) was originally developed for fast, economical data transmission in the automotive industry. Today, the CAN bus is also used in industrial automation technology and has been further developed for communication at fieldbus level.

Features of the CAN bus   The CAN bus is a standardized, open bus enabling communication between devices, sensors and actuators from different manufacturers. The features of the CAN bus comprise

- Multi-master capability
  Each device in the fieldbus can transmit and receive data independently without depending on an "ordering" master functionality.

- Message-oriented communication
  Devices can be integrated into a running network without re-configuration of the entire system. The address of a new device does not need to be specified on the network.

- Prioritization of messages
  Messages with higher priority are sent first for time-critical applications.

- Residual error probability
  Various security features in the network reduce the probability of undetected incorrect data transmission to less than 10^{-11}.

Transmission technology   In the CAN bus, multiple devices are connected via a bus cable. Each network device can transmit and receive messages. Data between network devices are transmitted serially.

Network devices   Examples of CAN bus devices are

- Automation devices, for example, PLCs
- PCs
- Input/output modules
- Drives
- Analysis devices
- Sensors and actuators

1.2   CANopen technology

1.2.1   CANopen description language

CANopen is a device- and manufacturer-independent description language for communication via the CAN bus. CANopen provides a common basis for interchanging commands and data between CAN bus devices.

1.2.2   Communication layers

CANopen uses the CAN bus technology for data communication.

CANopen is based on the basic network services for data communication as per the ISO-OSI model model. 3 layers enable data communication via the CAN bus.

- Physical Layer
- Data Link Layer
- Application Layer
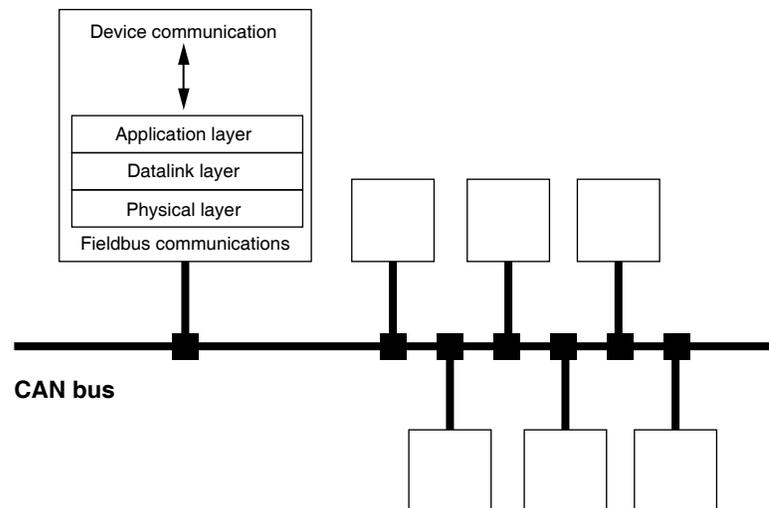
Figure 1.1: CANopen layer model

Physical Layer The physical layer defines the electrical properties of the CAN bus such as connectors, cable length and cable properties as well as bit coding and bit timing.

Data Link Layer The data link layer connects the network devices. It assigns priorities to individual data packets and monitors and corrects errors.

Application Layer The application layer uses communication objects (COB) to exchange data between the various devices. Communication objects are elementary components for creating a CANopen application.

1.2.3   Objects

Processes under CANopen are executed via objects. Objects carry out different tasks; they act as communication objects for data transport to the fieldbus, control the process of establishing a connection or monitor the network devices. If objects are directly linked to the device (device specific objects), the device functions can be used and changed via these objects.

Object dictionary   The object dictionary of each network device allows for communication between the devices. Other devices find the objects with which they can communicate in this dictionary.
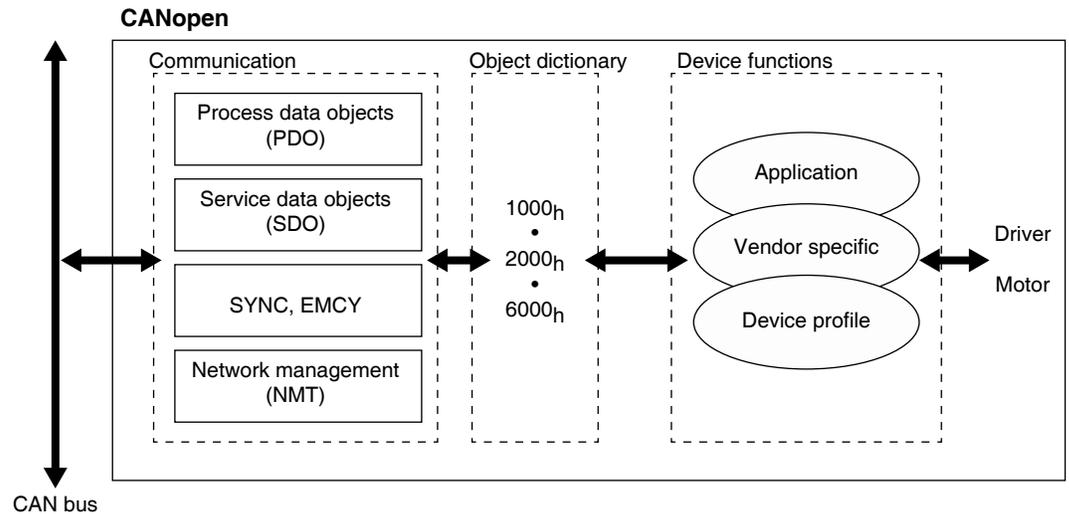
Figure 1.2: Device model with object dictionary

The object dictionary contains objects for describing the data types and executing the communication tasks and device functions under CANopen.

Object index   Each object is addressed by means of a 16 bit index, which is represented as a four-digit hexadecimal number. The objects are arranged in groups in the object dictionary. The following table shows an overview of the object dictionary supported by Lexium MDrive products as per the CANopen defini-tion.

Index range (hex)	Object group
1000h - 1FFFh	Communications profile
2000h - 5FFFh	Vendor specific objects
6000h - 9FFFh	Standardized device profiles

For a list of all CANopen objects see chapter 7 "Object dictionary".

1.2.4   CANopen profiles

Standardized profiles   Standardized profiles describe objects that are used with different devices without additional configuration. The users and manufacturers organization CAN in Automation has standardized various profiles. These include:

- DS301 communication profile
- DSP402 device profile
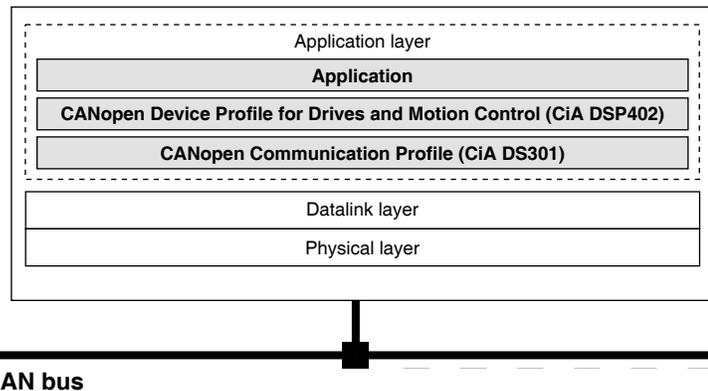
Figure 1.3: CANopen reference model

DS301 communication profile   The DS301 communication profile is the interface between device profiles and CAN bus. It was specified in 1995 under the name DS301 and defines uniform standards for common data exchange between different device types under CANopen.

The objects of the communication profile in the device carry out the tasks of data exchange and parameter exchange with other network devices and initialize, control and monitor the device in the network.

DSP402 device profile   The DSP402 device profile describes standardized objects for positioning, monitoring and settings of drives. The tasks of the objects include:

- Device monitoring and status monitoring (Device Control)
- Standardized parameterization
- Changing, monitoring and execution of operating modes

Vendor-specific profiles   The basic functions of a device can be used with objects of standardized device profiles. Only vendor-specific device profiles offer the full range of functions. The objects with which the special functions of a device can be used under CANopen are defined in these vendor-specific device profiles.

2   Before you begin - safety information

2

The information provided in this manual supplements the product manual. Carefully read the product manual before using the product.

2.1   Qualification of personnel

Only appropriately trained persons who are familiar with and understand the contents of this manual and all other pertinent product documentation are authorized to work on and with this product. In addition, these persons must have received safety training to recognize and avoid hazards involved. These persons must have sufficient technical training, knowledge and experience and be able to foresee and detect potential hazards that may be caused by using the product, by changing the settings and by the mechanical, electrical and electronic equipment of the entire system in which the product is used.

All persons working on and with the product must be fully familiar with all applicable standards, directives, and accident prevention regulations when performing such work.

2.2   Intended use

This product is a motor with an integrated drive and intended for industrial use according to this manual.

The product may only be used in compliance with all applicable safety regulations and directives, the specified requirements and the technical data.

Prior to using the product, you must perform a risk assessment in view of the planned application. Based on the results, the appropriate safety measures must be implemented.

Since the product is used as a component in an entire system, you must ensure the safety of persons by means of the design of this entire system (for example, machine design).

Operate the product only with the specified cables and accessories. Use only genuine accessories and spare parts. The product must NEVER be operated in explosive atmospheres (hazardous locations, Ex areas).

Any use other than the use explicitly permitted is prohibited and can result in hazards.

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel.

2.3   Hazard categories

Safety instructions to the user are highlighted by safety alert symbols in the manual. In addition, labels with symbols and/or instructions are attached to the product that alert you to potential hazards.

Depending on the seriousness of the hazard, the safety instructions are divided into 4 hazard categories.

⚠ DANGER
DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death or serious injury.

⚠ WARNING
WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

⚠ CAUTION
CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

CAUTION
CAUTION used without the safety alert symbol, is used to address practices not related to personal injury (e.g. **can result** in equipment damage).

2.4   Basic information

⚠ DANGER

UNINTENDED CONSEQUENCES OF EQUIPMENT OPERATION

When the system is started, the drives are usually out of the operator's view and cannot be visually monitored.

• Only start the system if there are no persons in the hazardous area.

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

LOSS OF CONTROL

•   The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop, overtravel stop, power outage and restart.

•   Separate or redundant control paths must be provided for critical functions.

•   System control paths may include communication links. Consideration must be given to the implication of unanticipated transmission delays or failures of the link.

•   Observe all accident prevention regulations and local safety guidelines. 1)

•   Each implementation of the product must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death or serious injury.

---

1) For USA: Additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems".

2.5 Standards and terminology

Technical terms, terminology and the corresponding descriptions in this manual are intended to use the terms or definitions of the pertinent standards.

In the area of drive systems, this includes, but is not limited to, terms such as "safety function", "safe state", "fault", "fault reset", "failure", "error", "error message", "warning", "warning message", etc.

Among others, these standards include:

- IEC 61800 series: "Adjustable speed electrical power drive systems"
- IEC 61158 series: "Industrial communication networks - Fieldbus specifications"
- IEC 61784 series: "Industrial communication networks - Profiles"
- IEC 61508 series: "Functional safety of electrical/electronic/ programmable electronic safety-related systems"

3 Basics

3

3.1 Communication profile

CANopen manages communication between the network devices with object dictionaries and objects. A network device can use process data objects (PDO) and service data objects (SDO) to request the object data from the object dictionary of another device and, if permissible, write back modified values.

The following can be done by accessing the objects of the network devices

- Exchange parameter values
- Start motion functions of individual CAN bus devices
- Request status information

3.1.1 Object dictionary

Each CANopen device manages an object dictionary which contains the objects for communication.

Index, subindex   The objects are addressed in the object dictionary via a 16 bit index. One or more 8 bit subindex entries for each object specify individual data fields in the object. Index and subindex are shown in hexadecimal notation with a subscript "$_h$".

Example   The following table shows index and subindex entries using the example of the object Homing Speeds (6098_h) for specifying the fast and slow speeds for homing functions.

Index	Subindex	Name	Meaning
6098_h	00_h	—	Number of data fields
	01_h	Homing speed fast	High speed during homing
	02_h	Homing speed slow	Low speed during homing

Table 3.1: Example index and subindex entries

Object descriptions in the manual   For CAN programming of a device, the objects of the following object groups are described in detail:

$1xxx_h$ objects: Communication objects.

$2xxx_h$ objects: Vendor-specific objects required to control the vendor specific functions of the device.

$6xxx_h$ objects: standardized objects of the device profile.

Standardized objects   Standardized objects allow you to use the same application program for different network devices of the same device type. This requires these objects to be contained in the object dictionary of the network devices. Standardized objects are defined in the DS301 communication profile and the DSP402 device profile.

3.1.2   Communication objects

Overview   The communication objects are standardized with the DS301 CANopen communication profile. The objects can be classified into 4 groups according to their tasks.

Process data objects

T_PDO1	R_PDO1
T_PDO2	R_PDO2
T_PDO3	R_PDO3
T_PDO4	R_PDO4

Special objects

SYNC

EMCY

Communication objects

Service data objects

T_SDO

R_SDO

Network management

NMT Services

NMT Node guarding

NMT Heartbeat

T_ = Transmit
R_ = Receive
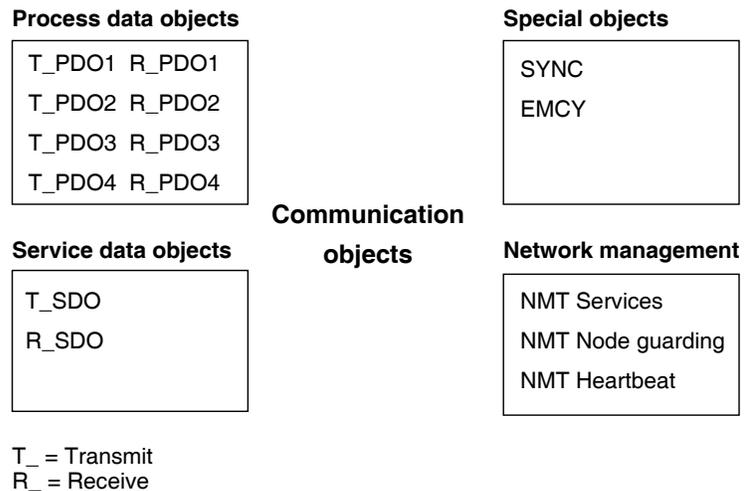
Figure 3.1: Communication objects

- PDOs (process data objects) for real-time transmission of process data
- SDOs (service data object) for read and write access to the object dictionary
- Objects for controlling CAN messages:
    1. SYNC object (synchronization object) for synchronization of networ devices
    2. EMCY object (emergency object), for signaling errors of a device or its peripherals.
- Network management services:
    1. NMT services for initialization and network control (NMT: network management)
    2. NMT Node Guarding for monitoring the network devices
    3. NMT Heartbeat for monitoring the network devices

CAN message   Data is exchanged via the CAN bus in the form of CAN messages. A CAN message transmits the communication object as well as numerous administration and control data.

CAN message

| 1 | 11 | 1 | . | 0 - 8 Bytes | 16 | 1 | 1 | 1 | .7 | ≥3 |

- Start bit
- Identifier
- RTR bit
- Control
- Data
- CRC
- Acknowledge
- End bits

11 COB ID

| 4 bits | 7 bits |

0 ... 8 Bytes data

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
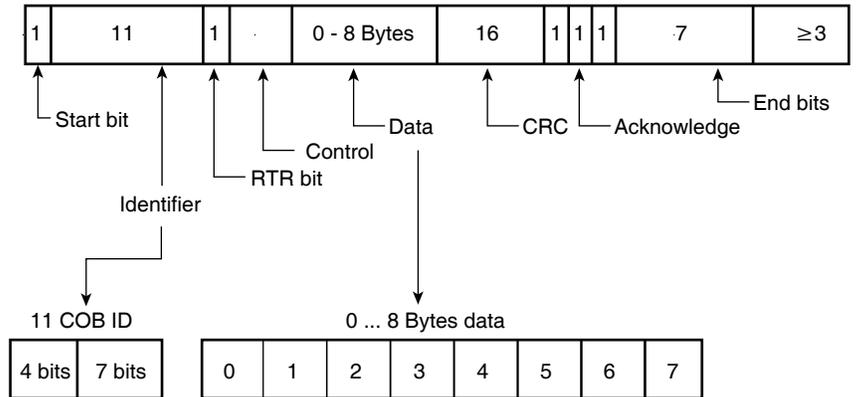
CANopen message (simplified)

Figure 3.2: CAN message and simplified CANopen message

CANopen message   For work with CANopen objects and for data exchange, the CAN message can be represented in simplified form because most of the bits are used for error correction. These bits are automatically removed from the receive message by the data link layer of the OSI model, and added to a message before it is transmitted.

The two bit fields "Identifier" and "Data" form the simplified CANopen message. The "Identifier" corresponds to the "COB ID" and the "Data" field to the data frame (maximum length 8 bytes) of a CANopen message.

COB ID   The COB ID (**C**ommunication **OB**ject Identifier) has 2 tasks as far as controlling communication objects is concerned:

- Bus arbitration: Specification of transmission priorities
- Identification of communication objects
  An 11 bit COB identifier as per the CAN 3.0A specification is defined for CAN communication; it comprises 2 parts
- Function code, 4 bits
- Node address (node ID), 7 bits.

COB ID

| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Function code
0 ... 15

Node ID
0 ... 127
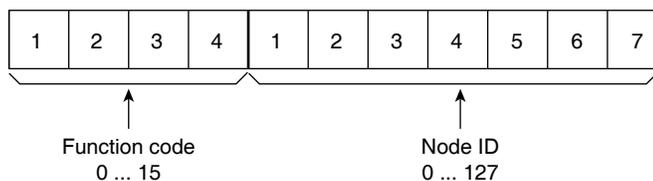
Figure 3.3: COB ID with function code and node address

COB IDs of the communication objects   The following table shows the COB IDs of all communication objects with the factory settings. The column «Index of object parameters» shows the index of special objects with which the settings of the communication objects can be read or modified via an SDO.

Communications object	Function code	Node address, node ID [1...127]	COB ID decimal (hex)	Index of object parameters
NMT Start/Stop Service	0 0 0 0	0 0 0 0 0 0 0	0 (0_h)	—
SYNC object	0 0 0 1	0 0 0 0 0 0 0	128 (80_h)	1005_h....1007_h
EMCY object	0 0 0 1	x x x x x x x	128 (80_h) + node ID	1014_h, 1015_h
T_PDO1	0 0 1 1	x x x x x x x	384 (180_h) + node ID	1800_h
R_PDO1	0 1 0 0	x x x x x x x	512 (200_h) + node ID	1400_h
T_PDO2	0 1 0 1	x x x x x x x	640 (280_h) + node ID	1801_h
R_PDO2	0 1 1 0	x x x x x x x	768 (300_h) + node ID	1401_h
T_PDO3	0 1 1 1	x x x x x x x	896 (380_h) + node ID	1802_h
R_PDO3	1 0 0 0	x x x x x x x	1024 (400_h) + node ID	1402_h
T_PDO4	1 0 0 1	x x x x x x x	1152 (480_h) + node ID	1803_h
R_PDO4	1 0 1 0	x x x x x x x	1280 (500_h) + node ID	1403_h
R_SDO	—	x x x x x x x	1408 (580_h) + node ID	—
R_SDO	—	x x x x x x x	1536 (600_h) + node ID	—
NMT error control	1 1 1 0	x x x x x x x	1792 (700_h) + node ID	—
LMT Services	1 1 1 1	1 1 0 0 1 0 x	2020 ($7E4_h$), 2021 ($7E5_h$)	—
NMT Identify Service	1 1 1 1	1 1 0 0 1 1 0	2022 ($7E6_h$)	—
NMT Services	1 1 1 1	1 1 0 1 0 0 x	2025 ($7E9_h$), 2026 ($7EA_h$)	—

Table 3.2: COB IDs of communication objects

Function code   The function code classifies the communication objects. Since the bits of the function code in the COB ID are more significant, the function code also controls the transmission priorities: Objects with a lower function code are transmitted with higher priority. For example, an object with function code "1" is transmitted prior to an object with function code "3" in the case of simultaneous bus access.

Node address   Each network device is configured before it can be operated on the network. The device is assigned a unique 7 bit node address (node ID) between 1 (01_h) and 127 ($7F_h$). The device address "0" is reserved for "broadcast transmissions" which are used to send messages to all reachable devices simultaneously.

Example   Selection of a COB ID

For a device with the node address 5, the COB ID of the communication object T_PDO1 is:

384+node ID = 384 (180h) + 5 = 389 (185h).

Data frame The data frame of the CANopen message can hold up to 8 bytes of data. In addition to the data frame for SDOs and PDOs, special frame types are specified in the CANopen profile:

- Error data frame
- Remote data frame for requesting a message

The data frames contain the respective communication objects.

3.1.3 Communication relationships

CANopen uses 3 relationships for communication between network devices:

- Master-slave relationship
- Client-server relationship
- Producer-consumer relationship

Master-slave relationship A network master controls the message traffic. A slave only responds when it is addressed by the master.

The master-slave relationship is used with network management objects for a controlled network start and to monitor the connection of devices.
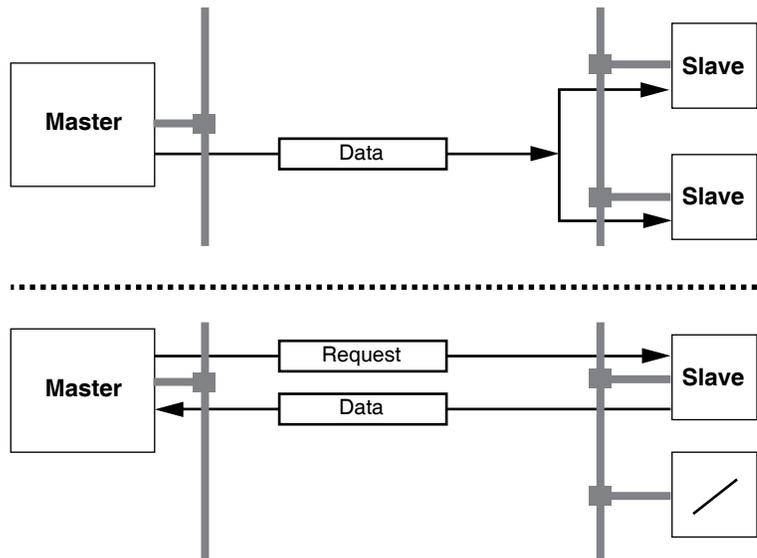
Figure 3.4 : Master – slave relationship

Messages can be interchanged with and without confirmation. If the master sends an unconfirmed CAN message, it can be received by a single slave or by all reachable slaves or by no slave.

To confirm the message, the master requests a message from a specific slave, which then responds with the desired data.

Client-server relationship

A client-server relationship is established between 2 devices. The «server» is the device whose object dictionary is used during data exchange. The «client» addresses and starts the exchange of messages and waits for a confirmation from the server.

A client-server relationship with SDOs is used to send configuration data and long messages.
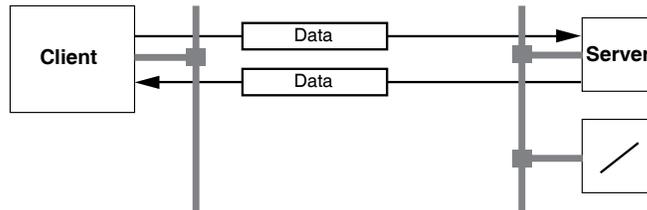
Figure 3.5: Client – server relationship

The client addresses and sends a CAN message to a server. The server evaluates the message and sends the response data as an acknowledgement.

Producer-consumer relationship

The producer-consumer relationship is used for exchanging messages with process data, because this relationship enables fast data exchange without administration data.

A "Producer" sends data, a "Consumer" receives data.
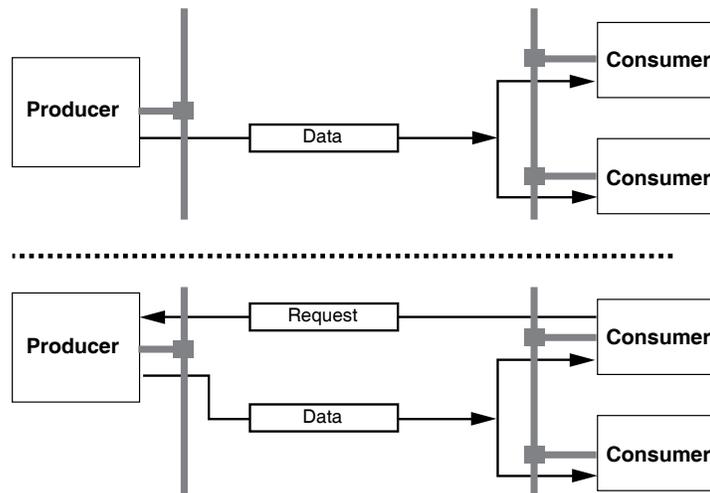
Figure 3.6: Producer – consumer relationship

The producer sends a message that can be received by one or more network devices. The producer does not receive an acknowledgement to the effect that the message was received. The message transmission can be triggered by

- An internal event, for example, "target position reached"
- The synchronization object SYNC
- A request of a consumer

See Section 3.3 "Process data communication" for details on the function of the producer-consumer relationship and on requesting messages.

3.2 Service data communication

3.2.1 Overview

Service Data Objects (SDO: Service Data Object) can be used to access the entries of an object dictionary via index and subindex. The values of the objects can be read and, if permissible, also be changed.

Every network device has at least one server SDO to be able to respond to read and write requests from a different device. A client SDO is only required to request SDO messages from the object dictionary of a different device or to change them in the dictionary.

The T_SDO of an SDO client is used to send the request for data exchange; the R_SDO is used to receive. The data frame of an SDO consist of 8 bytes.

SDOs have a higher COB ID than PDOs; therefore, they are transmitted over the CAN bus at a lower priority.

3.2.2 SDO data exchange

A service data object (SDO) transmits parameter data between 2 devices. The data exchange conforms to the client-server relationship. The server is the device to whose object dictionary an SDO message refers.
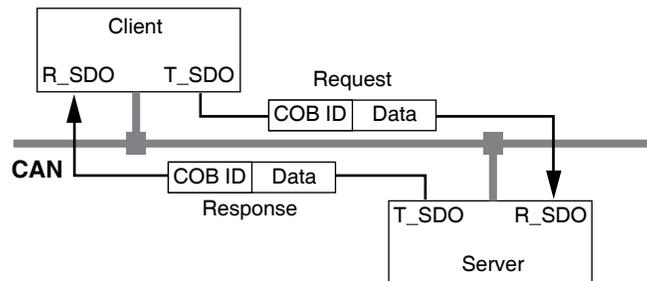
Figure 3.7: SDO message exchange with request and response

Message types Client-server communication is triggered by the client to send parameter values to the server or to get them from the server. In both cases, the client starts the communication with a request and receives a response from the server.

3.2.3   SDO message

Put simply, an SDO message consists of the COB ID and the SDO data frame, in which up to 4 bytes of data can be sent. Longer data sequences are distributed over multiple SDO messages with a special protocol.

The device transmits SDOs with a data length of up to 4 bytes. Greater amounts of data such as 8 byte values of the data type «Visible String 8» can be distributed over multiple SDOs and are transmitted successively in blocks of 7 bytes.

Example   The following illustration shows an example of an SDO message.
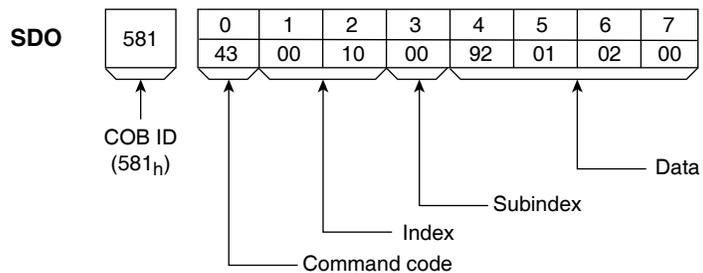
Figure 3.8: SDO message example

COB ID and data frame R_SDO and T_SDO have different COB IDs.

The data frame of an SDO messages consists of:

- Command code (ccd) which contains the SDO message type and the data length of the transmitted value
- Index and subindex which point to the object whose data is transported with the SDO message
- Data of up to 4 bytes

Evaluation of numeric values   Index and data are transmitted left-aligned in Intel, or little endian format. If the SDO contains numerical values of more than 1 byte in length, the data must be rearranged byte-by-byte before and after a transmission.
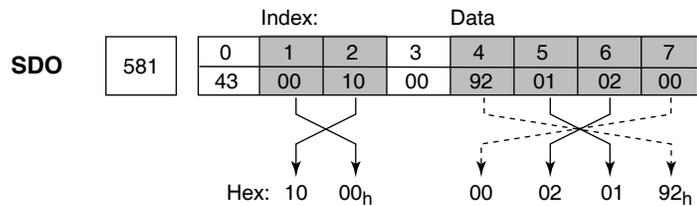
Figure 3.9: Rearranging numeric values greater than 1 byte

3.2.4 Reading and writing data

Writing data   The client starts a write request by sending index, subindex, data length and value. The server sends a confirmation indicating whether the data was correctly processed. The confirmation contains the same index andsubindex, but no data.
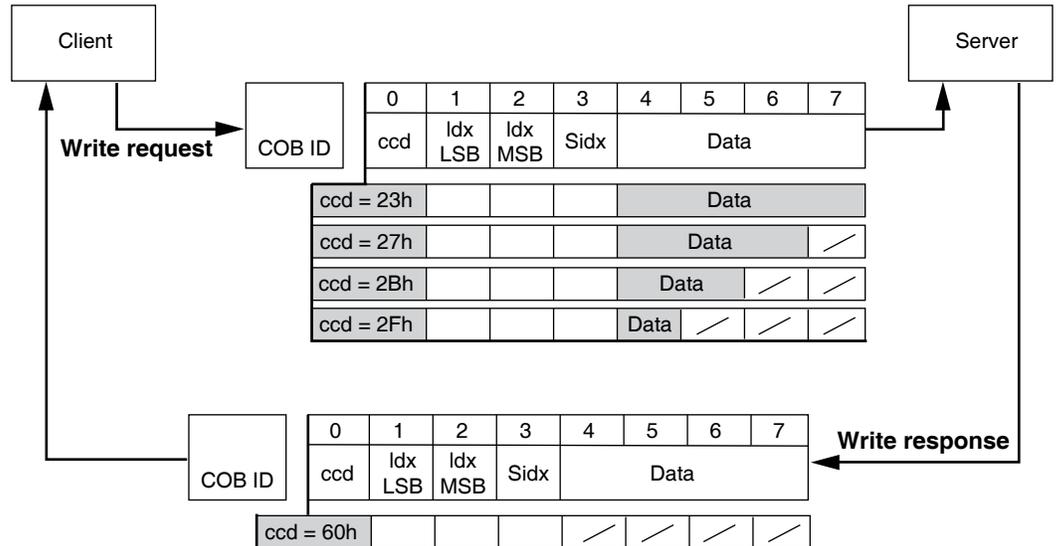
Figure 3.10: Writing parameter values

Unused bytes in the data field are shown with a slash in the graphic. The content of these data fields is not defined.

ccd coding   The table below shows the command code for writing parameter values. It depends on the message type and the transmitted data length.

Message type	Data length used				
	4 byte	**3 byte**	**2 byte**	**1 byte**	
Write request	23_h	27_h	$2B_h$	$2F_h$	Transmitting parameters
Write response	60_h	60_h	60_h	60_h	Confirmation
Error response	80_h	80_h	80_h	80_h	Error

Table 3.3: Command code for writing parameter values

Reading data   The client starts a read request by transmitting the index and subindex that point to the object or part of the object whose value it wants to read.

The server confirms the request by sending the desired data. The SDO response contains the same index and subindex. The length of the response data is specified in the command code «ccd».
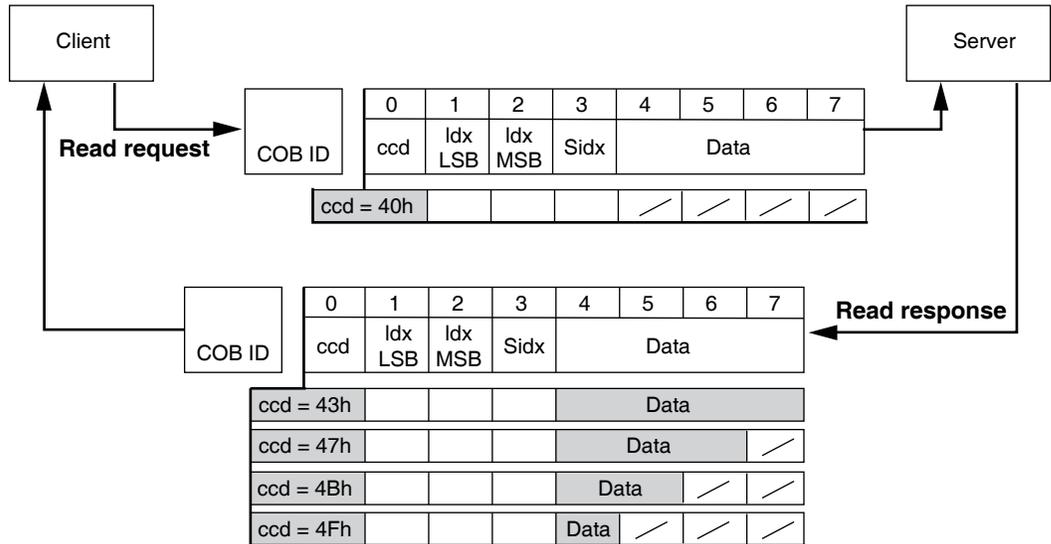
Figure 3.11: Reading a parameter value

The table below shows the command code transmitting a read value. It depends on the message type and the transmitted data length.

Message type	Data length used				
	4 byte	**3 byte**	**2 byte**	**1 byte**	
Read request	43_h	47_h	$4B_h$	$4F_h$	Request read value
Read response	40_h	40_h	40_h	40_h	Return read value
Error response	80_h	80_h	80_h	80_h	Error

Table 3.4: Command code for transmitting a read value

Error response   If a message could not be evaluated, the server sends an error message. See section 7.4.3 "SDO error message ABORT" for details on the evaluation of the error message.
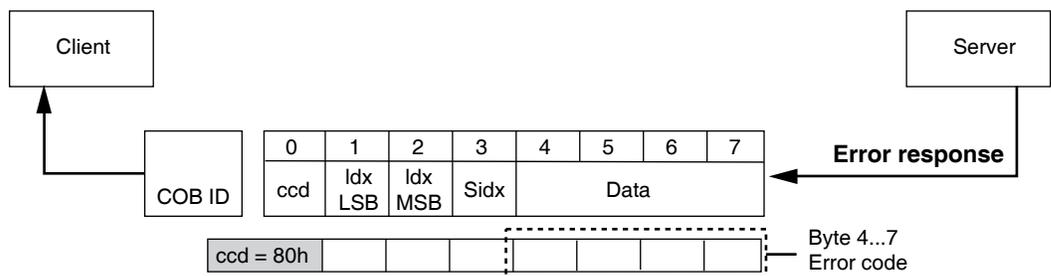
Figure 3.12: Response with error message (error response)

3.2.5 Reading data longer than 4 bytes

If values of more than 4 bytes are to be transmitted with an SDO message, the message must be divided into several frames. Each frame consists of 2 parts.

• Request by the SDO client,

• Confirmation by the SDO server.

The request by the SDO client contains the command code «ccd» with the toggle bit and a data segment. The confirmation frame also contains a toggle bit in the «ccd» segment. In the first frame, the toggle bit has the value «0», in the subsequent frames it toggles between 1 and 0.

Reading data The client starts a read request by transmitting the index and subindex that point to the object or the object value whose value it wants to read.

The server confirms the request by transmitting index, subindex, data length and the first 4 bytes of the requested data. The command code specifies that data of more than 4 bytes are transmitted. The command code of the read response from the server to the first message is 41h.
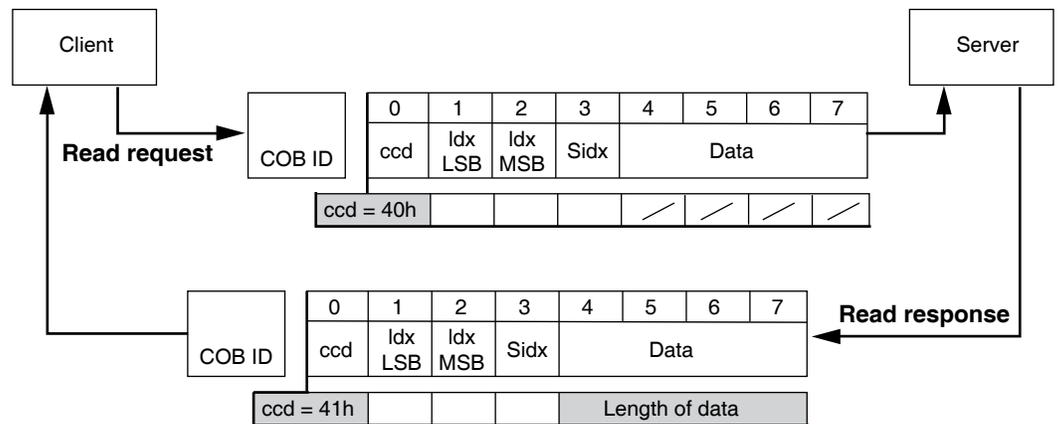
Figure 3.13: Transmitting the first message

In the next frames, the remaining data is requested and transmitted in packets of 7 bytes from the server.

ccd coding   The table below shows the command code for transmitting a read value. It depends on the message type, the value of the toggle bit, the transmitted data length and the value of the bit that indicates the end of the entire SDO message.

Message type	Data length used							
	7 byte	**6 byte**	**5 byte**	**4 byte**	**3 byte**	**2 byte**	**1 byte**	
Read request Toggle Bit = 0		60_h	60_h		60_h	60_h	60_h	Confirmation with Toggle Bit = 0
Read request Toggle Bit = 1	70_h	70_h	70_h	70_h	70_h	70_h	70_h	Confirmation with Toggle Bit = 1
Read response Toggle Bit = 0	00_h	—	—	—	—	—	—	Send parameter with Toggle Bit = 0
Read response Toggle Bit = 1	10_h	—	—	—	—	—	—	Send parameter with Toggle Bit = 1
Read response last message Toggle Bit = 0	01_h	03_h	05_h	07_h	09_h	$0B_h$	$0D_h$	Transmit parameter with last message and Toggle Bit = 0
Read response last message Toggle Bit = 1	11_h	13_h	15_h	17_h	19_h	$1B_h$	$1D_h$	Transmit parameter with last message and Toggle Bit = 1
Error response	80_h	80_h	80_h	80_h	80_h	80_h	80_h	Error

Table 3.5: Command code data lengths > 4 bytes

Refer to the DS301 of the CiA for additional information on this procedure.

3.3   Process data communication

3.3.1   Overview

Process data objects (PDO: Process Data Object) are used for realtime data exchange of process data such as actual and reference values or the operating state of the device. Transmission is very fast because the data is sent without additional administration data and data transmission acknowledgement from the recipient is not required.

The flexible data length of a PDO message also increases the data throughput. A PDO message can transmit up to 8 bytes of data. If only 2 bytes are assigned, only 2 data bytes are sent. The length of a PDO message and the assignment of the data fields are specified by PDO mapping. See section 3.3.4 «PDO mapping» for additional information.

PDO messages can be exchanged between devices that generate or process process data.
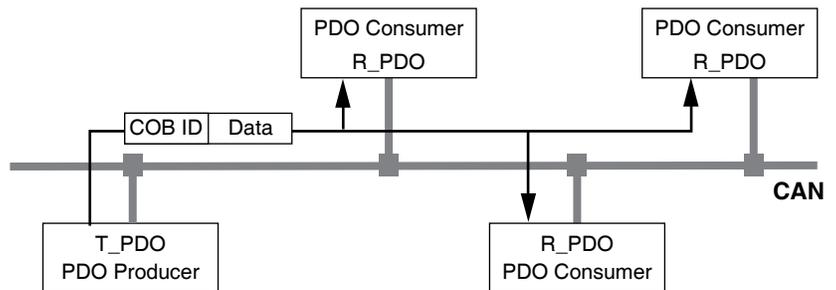
Figure 3:14: PDO data exchange

Data exchange with PDOs follows to the producer-consumer relationship and can be triggered in 3 ways:

*   Synchronized
*   Event-driven, asynchronous

The SYNC object controls synchronized data processing. Synchronous PDO messages are transmitted immediately like the standard PDO messages, but are only evaluated on the next SYNC. For example, several drives can be started simultaneously via synchronized data exchange.

The device immediately evaluates PDO messages that are called on request or in an event-driven way.

The transmission type can be specified separately for each PDO with subindex 02_h (transmission type) of the PDO communication parameter. The objects are listed in Table 3.5.

3.3.3 PDO message

T_PDO, R_PDO One PDO each is available for sending and receiving a PDO message:

- T_PDO to transmit the PDO message (T: Transmit),
- R_PDO to receive PDO messages (R: Receive).

The following settings for PDOs correspond to the defaults read and set via objects of the communication profile.

The device uses 6 PDOs, 3 receive PDOs and 3 transmit PDOs. By default, the PDOs are evaluated or transmitted in an event-driven way.

PDO settings The PDO settings can be read and changed with 8 communication objects:

Object	Meaning
1st receive PDO parameter (1400_h)	Settings for R_PDO1
2nd receive PDO parameter (1401_h)	Settings for R_PDO2
3rd receive PDO parameter (1402_h)	Settings for R_PDO3
4th receive PDO parameter (1403_h)	Settings for R_PDO4
1st transmit PDO parameter (1800_h)	Settings for T_PDO1
2nd transmit PDO parameter (1801_h)	Settings for T_PDO2
3rd transmit PDO parameter (1802_h)	Settings for T_PDO3
4th transmit PDO parameter (1803_h)	Settings for T_PDO4

Table 3.6: Communication objects for PDO

Activating PDOs With the default PDO settings, R_PDO1 and T_PDO1 are activated. The other PDOs must be activated first. A PDO is activated with bit 31 (valid bit) in subindex 01h of the respective communication object:

Subindex 01h   Objects 140xh, 180xh (x: 0, 1, 2)

31 (MSb)

valid bit
0: PDO activated
1: PDO not activated
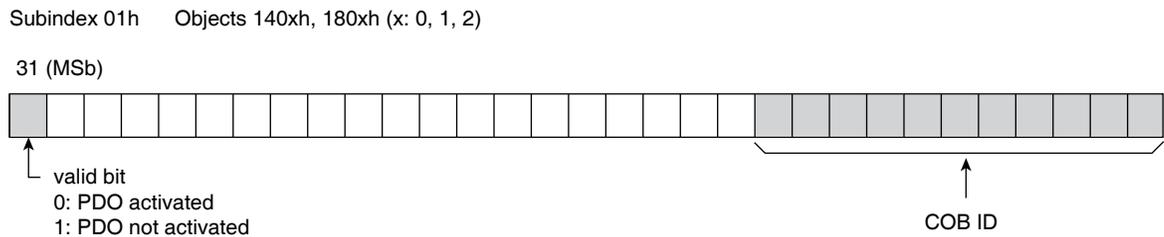
COB ID

Figure 3.15: Activating PDOs via subindex 01_h, bit 31

Example **Setting for R_PDO3 in object 1402_h**

- Subindex 01_h = 8000 04xx$_h$: R_PDO3 not activated
- Subindex 01_h = 0000 04xx$_h$: R_PDO3 activated.

Values for "x" in the example depend on the COB ID setting.

PDO time intervals   The time intervals «inhibit time» and «event timer» can be set for each transmit PDO.

- The time interval «inhibit time» can be used to reduce the CAN bus load, which can be the result of continuous transmission of T_PDOs. If an inhibit time not equal to zero is entered, a transmitted PDO will only be re-transmitted after the inhibit time has elapsed. The time is set with subindex 03h.

- The time interval «event timer» cyclically triggers an event message. After the time intervals has elapsed, the device transmits the event controlled T_PDO. The time is set with subindex 05h.

Receive PDOs   The objects for R_PDO1, R_PDO2, R_PDO3 and R_PDO4 are preset. They may be re-mapped to suit the end-user application.
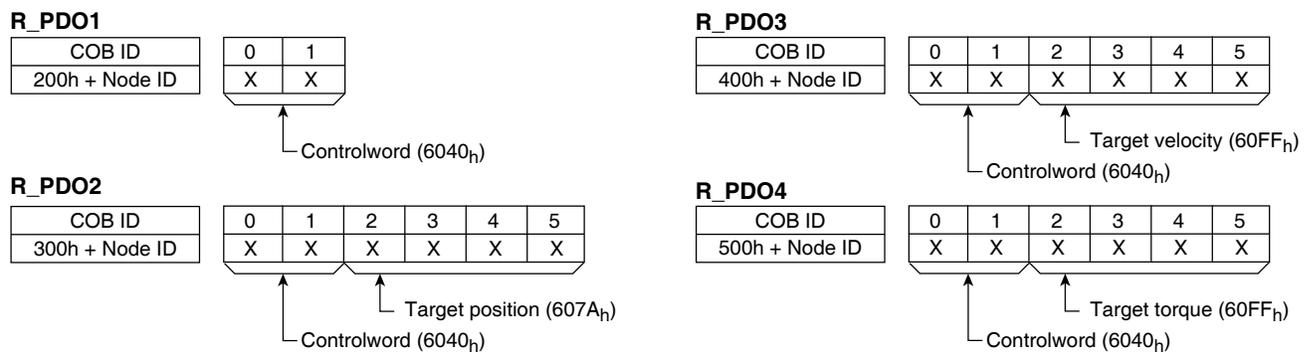
Figure 3.16: Receive PDOs

R_PDO1   R_PDO1 contains the control word, object `controlword (6040`$_h$`)`, of the state machine which can be used to set the operating state of the device.

R_PDO1 is evaluated asynchronously, i.e. it is event-driven. R_PDO1 is preset.

R_PDO2   With R_PDO2, the control word and the target position of a motion command, object `target position (607A`$_h$`)`, are received for a movement in the operating mode "Profile Position".

R_PDO2 is evaluated asynchronously, i.e. it is event-driven. R_PDO2 is preset.

For details on the SYNC object see section 3.4 "Synchronization".

R_PDO3   R_PDO3 contains the control word and the target velocity, object `Target velocity (60FF`$_h$`)`, for the operating mode "Profile Velocity".

R_PDO3 is evaluated asynchronously, i.e. it is event-driven. R_PDO3 is preset.

R_PDO4   R_PDO4 contains the control word and the target torque, object `Target torque (6071`$_h$`)`, for the operating mode "Profile Torque" (closed loop LMD products only).

R_PDO4 is evaluated asynchronously, i.e. it is event-driven. R_PDO4 is preset.

Transmit PDOs   The objects for T_PDO1, T_PDO2 and T_PDO3 can be changed by means of PDO mapping.

T_PDO1

COB ID		0	1
180h + Node ID		X	X

└─ Statusword (6041_h)

T_PDO2

COB ID		0	1	2	3	4	5
280h + Node ID		X	X	X	X	X	X

└─ Position actual value (6064_h)

└─ Statusword (6041_h)

T_PDO3

COB ID		0	1	2	3	4	5
380h + Node ID		X	X	X	X	X	X

└─ Velocity actual value ($606C_h$)

└─ Statusword (6041_h)

T_PDO4

COB ID		0	1
480h + Node ID		X	X
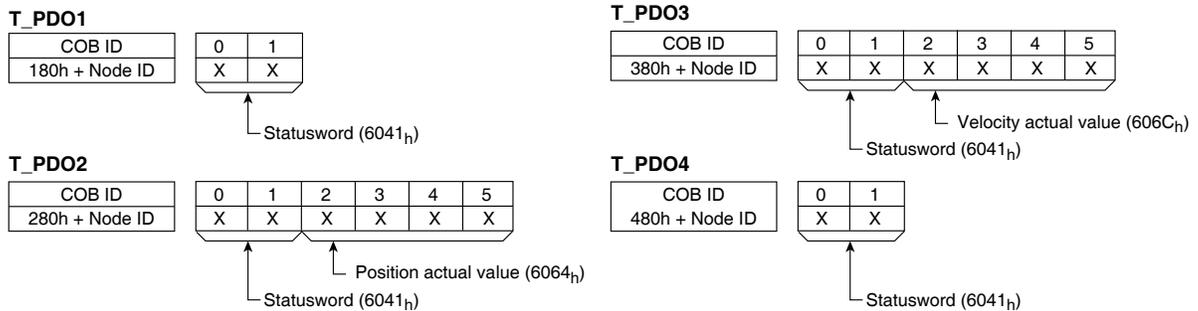
└─ Statusword (6041_h)

Figure 3.17: Transmit PDOs

T_PDO1   T_PDO1 contains the status word, object statusword (6041_h), of the state machine.

T_PDO1 is transmitted asynchronously and in an event-driven way whenever the status information changes.

T_PDO2   T_PDO2 contains the status word and the actual position of the motor, object Position actual value (6064_h), to monitor movements in the operating mode "Profile Position".

T_PDO2 is transmitted after receipt of a SYNC object and in an eventdriven way.

T_PDO3   T_PDO3 contains the status word and the actual velocity, object Velocity actual value ($606C_h$), for monitoring the velocity profile in the operating mode "Profile Velocity".

T_PDO3 is transmitted asynchronously and in an event-driven way whenever the status information changes.

T_PDO4   T_PDO4 contains the status word, object statusword (6041_h), of the state machine.

T_PDO4 is transmitted asynchronously and in an event-driven way whenever the status information changes.

V1.00, 09.2014

3.3.4 PDO mapping

Up to 8 bytes of data from different areas of the object dictionary can be transmitted with a PDO message. Mapping of data to a PDO message is referred to as PDO mapping.

The diagram below shows the data exchange between PDOs and object dictionary on the basis of two examples of objects in T_PDO3 and R_PDO3 of the PDOs.
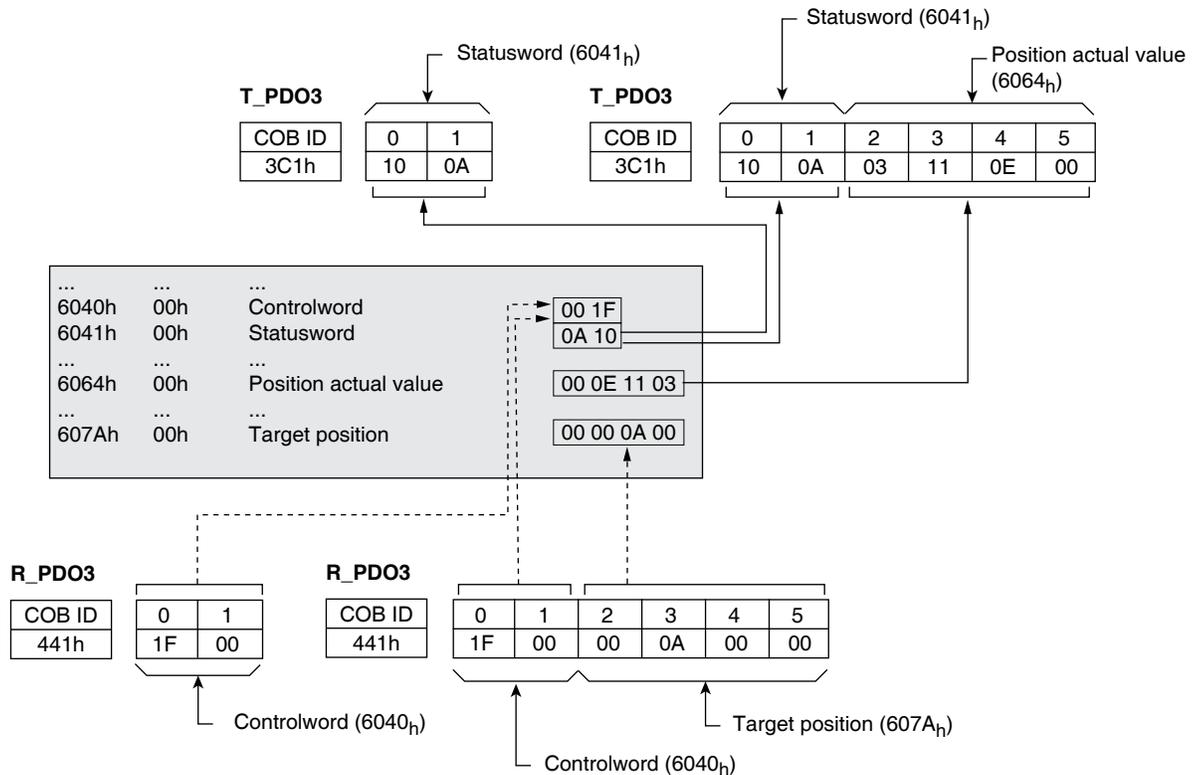
Figure 3.18:PDO mapping

Dynamic PDO mapping

The device uses dynamic PDO mapping. Dynamic PDO mapping means that objects can be mapped to the corresponding PDO using adjustable settings.

The settings for PDO mapping are defined in an assigned communication object for each PDO.

Object	PDO mapping for	Type
1st receive PDO mapping (1600_h)	R_PDO1	Dynamic
2nd receive PDO mapping (1601_h)	R_PDO2	Dynamic
3rd receive PDO mapping (1602_h)	R_PDO3	Dynamic
4threceive PDO mapping (1603_h)	R_PDO4	Dynamic
1st transmit PDO mapping ($1A00_h$)	T_PDO1	Dynamic
2nd transmit PDO mapping ($1A01_h$)	T_PDO2	Dynamic
3rd transmit PDO mapping ($1A02_h$)	T_PDO3	Dynamic
4th transmit PDO mapping ($1A03_h$)	T_PDO4	Dynamic

Table 3.7: Dynamic PDO mapping parameters

Structure of the entries   Up to 8 bytes of 8 different objects can be mapped in a PDO. Each communication object for setting the PDO mapping provides 4 subindex entries. A subindex entry contains 3 pieces of information on the object: the index, the subindex and the number of bits that the object uses in the PDO.

PDO mapping object

00h	2		
01h	6041h	00h	10h
02h	606Ch	00h	20h
. . .	. . .	. . .	. . .

LSB

31	15	7	0 Bit
XX XXh	XXh	XXh	

Index   Subindex   Object length
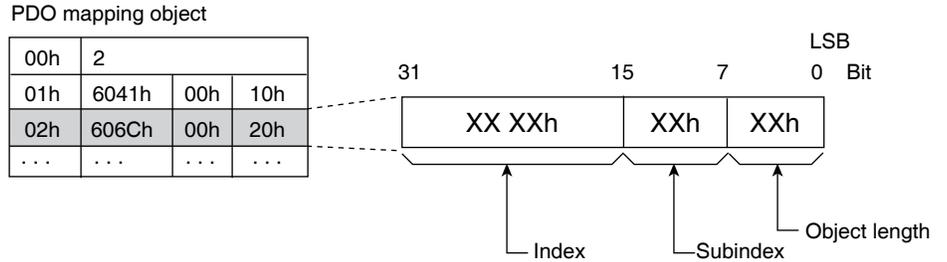
Figure 3.19: Structure of entries for PDO mapping

Subindex 00h of the communication object contains the number of valid subindex entries.

Object length	**Bit value**
08_h	8 bits
10_h	16 bits
20_h	30 bits

Table 3.8: Subindex object length entries

PDO mapping objects

Index	Subindex	Object	PDO	Data type	Parameter name
2009$_h$	1	Read value of output fault	T_PDO	UINT8	Out_Fault
2010$_h$	1	Read value of analog input	T_PDO	UINT16	Analog_In_Reading
2014$_h$	1	Read Aux voltage level	T_PDO	UINT16	AuxPower_Voltage
2015$_h$		Read +V level	T_PDO	UINT16	Vin_Voltage
2018$_h$	1	Read internal temperature	T_PDO	INT8	Temperature_Reading
2019$_h$	1	Read bridge temperature	T_PDO	UINT16	TempC_Bridge_Reading
2033$_h$	4	Read captured position	T_PDO	INT32	CaptureInPositn_user
2741$_h$	0	Read hybrid status byte	T_PDO	UINT8	HybridStatusByte

Index	Subindex	Object	PDO	Data type	Parameter name
603F$_h$	0	Read error code	T_PDO	UINT16	ErrorCode
6040$_h$	0	Controlword	R_PDO	UINT16	Controlword
6041$_h$	0	Statusword	T_PDO	UINT16	Statusword
6060$_h$	0	Modes of operation	R_PDO	INT8	Modes_of_operation
6061$_h$	0	Read mode of operation	T_PDO	INT8	Modes_of_operation_display
6062$_h$	0	Read position value	T_PDO	INT32	Position_demand_value_user
6063$_h$	0	Position actual value	T_PDO	INT32	Position_actual_value_inc
6064$_h$	0	Position actual value	T_PDO	INT32	Position_actual_value
606C$_h$	0	Velocity actual value	T_PDO	INT32	Velocity_actual_value
6071$_h$	0	Target torque	R_PDO	INT16	Target_torque
607A$_h$	0	Target position	R_PDO	INT32	Target_position
607E$_h$	0	Polarity	R_PDO	UINT8	Polarity
6081$_h$	0	Profile velocity	R_PDO	UINT32	Profile_velocity
6082$_h$	0	End velocity	R_PDO	UINT32	End_velocity
6083h	0	Profile acceleration	R_PDO	UINT32	Profile_acceleration
6084h	0	Profile deceleration	R_PDO	UINT32	Profile_deceleration
6086$_h$	0	Motion profile type	R_PDO	INT16	Motion_profile_type
60FD$_h$	0	Digital inputs	T_PDO	UINT32	Digital_inputs
60FE$_h$	1	Digital outputs	R_PDO	UINT32	Digital_outputs
60FF$_h$	0	Target velocity	R_PDO	INT32	Target_velocity

Table 3.9: Supported PDO mapping objects

3.4   Synchronization

The synchronization object SYNC controls the synchronous exchange of messages between network devices for purposes such as the simultaneous start of multiple drives.

The data exchange conforms to the producer-consumer relationship. The SYNC object is transmitted to all reachable devices by a network device and can be evaluated by the devices that support synchronous PDOs.
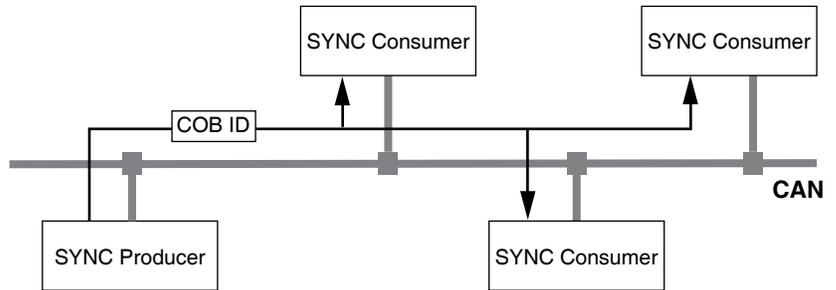
Figure 3.20: SYNC message

Time values for synchronization

Two time values define the behavior of synchronous data transmission:

* The cycle time specifies the time intervals between 2 SYNC messages. It is set with the object `Communication cycle period(1006h).`

* The synchronous time window specifies the time span during which the synchronous PDO messages must be received and transmitted. The time window is defined with the object `Synchronous window length (1007h).`
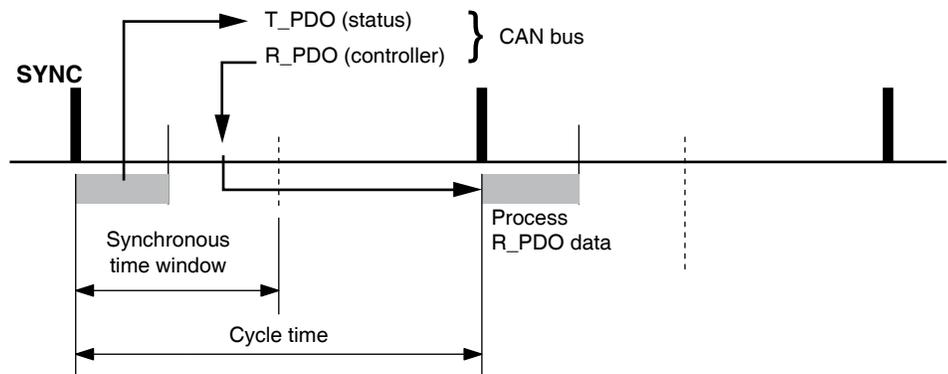
Figure 3.21: Synchronization times

Synchronous data transmission

From the perspective of a SYNC recipient, in one time window the status data is transmitted first in a T_PDO, then new control data is received via an R_PDO. However, the control data is only processed when the next SYNC message is received. The SYNC object itself does not transmit data.

Cyclic and acyclic data transmission   Synchronous exchange of messages can be cyclic or acyclic.
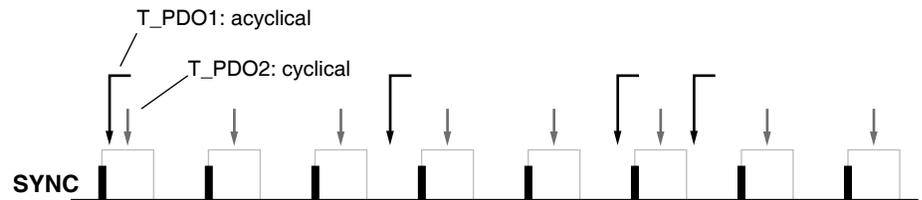
Figure 3.22: Cyclic and acyclic transmission

In the case of cyclic transmission, PDO messages are exchanged continuously in a specified cycle, for example with each SYNC message.

If a synchronous PDO message is transmitted acyclically, it can be transmitted or received at any time; however, it will not be valid until the next SYNC message.

Cyclic or acyclic behavior of a PDO is specified in the subindex transmission type (02h) of the corresponding PDO parameter, for example, in the object 1st receive PDO parameter (1400h:02h) for R_PDO1.

COB ID, SYNC object   For fast transmission, the SYNC object is transmitted unconfirmed and with high priority.

The COB ID of the SYNC object is set to the value 128 (80h) by default. The value can be changed after initialization of the network with the object COB-ID SYNC Message (1005h) .

"Start" PDO   With the default settings of the PDOs, R_PDO1 ... R_PDO4 and T_PDO1 ... T_PDO4 are received and transmitted asynchronously.
T_PDO2 ... T_PDO3 are transmitted additionally after the event timer has elapsed. The synchronization allows an operating mode to be started simultaneously on multiple devices so that, for example, the feed of a portal drive with several motors can be synchronized.

3.5   Emergency service

The Emergency Service signals internal device errors via the CAN bus. The error message is transmitted to the network devices with an EMCY object according to the Consumer-Producer relationship.
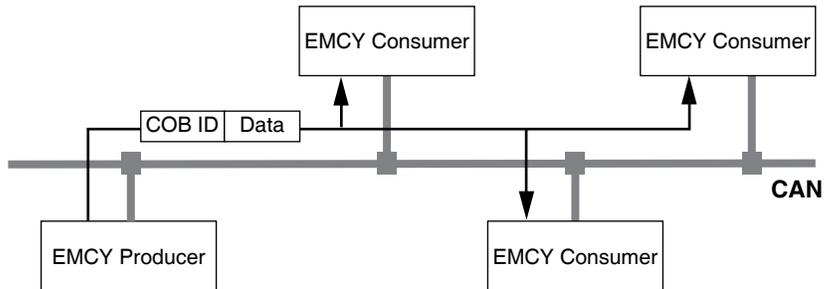
Figure 3.23: Error message via EMCY objects

Boot-up message   The communication profile DS301, version 3.0, defines an additional task for the EMCY object: sending a boot-up message. A boot-up message informs the network devices that the device that transmitted the message is ready for operation in the CAN network.

The boot-up message is transmitted with the COB ID 700h + node ID and one data byte (00h).

3.5.1 Error evaluation and handling   EMCY message If an internal device error occurs, the device switches to the operating state 9 Fault as per the CANopen state machine. At the same time, it transmits an EMCY message with error register and error code.
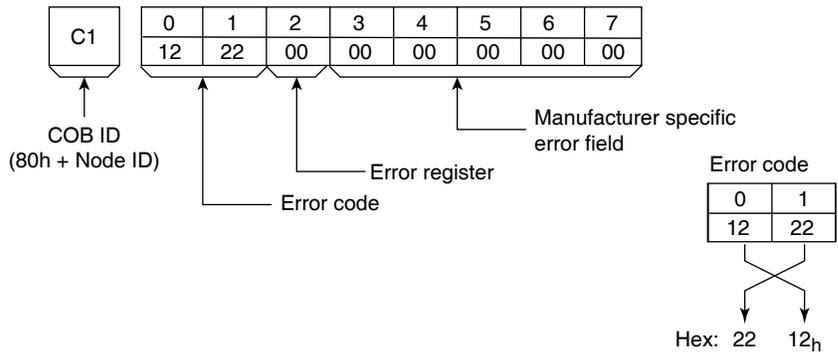
Figure 3.24: EMCY message

Bytes 0, 1 - Error code, value is also saved in the object `Error code` (603Fh)

Byte 2 - Error register, value is also saved in the object `Error register` (1001h)

Bytes 3, 4 - Reserved

Byte 5 - PDO: Number of the PDO

Bytes 6, 7 - Vendor-specific error code

COB ID   The COB ID for each device on the network supporting an EMCY object is determined on the basis of the node address:

COB ID = Function code EMCY object (80h) + node ID

The function code of the COB ID can be changed with the object `COBID emergency(1014h).`

Error register and error code   The error register contains bit-coded information on the error. Bit 0 remains set as long as an error is active. The remaining bits identify the error type. The exact cause of error can be determined on the basis of the error code. The error code is transmitted in Intel format as a 2 byte value; the bytes must be reversed for evaluation.

See chapter 7 «Diagnostics and troubleshooting» for a list of the error messages and error responses by the device as well as remedies.

Error memory   The device saves the error register in the object `Error register (1001h)` and the last error that occurred in the object `Error code (603Fh).`

3.6   Network management services

Network management (NMT) is part of the CANopen communication profile; it is used to initialize the network and the network devices and to start, stop and monitor the network devices during operation on the network.

NMT services are executed in a master-slave relationship. The NMT master addresses individual NMT slaves via their node address. A message with node address «0» is broadcast to all reachable NMT slaves simultaneously.
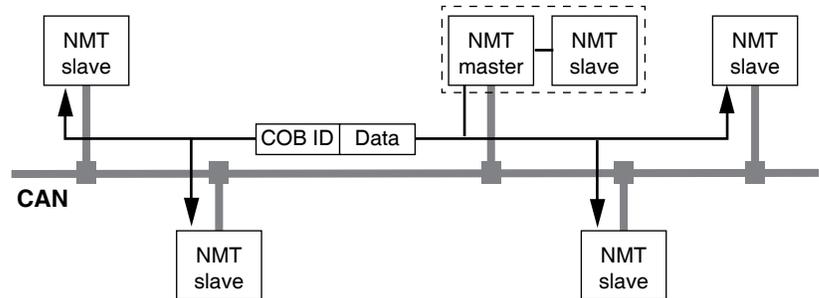
Figure 3.25: NMT services via the master-slave relationship

Lexium MDrive and Lexium MDrive devices can only take on the function of an NMT slave.

NMT services NMT services can be divided into 2 groups:

• Services for device control, to initialize devices for CANopen communication and to control the behavior of devices during operation on the network

• Services:for connection monitoring

3.6.1 NMT services for device control

NMT state machine   The NMT state machine describes the initialization and states of an NMT slave during operation on the network.
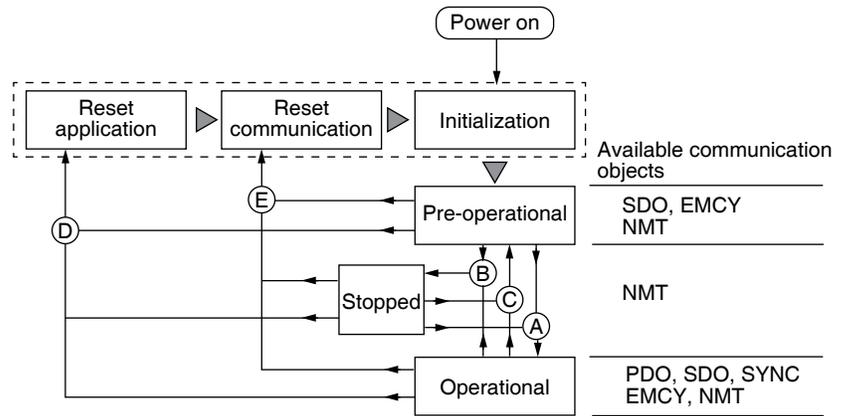
Figure 3.26: NMT state machine and available communication objects

To the right, the graphic shows the communication objects that can be used in the specific network state.

Initialization

An NMT slave automatically runs through an initialization phase after the supply voltage is switched on (power on) to prepare it for CAN bus operation. On completion of the initialization, the slave switches to the operating state "Pre Operational" and sends a boot-up message. From now on, an NMT master can control the operational behavior of an NMT slave on the network via 5 NMT services, represented in Figure 3.26 by the letters A to E.

NMT service	Transition	Meaning
Start remote node (Start network node)	A	Transition to operating state "Operational" Start normal operation on the network
Stop remote node (Stop network node)	B	Transition to operating state "Stopped" Stops communication of the network device on the network. If connection monitoring is active, it remains on. If the power stage is enabled (operating state "Operation Enabled" or "Quick Stop"), an error of error class 2 is triggered. The motor is stopped and the power stage disabled.
Enter Pre-Operational (Transition to "Pre-Operational")	C	Transition to operating state "Pre-Operational" The communication objects except for PDOs can be used. The operating state "Pre-Operational" can be used for configuration via SDOs: - PDO mapping - Start of synchronization - Start of connection monitoring
Reset node (Reset node)	D	Transition to operating state "Reset application" Load stored data of the device profiles and automatically switch via operating state "Reset communication" to "Pre-Operational".
Reset communication (Reset communication data)	E	Transition to operating state "Reset communication" Load stored data of the communication profile and automatically transition to operating state "Pre-Operational". If the power stage is enabled (operating state "Operation Enabled" or "Quick Stop"), an error of error class 2 is triggered. The motor is stopped and the power stage disabled.

Table 3.10: NMT state machine transitions

Persistent data memory   When the supply voltage is switched on (power on), the device loads the saved object data from the non-volatile EEPROM for persistent data to the RAM.

NMT message   The NMT services for device control are transmitted as unconfirmed messages with the COB ID = 0 . By default, they have the highest priority on the CAN bus.

The data frame of the NMT device service consists of 2 bytes.
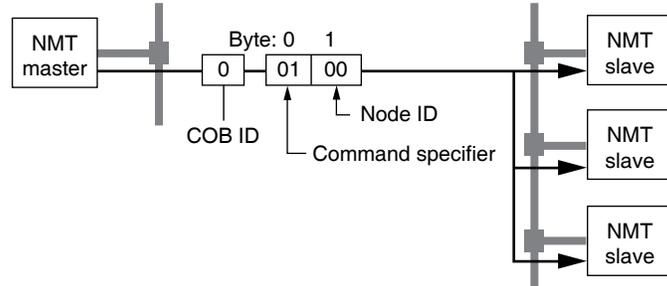
Figure 3.27: NMT message

The first byte, the "Command specifier", indicates the NMT service used

Command specifier	NMT service	Transition
1 (01_h)	Start remote node	A
2 (02_h)	Stop remote node	B
128 (80_h)	Enter pre-operational	C
129 (81_h)	Reset node	D
130 (82_h)	Reset communication	E

Table 3.10: NMT command specifiers.

The second byte addresses the recipient of an NMT message with a node address between 1 and 127 ($7F_h$). A message with node address "0" is broadcast to all reachable NMT slaves.

3.6.2   NMT services for connection monitoring

Connection monitoring monitors the communication status of network devices.

3 NMT services for connection monitoring are available:

- «Node guarding» for monitoring the connection of an NMT slave
- «Life guarding» for monitoring the connection of an NMT master
- «Heartbeat» for unconfirmed connection messages from network devices.

3.6.2.1   Node guarding / Life guarding

COB ID

The communication object `NMT error control (700h+node-Id)` is used for connection monitoring. The COB ID for each NMT slave is determined on the basis of the node address:

COB ID = function code `NMTerror control (700h) + node-Id.`

Structure of the NMT message

After a request from the NMT master, the NMT slave responds with one data byte.
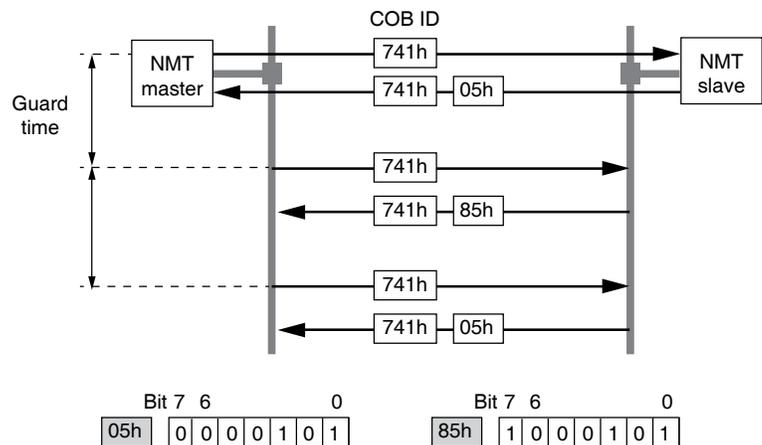
Figure 3.28: Acknowledgement of the NMT slave

Bits 0 to 6 identify the NMT state of the slave:

- 4 (04h): "Stopped"
- 5 (05h): "Operational"
- 127 (7Fh): "Pre-Operational"

After each "guard time" interval, bit 7 switches toggles between "0" and "1", so the NMT master can detect and ignore a second response within the "guard time" interval. The first request when connection monitoring is started begins with bit 7 = 0.

Connection monitoring must not be active during the initialization phase of a device. The status of bit 7 is reset as soon as the device runs though the NMT state "Reset communication".

Connection monitoring remains active in the NMT state "Stopped".

Configuration

Node Guarding/Life Guarding is configured via:

- Guard time (100Ch)
- Life time factor (100Dh)

Connection error   The NMT master signals a connection error to the master program if:

- the slave does not respond within the «guard time» period
- the NMT state of the slave has changed without a request by the NMT master.

Figure 3.29 shows an error message after the end of the third cycle because of a missing response from an NMT slave.
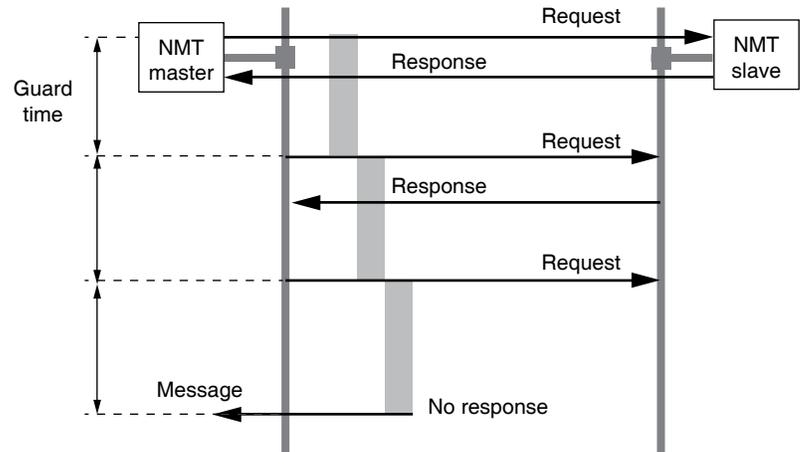
Figure 3.29: «Node Guarding» and «Life Guarding» with time intervals

3.6.2.2   Heartbeat

The optional Heartbeat protocol replaces the node guarding/life guarding protocol. It is recommended for new device versions.

A heartbeat producer transmits a heartbeat message cyclically at the frequency defined in the object `Producer heartbeat time (1017h)`. One or several consumers can receive this message. `Producer heartbeat time (1016h) = 0` deactivates heartbeat monitoring.

The relationship between producer and consumer can be configured with objects. If a consumer does not receive a signal within the period of time set with `Consumer heartbeat time (1016h)`, it generates an error message (heartbeat event). Consumer heartbeat time (1016h) = 0 deactivates monitoring by a consumer.
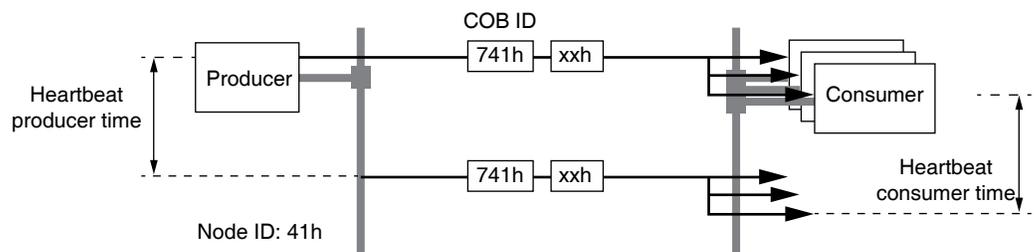
Figure 3.30: «Heartbeat» monitoring

Data byte for NMT state evaluation of the "Heartbeat" producer:

- 0 (00_h): "Boot-Up"
- 4 (04_h): "Stopped"
- 5 (05_h): "Operational"
- 127 ($7F_h$): "Pre-Operational"

Time intervals   The time intervals are set in increments of 1 ms steps; the values for the consumer must not be less than the values for the producer. Whenever the "Heartbeat" message is received, the time interval of the producer is restarted.

Start of monitoring   "Heartbeat" monitoring starts as soon as the time interval of the producer is greater than zero. If "Heartbeat" monitoring is already active during the NMT state transition to "Pre-Operational", "Heartbeat" monitoring starts with sending of the boot-up message. The boot-up message is a Heartbeat message with one data byte 00_h.

Devices can monitor each other via "Heartbeat" messages. They assume the function of consumer and producer at the same time.

Page intentionally left blank

4   Installation

4

⚠ **WARNING**
SIGNAL AND DEVICE INTERFERENCE
Signal interference can cause unexpected responses of device.
●   Install the wiring in accordance with the EMC requirements.   ●   Verify compliance with the EMC requirements.
Failure to follow these instructions can result in death, serious injury or equipment damage.

For information on installation of the device and connecting the device to the fieldbus see the product hardware manual.

Page intentionally left blank

5 Commissioning

5

⚠ WARNING
LOSS OF CONTROL
The product is unable to detect an interruption of the network link if
• Verify that connection monitoring is on.
• The shorter the time for monitoring, the faster the detection of the interruption.
Failure to follow these instructions can result in death, serious injury or equipment damage.

⚠ WARNING
UNINTENDED OPERATION
The product is unable to detect an interruption of the network link if
• Do not write values to reserved parameters.
• Do not write values to parameters unless you fully understand the function.
• Run initial tests without coupled loads.
• Verify that the system is free and ready for the movement before changing parameters.
• Verify the use of the word sequence with fieldbus communication.
• Do not establish a fieldbus connection unless you have fully understood the communication principles.
Failure to follow these instructions can result in death, serious injury or equipment damage.

5.1 Commissioning the device

For installation in the network, the device must first be properly installed (mechanically and electrically) and commissioned. Commission the device as per product manual.

Two conditions must be fulfilled in order for a CANopen device to operate on a network:

• It must have a unique Node ID

• It must have the same communication Baud rate as all other devices on the network

5.2   Address and baud rate

Up to 64 devices can be addressed in one CAN bus network branch and up to 127 devices in the extended network. Each device is identified by a unique address or Node ID. The default Node ID for a Lexium MDrive is 41_h.

The default baud rate is 1 MBaud (1000 kbps).

Each device must be assigned a unique node address, i.e. any given node address may be assigned only once in the network.

The baud rate MUS match the baud rate setting of the network into which the device is being installed.

5.3   Commissioning via supplied utility

The NODE ID and BAUD rate may be configured using the supplied utility: The CANOpen Configuration Utility, which is part of the Lexium MDrive Software Suite.

Instructions for installing and using this tool to commission your Lexium MDrive CANopen device are found in the Lexium MDrive Software Suite Manual.

Both the software and user manual are available for download at:

www.motion.shneider-electric.com

5.4   Commissioning via Layer Setting Services (LSS)

The device may be commissioned using CAN Layer Setting Services (LSS). Reference CiA DSP305.

The scope of LSS is to allow the Node ID and communication baud rate to

be read or written through the network.

The Node ID and baud rate must be set using LSS commands. It is recommended that the parameters be set prior to the installation of the device into a network.

LSS messages are 8 bytes in length.
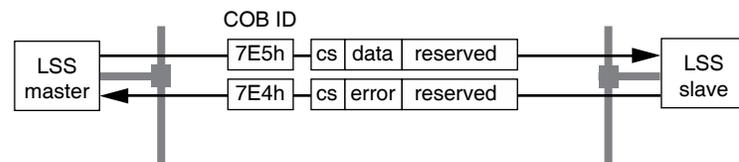
Figure 5.1: LSS message structure

The LSS message consists of a COB IDs specific toe the LSS master and LSS slave.

The command specifier identifies the action to be taken.

COB ID   LSS commands use two specific COB IDs to request and respond to LSS commands:

- $07E5_h$ – COB ID of the LSS slave device
- $07E4_h$ – COB ID of the LSS master device to which message

responses are sent

Switch Modes   There are two methods of initiating communications with the device to be commissioned:

- Switch mode global: can set all connected devices into configuration mode. Can be used to set the baud rate of multiple connected devices, can only be used to set the Node ID if one device is connected.

- Switch mode selective: can be used to set the parameters of a single device in the network using vendor specific objects such as the serial number to communicate directly to the device.

The switch mode commands are used to set the device into either and operational mode, or in configuration mode. In order to make changes to the parameters it must be in configuration mode.

5.5 Commissioning via switch mode global

Procedure   The following procedure will step through configuring the Node ID and Baud rate parameters from the perspective of a single device connected to the LSS master.

1) Transmit the command to the device setting it into switch mode global — configuration mode:

7E5h	0	1	2	3	4	5	6	7	8
	04	01	00	00	00	00	00	00	00

COB ID

01h = configuration mode

04h = switch mode

Reserved
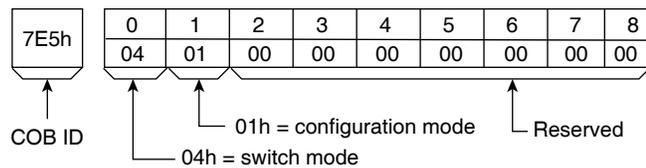
This will place the device in configuration mode. Because this is an unacknowledged LSS service there will be no response.

2) Set the new Node ID (42_h used as an example).

request

7E5h	0	1	2	3	4	5	6	7	8
	11	42	00	00	00	00	00	00	00

COB ID

Node ID = 42h

11h = set Node ID

Reserved

response

7E4h	0	1	2	3	4	5	6	7	8
	04	00	00	00	00	00	00	00	00

COB ID

00h = success
01h = failure

04h = switch mode global

Reserved
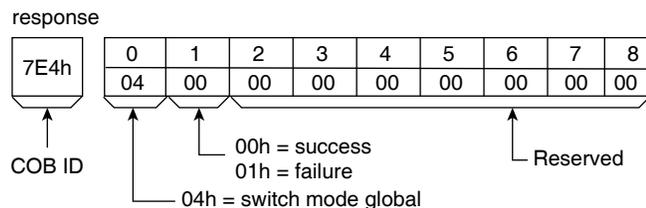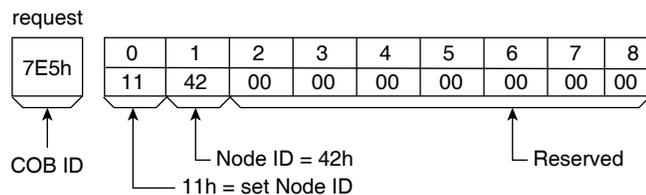
3) Set the desired baud rate using the table below as a reference, for example purposes a bitrate of 50 kbps will be used):

Table index	Baud rate (kbps)
00_h	1000 (default)

01h	800*
02h	500
03h	250
04h	125
05h	100
06h	50
07h	20
08h	10

*Not available if using MD-CC500-000 USB to CANopen converter cable.

request

7E5h	0	1	2	3	4	5	6	7	8
	13	00	06	00	00	00	00	00	00

COB ID

Reserved

Table index (see table)

Table selector (always 00h)

13h = set bit rate

response

7E4h	0	1	2	3	4	5	6	7	8
	13	00	00	00	00	00	00	00	00

COB ID

00h = success
01h = failure

Reserved

13h = set bit rate
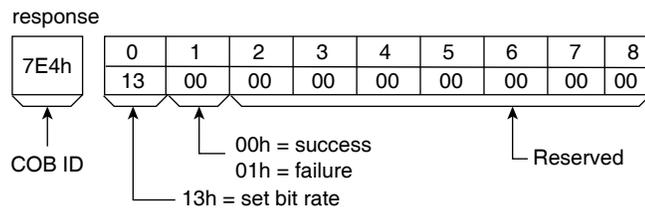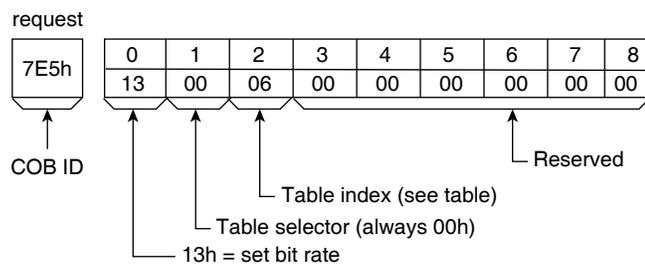
4) Save the changed parameters using the "Store configuration" service (17h)

request

7E5h	0	1	2	3	4	5	6	7	8
	17	00	00	00	00	00	00	00	00

Reserved

COB ID

17h = store configuration

response

7E4h	0	1	2	3	4	5	6	7	8
	17	00	00	00	00	00	00	00	00

COB ID

00h = success
01h = saving not supported
02h = failure

Reserved

17h = store configuration
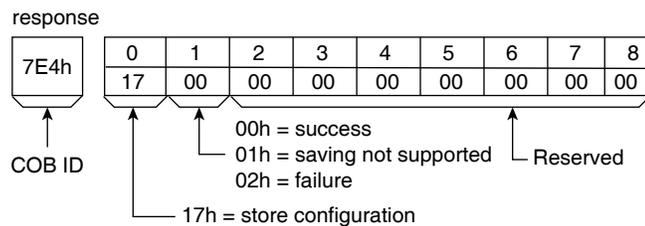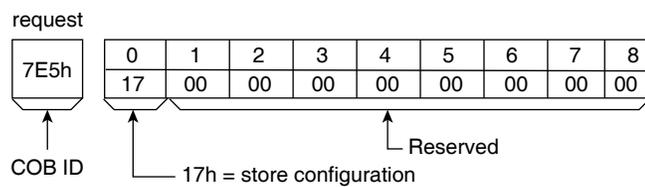
5)   Change the mode from configuration to operational

7E5h	0	1	2	3	4	5	6	7	8
	04	00	00	00	00	00	00	00	00

COB ID   00h = operation mode   Reserved

04h = switch mode global
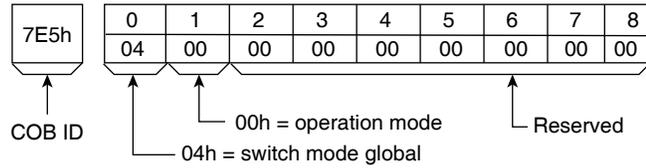
6)   The new Node ID is now active. To make the new bit rate active, cycle power to the device.

7)   The device is now commissioned and ready to be placed in a network having a Node ID of 42_h and a bitrate of 50 kbps.

5.6   Commissioning via switch mode selective

Using switch mode selective it is possible to isolate a single unit installed on a CANopen network and commission it alone by sending out a four-part signature.

The signature consists of

- VendorID = $0800005A_h$
- Product code = 00
- Revision number = 507_h
- Serial number

The first three parameters are identical throughout the product line. The serial number may be retrieved from the label on the device or from the Indexes 1018_h4 or 5002_h1.

Using only the numeric portion of the serial, convert it to a 4 byte hex number, for example:

```
341070398 = 1454523E
```

Procedure   8)   Send the four parameters to place the desired device in configuration mode.

request

	0	1	2	3	4	5	6	7	8
7E5h	40	1B	02	00	00	00	00	00	00

COB ID   └ 40h = address vendor id   └ vendor id 021Bh   └ Reserved

request

	0	1	2	3	4	5	6	7	8
7E5h	41	00	00	00	00	00	00	00	00

COB ID   └ 40h = address product code   └ product code   └ Reserved

request

	0	1	2	3	4	5	6	7	8
7E5h	42	07	05	00	00	00	00	00	00

COB ID   └ 42h = address revision   └ revision 0507   └ Reserved

request

	0	1	2	3	4	5	6	7	8
7E5h	43	3E	52	54	14	00	00	00	00

COB ID   └ 43h = address sn   └ sn 1451523Eh   └ Reserved

9)   Perform steps 2 – 7 of the Switch Mode Global procedure to commission the device.

6 Operation

6

> ## ⚠ WARNING
>
> **UNINTENDED OPERATION**
>
> The product is unable to detect an interruption of the network link if
>
> - Do not write values to reserved parameters.
> - Do not write values to parameters unless you fully understand the function.
> - Run initial tests without coupled loads.
> - Verify that the system is free and ready for the movement before changing parameters.
> - Verify the use of the word sequence with fieldbus communication.
> - Do not establish a fieldbus connection unless you have fully understood the communication principles.
>
> **Failure to follow these instructions can result in death, serious injury or equipment damage.**

The section "Operation" describes the basic operating states, operating modes and functions of the device.

6.1 Operating states

6.1.1 State diagram

After switching on and when an operating mode is started, the product goes through a number of operating states.

The state diagram (state machine) shows the relationships between the operating states and the state transitions.

The operating states are monitored and influenced by internal monitoring functions and system functions such as temperature monitoring or current monitoring.

Graphical representation   The state diagram is represented as a flowchart
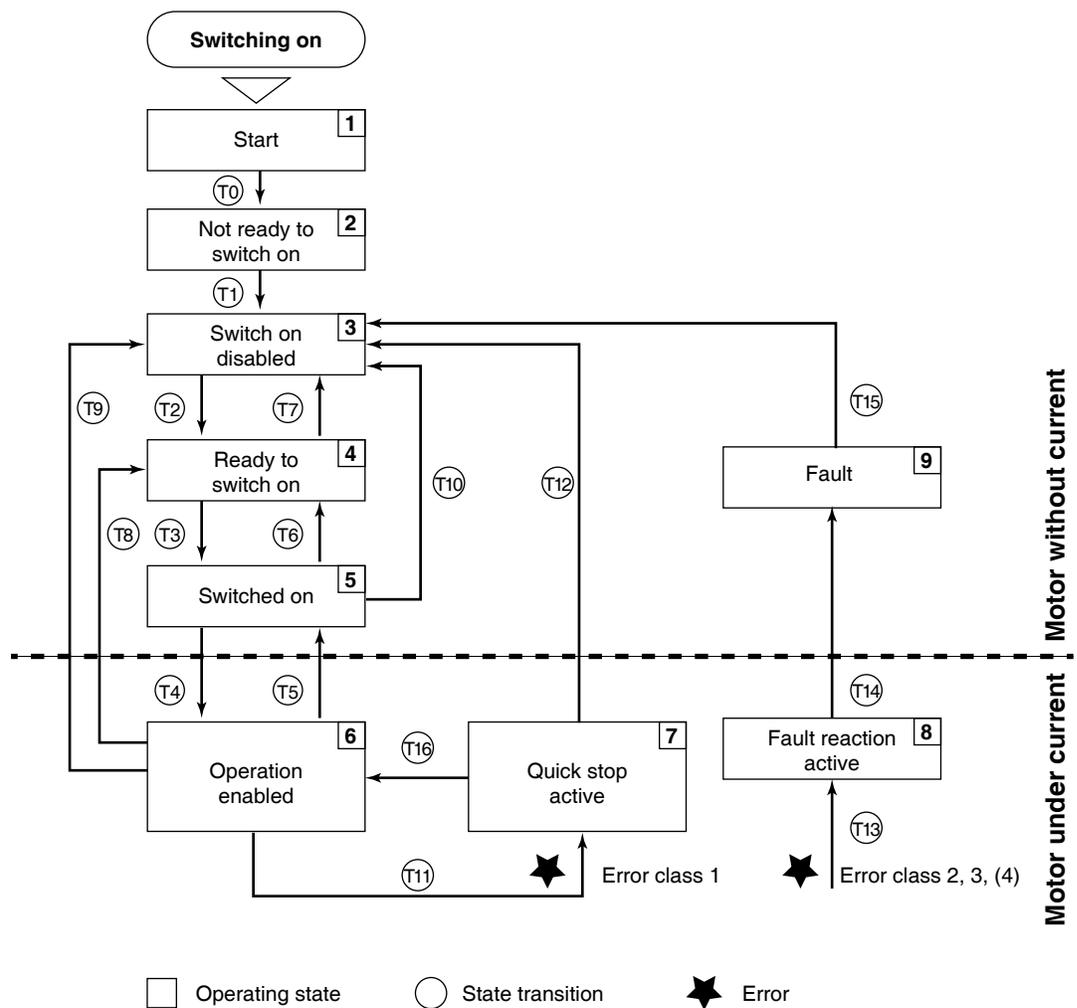
Figure 6.1: State diagram

Operating states

	Operating state	Description
1	Start	Controller supply voltage switched on Electronics are initialized
2	Not Ready To Switch On	The power stage is not ready to switch on
3	Switch On Disable	Impossible to enable the power stage
4	Ready To Switch On	The power stage is ready to switch on.
5	Switched On	Power stage is switched on
6	Operation Enabled	Power stage is enabled Selected operating mode is active
7	Quick Stop Active	"Quick Stop" is being executed
8	Fault Reaction Active	Error response is active
9	Fault	Error response terminated Power stage is disabled

Table 6.1: Operating states

Error class The product triggers an error response if an error occurs. Depending upon the severity of the error, the device responds in accordance with one of the following error classes:

Error Class	Response	Description
0	Warning	A monitoring function has detected a problem. No interruption of the movement.
1	"Quick Stop"	Motor stops with "Quick Stop", the power stage remains enabled.
2	Qu ick Stop" with switch-off	Motor stops with "Quick Stop", the power stage is disabled after standstill has been achieved.
3	Fatal error	The power stage is immediately disabled without stopping the motor first.
4	Uncontrolled operation	The power stage is immediately disabled without stopping the motor first. The error can only be reset by switching off the product.

Table 6.2: Error class

Error response The state transition T13 (error class 2, 3 or 4) initiates an error response as soon as an internal occurrence signals an error to which the device must react.

Error Class	State from -> to	Description
2	x -> 8	Stop movement with "Quick Stop" Power stage is disabled
3, 4 or Safety function STO	x -> 8 -> 9	Power stage is disabled immediately, even if "Quick Stop" is still active.

An error can be triggered by a temperature sensor, for example. The product cancels the running movement and performs an error response, for example stopping with «Quick Stop» or disabling the power stage. Subsequently, the operating state changes to 9 Fault. To exit the 9 Fault operating state, the cause of the error must be remedied and a Fault Reset must be executed.

Resetting an error message   A "Fault Reset" resets an error message.

6.2   Control and status

6.2.1   Controlling the state machine (control word 6040$_h$)

`Control word 6040`$_h$ is a mandatory index which sets the operating states and modes of the state machine.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
mfg specific					reserved	op mode specific	halt	fault reset	op mode specific			enable op	quick stop	enable voltage	switch on

MSB   LSB

Command	Bits of the control word					Transitions
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X	1	1	0	2, 6, 8
Switch on	0	0	1	1	1	3
Switch on + enable operation	0	1	1	1	1	3 + 4
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, 16
Fault reset	0 –> 1	X	X	X	X	15

Op mode	Bits of the control word			
	Bit 8	Bit 6	Bit 5	Bit 4
Profile position	Halt	Abs/rel	Change set	New setpoint
Profile velocity	Halt	Reserved	Reserved	Reserved
Homing	Halt	Homing start	Reserved	Reserved

Table 6.3: Control word value range

Object description

Index	6040$_h$
Name	Control word
Object code	VAR
Data type	Unsigned16
Category	Mandatory

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	See table 6.3
Default value	Device and operation mode specific

6.2.2   Indication of the operating state (status word 6041:0h)

The status word 6041:0$_h$ provides information on the operating state of the device and the processing status of the operating mode

15 14	13 12	11	10	9	8	7	6 5	4	3 2 1 0
mfg specific	op mode specific	Int limit active	target reached	remote	mfg specific	warn	status bits	voltage enabled	status bits

MSB   LSB
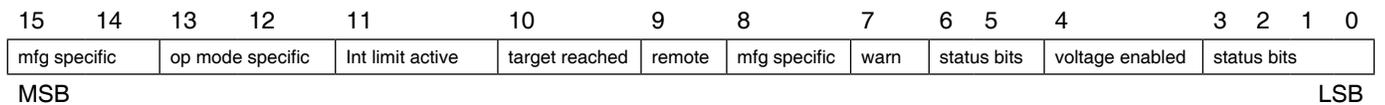
Status bits Bit 6: Switch on disabled
Bit 5: Quick stop
Bit 3: Fault
Bit 2: Operation enabled
Bit 1: Switch on
Bit 0: Ready to switch on

Status	Bits of the status word					
	Bit 6	**Bit 5**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Not Ready To Switch On	0	X	0	0	0	0
Switch On Disabled	1	X	0	0	0	0
Ready To Switch On	0	1	0	0	0	1
Switched On	0	1	0	0	1	1
Operation Enabled	0	1	0	1	1	1
Quick Stop Active	0	0	0	1	1	1
Fault Reaction Active	0	X	1	1	1	1
Fault	0	X	1	0	0	0

Table 6.4: Status word status bit states

Bit 4: Voltage Enabled

Bit 4=1 indicates whether the DC bus voltage is correct. If the voltage is missing or is too low, the device does not transition from operating state 3 to operating state 4.

Bit 5: Quick Stop Active

When reset, this bit indicates that the drive is reacting on a quick stop request. Bits 0, 1 and 2 of the `statusword` must be set to 1 to indicate that the drive is capable to regenerate. The setting of the other bits indicates the status of the drive (e.g. the drive is performing a quick stop as result of a reaction to a non-fatal fault. The fault bit is set as well as bits 0, 1 and 2).

Bit 7: Warning

A drive warning is present if bit 7 is set. The cause means no error but a state that has to be mentioned, e.g. temperature limit, job refused. The status of the drive does not change. The cause of this warning may be found by reading the fault code parameter. The bit is set and reset by the device.

Bit 8: Manufacturer Specific

This bit may be used by a drive manufacturer to implement any manufacturer specific functionality. Not used by Lexium MDrive products.

Bit 9: Remote

If bit 9 is set, then parameters may be modified via the CAN-network, and the drive executes the content of a command message. If the bit remote is reset, then the drive is in local mode and will not execute the command message. The drive may transmit messages containing valid actual values like a `position_actual_value`, depending on the actual drive configuration. The drive will accept accesses via service data objects (SDOs) in local mode.

Bit 10: Target Reached

If bit 10 is set by the drive, then a setpoint has been reached (torque, speed or position depending on the `modes_of_operation`). The change of a target value by software alters this bit. If `quickstop_option_code` is 5, 6, 7 or 8, this bit must be set, when the quick stop operation is finished and the drive is halted. If Halt occurred and the drive has halted then this bit is set too.

Bit 11: Internal Limit Active

This bit set by the drive indicates, that an internal limitation is active (e.g. `position_range_limit`).

Bits 12-13: Operation Mode Specific

Op mode	Bits of the control word	
	Bit 13	**Bit 12**
Profile position	Following error	Set point acknowledge
Profile velocity	Max slippage error	Speed
Homing	Homing error	Homing attained

Table 6.5: Operation mode status

Bit 14-15: Unused

6.3   Option code objects

6.3.1   Abort connection (6007$_h$)

This object shall indicate what action shall be performed when one of the following events occurs: bus-off, heartbeat, life guarding, NMT stopped state entered, reset application, and reset communication

Object description

Index	6007$_h$
Name	Abort connection
Object code	VAR
Data type	Integer16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	0 to 3
Default value	0

Value	Meaning
0	No action
1	Fault signal
2	Disable voltage command
3	Quick stop command

Table 6.6: Abort Connection Option Code

6.3.2   Error code (603F$_h$)

This object shall provide the error code of the last error occurred in the drive device. This object provides the same information as the lower 16-bit of sub-index 01h of the pre-defined error field (1003$_h$).

Object description

Index	603F$_h$
Name	Error code
Object code	VAR
Data type	Unsigned16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned16
Default value	0

6.3.3   Quick stop (605A$_h$)

This object shall indicate what action is performed when the quick stop function is executed. The slow down ramp is the deceleration value of the used mode of operations.

Object description

Index	605A$_h$
Name	Quick stop
Object code	VAR
Data type	Integer16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	No
Value range	0 to 6
Default value	2

Value	Meaning
0	Disable drive function
1	Slow down on slow down ramp and transit into Switch On Disabled
2	Slow down on quick stop ramp and transit into Switch On Disabled
3	Slow down on current limit and transit into Switch On Disabled
4	Slow down on voltage limit and transit into Switch On Disable
5	Slow down on slow down ramp and stay in Quick Stop Active
6	Slow down on quick stop ramp and stay in Quick Stop Active

Table 6.7: Quick stop option codes

6.3.4 Shutdown (605B$_h$)

This object shall indicate what action is performed if there is a transition from Operation Enabled state to Ready To Switch On state. The slow down ramp is the deceleration value of the used mode of operations.

Object description

Index	605B$_h$
Name	Shutdown
Object code	VAR
Data type	Integer16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	No
Value range	0 to 1
Default value	0

Value	Meaning
0	Disable drive function (Switch off the drive power stage)
1	Slow down with slow down ramp; disable of the drive function

Table 6.8: Shutdown option codes

6.3.5 Disable operation (605C$_h$)

This object shall indicate what action is performed if there is a transition from Operation Enabled state to Switched on state. The slow down ramp is the deceleration value of the used mode of operation.

Object description

Index	605C$_h$
Name	Disable operation
Object code	VAR
Data type	Integer16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	No
Value range	0 to 1
Default value	0

Value	Meaning
0	Disable drive function (Switch off the drive power stage)
1	Slow down with slow down ramp; disable of the drive function

Table 6.9:: Disable operation option codes

6.3.6   Halt (605D$_h$)

Indicates what action is performed if there is a transition from Operation Enabled state to Switched on state. The slow down ramp is the deceleration value of the used mode of operations.

Object description

Index	605D$_h$
Name	Halt
Object code	VAR
Data type	Integer16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	No
Value range	1 or 2
Default value	2

Value	Meaning
1	Slow down on slow down ramp and stay in Operation Enabled
2	Slow down on quick stop ramp and stay in Operation Enabled

Table 6.10: Halt option codes

6.3.7   Fault reaction (605E$_h$)

Indicates what action is performed when fault is detected in the PDS. The slow down ramp is the deceleration value of the used mode of operations.

Object description

Index	605E$_h$
Name	Halt
Object code	VAR
Data type	Integer16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	No
Value range	0 to 2
Default value	2

Value	Meaning
0	Disable drive function, motor is free to rotate
1	Slow down on slow down ramp
2	Slow down on quick stop ramp

Table 6.11: Fault reaction option codes

6.4 Supported modes of operation

The function of the product depends on the selected modes of operation. It is not possible to operates the modes in parallel. The user must select a mode to operate in. An example of exclusive functions are Profile Velocity and Profile Position modes. Supported modes are:

1) Profile position
2) Homing mode
3) Profile velocity

The product allows the user to switch dynamically from operation mode to operation mode.

The mode of operation is set or read using Mode of Operation (6060_h) and Mode of Operation display (6061_h).

6.4.1 Mode of operation (6060_h)

This object shall indicate the requested operation mode.

Object description

Index	6060_h
Name	Mode of operation
Object code	VAR
Data type	Integer8
Category	Optional

Entry description

Sub-index	00_h
Access	rw
PDO mapping	No
Value range	1, 3 or 6
Default value	1

Value	Meaning
1	Profile Position
3	Profile Velocity
6	Homing

Table 6.12: Mode of operation

6.4.2　Mode of operation display (6061$_h$)

The Modes of Operation Display shows the current mode of operation. The meaning of the returned value corresponds to that of the Modes of Operation option code (index 6060h).

Object description

Index	6061$_h$
Name	Mode of operation display
Object code	VAR
Data type	Integer8
Category	Optional

Entry description

Sub-index	00$_h$
Access	ro
PDO mapping	Yes
Value range	1, 3 or 6
Default value	1

6.4.3　Supported drive modes (6502$_h$)

This object shall provide information on the supported drive modes.

31　　16	15　　10	9 8	7	6	5	4	3	2	1	0
Mfg specific	Reserved	ns	Cyclic sync position	ns	Homing	ns	Profile torque	Profile velocity	ns	Profile position

MSB　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　LSB

ns = not supported

Object description

Index	6502$_h$
Name	Supported drive modes
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	ro
PDO mapping	No
Value range	Unsigned32
Default value	00000025$_h$

6.5   Profile position mode

6.5.1   Overview

A `target_position` is applied to the trajectory generator. It is generating a `position_demand_value` for the position control loop described in the position control function section. These two function blocks are optionally controlled by individual parameter sets.

At the input to the trajectory generator, parameters may have optional limits applied before being normalized to internal units. Normalized parameters are denoted with an asterisk. The simplest form of a trajectory generator is just to pass through a target_position and to transform it to a `position_demand_value` with internal units (increments) only.
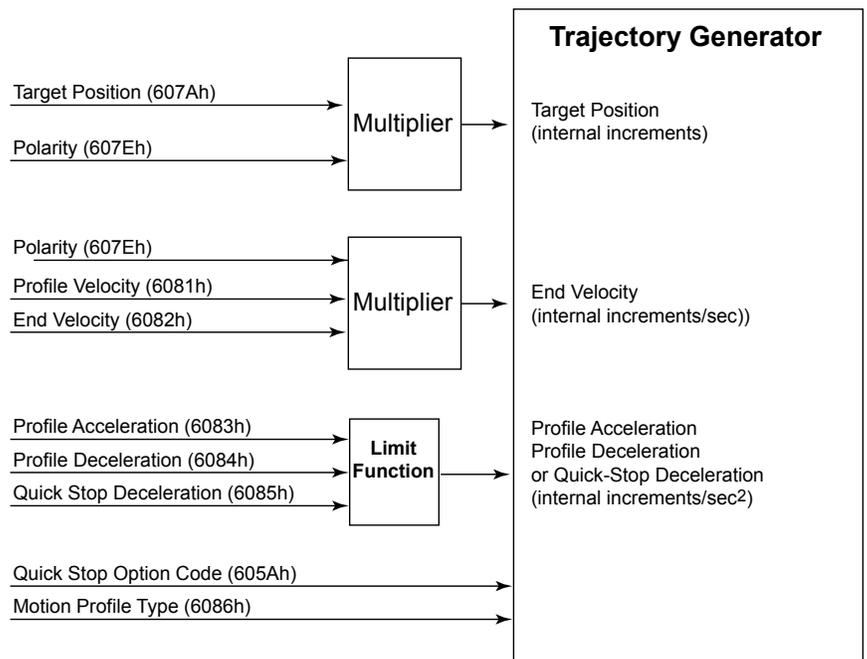
Figure 6.2: Trajectory generator for profile position block diagram

Input data   The input data objects for profile position:

Object	Name	Description
607A$_h$	`target_position`	Defines the targeted absolute or relative position for a move
607E$_h$	`polarity`	Sets the polarity for a position or speed commands
6081$_h$	`profile_velocity`	Sets the velocity for profile position motion
6082$_h$	`end_velocity`	Sets the velocity upon reaching target.
6083$_h$	`profile_acceleration`	Sets the acceleration for profile posistion and profile velocity
6084$_h$	`profile_deceleration`	Sets the deceleration for profile posistion and profile velocity
6085$_h$	`quick_stop_decel`	Sets the deceleration for quick stop active

Table 6.13: Input data objects for profile position

Output data   The output data objects for profile position:

Object	Name	Description
607A$_h$	`position_demand_value`	Displays the motor position

Table 6.13: Input data objects for profile position

6.5.2   Functional description

There are two different ways to apply `target_positions` to a drive, are supported by this device profile.

1. Set of set-points:

    After reaching the `target_position` the drive unit immediately processes the next `target_position` which results in a move where the velocity of the drive normally is not reduced to zero after achieving a set-point.

2. Single set-point:

    After reaching the `target_position` the drive unit signals this status to a host computer and then receives a new set-point. After reaching a `target_position` the velocity normally is reduced to zero before starting a move to the next set-point.

The two modes are controlled by the timing of the bits `new_set-point` and `change_set_immediately` in the controlword and `set-point_acknowledge` in the `statusword`.

These bits allow to set up a request-response mechanism in order to prepare a set of set-points while another set still is processed in the drive unit. This minimizes reaction times within a control program on a host computer.
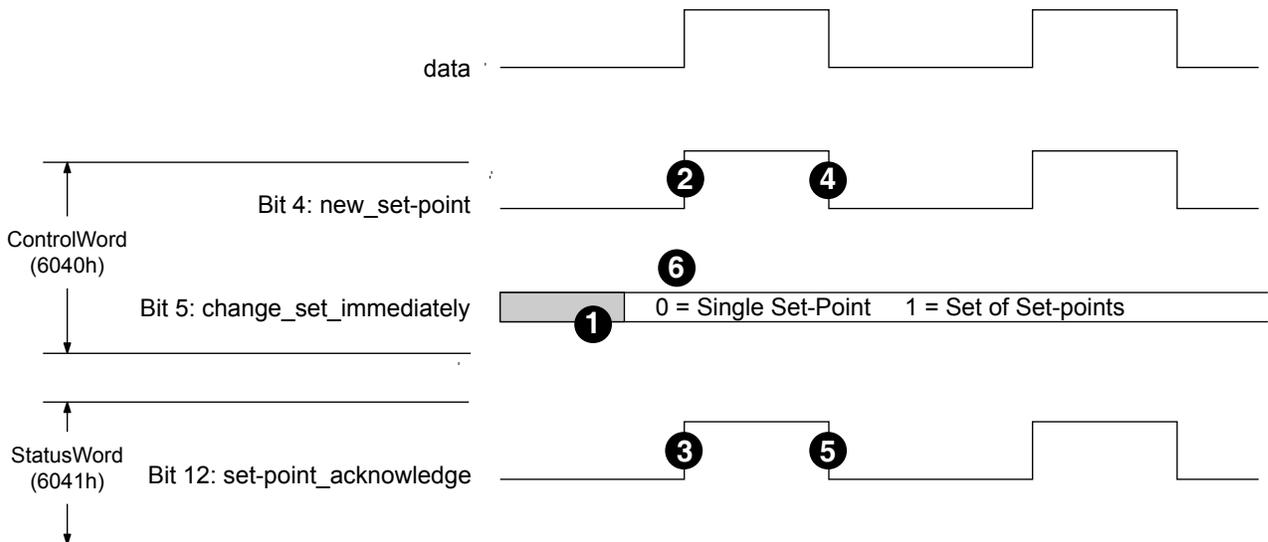
Figure 6.3: Set-point transmission from host (see table 6:14 for states)

Bit state #	Meaning
(1)	Single set-point is expected by device
(2)	Host signals "data is valid" new set-point = 1
(3)	Device response: sets bit 12, set-point acknowledge = 1
(4)	Data is validated, host may release new set-point
(5)	Device response: sets bit 12, set-point acknowledge = 0 Device ready to receive new data
(6)	Indicates state of change_set_immediately = 1

Table 6.14: Set-point transmission from host bit states

Figure 6.3, Figure 6.4 and Figure 6.5 illustrate the difference between the "set of set-points" mode and the "single set-point" mode. The initial status of the bit `change_set_immediately` in the controlword determines which mode is used. Trapezoidal moves are used as this is the only `motion_profile_type` the Lexium MDrive CANopen supports.

If the bit `change_set_immediately` is "0" (shaded area in Figure 6.3) a single set-point is expected by the drive *(1)*. After data is applied to the drive, a host signals that the data is valid by changing the bit `new_set-point` to "1" in the controlword *(2)*. The drive responds with `set-point_acknowledge` set to "1" in the statusword *(3)* after it recognized and buffered the new valid data. Now the host may release new_setpoint *(4)* and afterwards the drive signals with set-point_acknowledge equal "0" its ability to accept new data again *(5)*. In Figure 6.4 this mechanism results in a velocity of zero after ramping down in order to reach a `target_position` X_1 at T_1. After signalling to the host, that the set-point is reached like described above, the next `target_position` X_2 is processed at T_2 and reached at T_3.
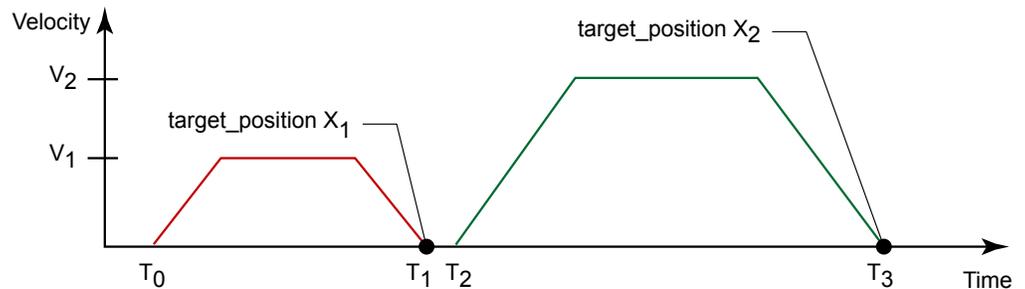
Figure 6.4: Single set-point mode (move after a move) 6040_h bit 5=0

With `change_set_immediately` set to "1" *(6)*, symbolized by the clear area in Figure 6.3, the host advises the drive to apply a new set-point immediately after reaching the last one. The relative timing of the other signals is unchanged. This behavior causes the drive to already process the next set-point X_2 and to keep its velocity when it reaches the `target_position` X_1 at T_1. Then drive moves immediately to the already calculated next `target_position` X_2.
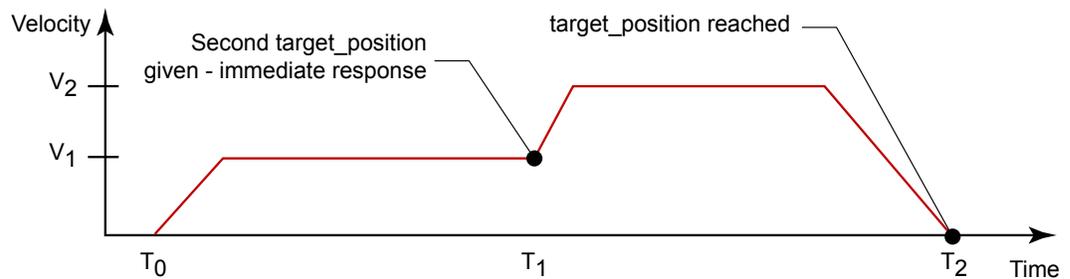
Figure 6.5: Set of set-points (move on a move) 6040h bit 5=1

6.5.3   Control word definition for profile position

15 ... 9	8	7	6	5	4	3 ... 0
See 6.2.1	halt	See 6.2.1	abs/rel	change set immediately	new set point	See 6.2.1

Bit	Name	Value	Meaning
4	New set-point	0	Does not assume target position
		1	Assume target position
5	Change set immediately	0	Finish the actual positioning and then start the next positioning
		1	Interrupt the actual positioning and start the next positioning
6	abs/rel	0	Target position is an absolute value
		1	Target position is a relative value
8	Halt	0	Execute positioning
		1	Stop motion with profile deceleration

Table 6.15: Profile position mode control word (6040_h) bit state meanings

6.5.4   Status word definition for profile position

15 ... 14	13	12	11	10	9 ... 0
See 6.2.2	following error	set-point acknowledge	See 6.2.2	target reached	See 6.2.2

Bit	Name	Value	Meaning
10	Target reached	0	Halt=0: Target position not reached Halt=1: Axis decelerating
		1	Halt=0: Target position reached Halt=1: Axis velocity is 0
12	Set-point acknowledge	0	Trajectory generator has not assumed the positioning values yet
		1	Trajectory generator has assumed the positioning values
13	Following error	0	No following error
		1	Following error

Table 6.16: Profile position mode status word (6041_h) bit state meanings

Position, velocity and acceleration objects

6.5.5   607A$_h$ Target position

The target position is the position that the drive should move to in position profile mode using parameters such as velocity, acceleration, deceleration, motion profile type etc. The target position is given in terms of 51,200 units per motor shaft revolution. The target position will be interpreted as absolute or relative depending on the absolute relative flag (bit 6) in the controlword.

Object description

Index	607A$_h$
Name	Target position
Object code	VAR
Data type	Integer32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	8000 0000$_h$ to 7FFF FFFF$_h$
Default value	0000 0000$_h$

6.5.6   607E$_h$ Polarity

Position demand value and position actual value are multiplied by 1 or -1, depending on the value of the polarity flag.

7	6	5 ... 0
position polarity	velocity polarity	reserved

Object description

Index	607E$_h$
Name	Polarity
Object code	VAR
Data type	Unsigned8
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned8
Default value	00$_h$

6.5.7   6081ₕ Profile velocity

The profile velocity is the velocity normally attained at the end of the acceleration ramp during a profiled move and is valid for both directions of motion. The profile velocity is given in steps per second.

Object description

Index	6081$_h$
Name	Profile velocity
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned32
Default value	512000$_d$

6.5.8   6082ₕ End velocity

The end velocity defines the velocity which the drive must have on reaching the target position. Normally, the drive stops at the target position, i.e. the end_velocity = 0. The end velocity is given in the same units as profile velocity.

Object description

Index	6082$_h$
Name	Polarity
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned32
Default value	51200$_d$

6.5.9   6083$_h$ Profile acceleration

Profile acceleration is given in steps/sec^2

bject description

Index	6083$_h$
Name	Profile acceleration
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned32
Default value	100000$_d$

6.5.10   6084$_h$ Profile deceleration

Profile deceleration is given in steps/sec^2

Object description

Index	6084$_h$
Name	Profile deceleration
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned32
Default value	100000$_d$

6.5.11   6085$_h$ Quick stop deceleration

This object shall indicate the configured deceleration used to stop the motor when the quick stop function is activated and the quick stop code object ($Idx\ 605A_h$) is set to 2 or 6. The quick stop deceleration is also used if the fault reaction code object ($Idx\ 605Eh$) is 2 and the halt option code object ($Idx\ 605Dh$) is 2. The value shall be given in the same physical unit as profile acceleration object ($Idx\ 6083h$).

Object description

Index	6085$_h$
Name	Quick stop deceleration
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned32
Default value	200000$_d$

6.5.12   6086$_h$ Motion profile type

The motion profile type is used to select the type of motion profile used to perform a move. The represented devices are fixed at value 0: linear ramp (trapezoidal profile)

bject description

Index	6085$_h$
Name	Motion profile type
Object code	VAR
Data type	Integer16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	0$_d$
Default value	0$_d$

6.5.13   Profile position application example

The represented device(s) support relative and absolute moves to position. Using either relative or absolute moves, the user can also select (by the control word data) if the target position should be reached before another target position is allowed (finish first) or if the product should execute a newly received target position even if still in motion (immediate).

The below example sets typical motion profile commands a system would configure[1], enabling the motor power[2] and the four different move types[3] supported in profile position mode using SDOs with Node ID41$_h$.

[1] Typical motion profile commands could be set each time on power up from host or set using a configuration file and stored to NVM once.

[2] Enabling the motor power only has to be done once on power up.

[3] The Control Word data selects the move type.

All values shown are hexadecimal.

ID	RTR	Data String	Action
Typical motion parameters			
0641	00	2F 04 22 00 50 00 00 00	Set run current to 80%
0641	00	23 84 60 00 40 42 0F 00	Set deceleration to 1M steps/sec^2
0641	00	23 83 60 00 40 42 0F 00	Set acceleration to 1M steps/sec^2
0641	00	23 81 60 00 00 D0 07 00	Set max velocity to 512000 steps/sec
Enable motor power - DSP402 state machine			
0641	00	2B 40 60 00 06 00 00 00	Ready to switch on
0641	00	2B 40 60 00 07 00 00 00	Switched on
0641	00	2B 40 60 00 0F 00 00 00	Operation enable
Set to profile position mode			
0641	00	2F 60 60 00 01 00 00 00	Set to profile position mode
Perform absolute move, finish before performing additional moves			
0641	00	23 7A 60 00 30 75 00 00	Set target position to 30000 steps
0641	00	2B 40 60 00 1F 00 00 00	Set control word bit 4 to 1
0641	00	2B 40 60 00 0F 00 00 00	Set control word bit 4 to 0
Perform absolute move, move immediate			
0641	00	23 7A 60 00 B8 0B 00 00	Set target position to 3000 steps
0641	00	2B 40 60 00 3F 00 00 00	Set control word bit 4 to 1
0641	00	2B 40 60 00 2F 00 00 00	Set control word bit 4 to 0
Perform relative move, finish before performing additional moves			
0641	00	23 7A 60 00 A0 86 01 0	Set target position to 100000
0641	00	2B 40 60 00 5F 00 00 00	Set control word bit 4 to 1
0641	00	2B 40 60 00 4F 00 00 00	Set control word bit 4 to 0
Perform relative move, move immediate			
0641	00	23 7A 60 00 B8 0B 00 00	Set target position to 3000 steps
0641	00	2B 40 60 00 7F 00 00 00	Set control word bit 4 to 1
0641	00	2B 40 60 00 6F 00 00 00	Set control word bit 4 to 0

Table 6.17: Profile position mode application example

6.6 Profile velocity mode

6.6.1 Overview

The profile velocity mode covers the following sub-functions:

- Demand value input via trajectory generator
- Velocity capture using position sensor or velocity sensor
- Velocity control function with appropriate input and output signals
- Monitoring of the profile velocity using a window-function
- Monitoring of velocity actual value using a threshold
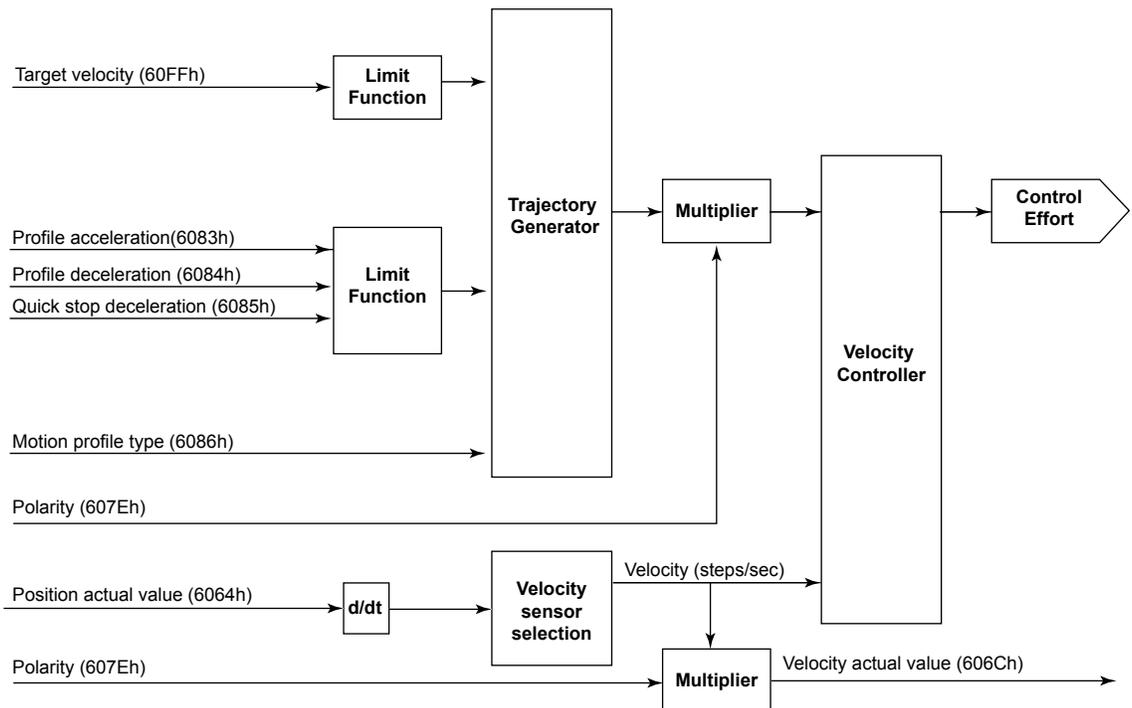
Figure 6.6: Profile velocity control function

6.6.2 Control word definition for profile velocity

15 ... 9	8	7	6 ... 4		3 ... 0
See 6.2.1	halt	See 6.2.1	reserved		See 6.2.1

Bit	Name	Value	Meaning
8	Halt	0	Execute the motion
		1	Stop axis

Table 6.18: Profile velocity mode control word (6040h) bit state meanings

6.6.3 Status word definition for profile velocity

15 ... 14	13	12	11	10	9 ... 0
See 6.2.2	max slippage error	speed	See 6.2.2	target reached	See 6.2.2

Bit	Name	Value	Meaning
10	Target reached	0	Halt=0: Target position not reached Halt=1: Axis decelerating
		1	Halt=0: Target position reached Halt=1: Axis velocity is 0
12	Speed	0	Speed is not equal to 0
		1	Speed is equal to 0
13	Max slippage error	0	Maximum slippage not reached
		1	Maximum slippage reached

Table 6.19: Profile velocity mode status word (6041h) bit state meanings

Profile velocity mode objects

6.6.4   606C$_h$ Velocity actual value

This object shall provide the actual velocity value derived either from the velocity sensor or the position sensor. The value shall be given in microsteps per second.

Object description

Index	606C$_h$
Name	Velocity actual value
Object code	VAR
Data type	Integer32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Integer32
Default value	0000 0000$_h$

6.6.5   60F8$_h$ Maximum slippage

This object shall indicate the configured maximal slippage of an asynchronous motor. When the max slippage has been reached, the corresponding bit 13 max slippage error in the statusword shall be set to 1. The reaction of the drive device, when the max slippage error occurs, is manufacturer-specific. This value shall be given in microsteps.

Object description

Index	60FF$_h$
Name	Maximum slippage
Object code	VAR
Data type	Integer32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Integer32
Default value	512$_d$

6.6.6   60FF$_h$ Target velocity

The Target Velocity is the input to the trajectory generator and the value is given in microsteps/second.

Object description

Index	60FF$_h$
Name	Target velocity
Object code	VAR
Data type	Integer32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	8000 0000$_h$ to 7FFF FFFF$_h$
Default value	0000 0000$_h$

6.6.7   Profile velocity application example

The represented device(s) suppor the ability to move in velocity mode. Once in Profile Velocity Mode, any new target velocity will be executed immediately.

The below example sets typical motion profile commands a system would configure[1], enabling the motor power[2] and sending a new target velocity using SDOs with Node ID41$_h$.

[1] Typical motion profile commands could be set each time on power up from host or set using a configuration file and stored to NVM once.

[2] Enabling the motor power only has to be done once on power up.

All values shown are hexadecimal.

ID	RTR	Data String	Action
Typical motion parameters			
0641	00	2F 04 22 00 50 00 00 00	Set run current to 80%
0641	00	23 84 60 00 40 42 0F 00	Set deceleration to 1M steps/sec^2
0641	00	23 83 60 00 40 42 0F 00	Set acceleration to 1M steps/sec^2
0641	00	23 81 60 00 00 D0 07 00	Set max velocity to 512000 steps/sec
Enable motor power - DSP402 state machine			
0641	00	2B 40 60 00 06 00 00 00	Ready to switch on
0641	00	2B 40 60 00 07 00 00 00	Switched on
0641	00	2B 40 60 00 0F 00 00 00	Operation enable
Set to profile velocity mode			
0641	00	2F 60 60 00 03 00 00 00	Set to profile velocity mode
Send new target velocity			
0641	00	23 FF 60 00 50 C3 00 00	Set target velocity 50000 steps/sec

Table 6.20: Profile velocity mode application example

6.7   Homing mode

6.7.1   Overview

This subsection describes the method by which a drive seeks the home position (also called, the datum, reference point or zero point). There are various methods of achieving this using limit switches at the ends of travel or a home switch (zero point switch) in mid-travel, most of the methods also use the index (zero) pulse train from an incremental encoder.
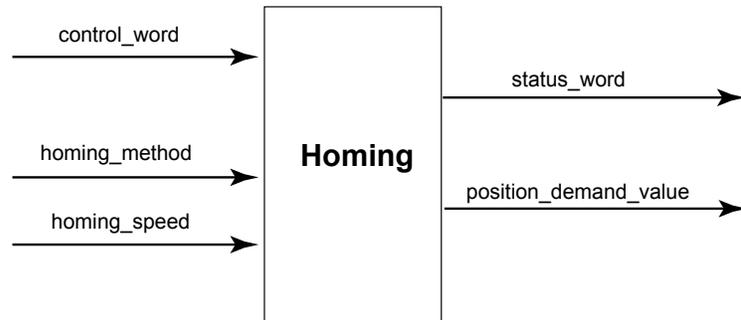
Figure 6.8 : The homing function

Input data description   The user can specify the speeds and the method of homing. There are two homing_speeds; in a typical cycle the faster speed is used to find the home switch and the slower speed is used to find the index pulse. The manufacturer is allowed some discretion in the use of these speeds as the response to the signals may be dependent upon the hardware used.

Output data description   There is no output data except for those bits in the statusword which return the status or result of the homing process and the demand to the position control loops.

6.7.2   Control word definition for homing mode

15 ... 9	8	7	6 ... 5	4	3 ... 0
See 6.2.1	halt	See 6.2.1	reserved	homing operation start	See 6.2.1

Bit	Name	Value	Meaning
4	Homing operation start	0	Execute the motion
		0 ⇨ 1	Start homing mode
		1	Homing mode active
		1 ⇨ 0	Interrupt homing mode
8	Halt	0	Execute the instruction of bit 4
		1	Stop axis

Table 6.21: Homing mode control word (6040h) bit state meanings

6.7.3 Status word definition for homing mode

15 ... 14	13	12	11	10	9 ... 0
See 6.2.2	homing error	homing attained	See 6.2.2	target reached	See 6.2.2

Bit	Name	Value	Meaning
10	Target reached	0	Halt=0: Target position not reached Halt=1: Axis decelerating
		1	Halt=0: Target position reached Halt=1: Axis velocity is 0
12	Homing Attained	0	Homing mode not yet complete
		1	Homing mode carried out successfully
13	Following error	0	No homing error
		1	Homing error

Table 6.22: Homing mode status word (6041h) bit state meanings

Homing mode objects

6.7.4 607C$_h$ Homing offset

This object shall indicate the configured difference between the zero position for the application and the machine home position (found during homing). During homing the machine home position is found and once the homing is completed the zero position is offset from the home position by adding the home offset to the home position. All subsequent absolute moves shall be taken relative to this new zero position. If this object is not implemented then the home offset shall be regarded as zero. The value of this object shall be given in micro steps. Negative values shall indicate the opposite direction.

Figure 6.7: The homing offset

Object description

Index	607C$_h$
Name	Homing offset
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned32
Default value	0$_d$

6.7.5   6098ₕ Homing method

The homing method object determines the method that will be used during homing.

Object description

Index	6098ₕ
Name	Homing method
Object code	VAR
Data type	Integer8
Category	Optional

Entry description

Sub-index	00ₕ
Access	rw
PDO mapping	Yes
Value range	0_d (no homing) $1 - 35_d$ (method)
Default value	0_d

Functional description of homing methods

Method 1: Homing on the negative limit switch and index pulse

Using this method the initial direction of movement is leftward if the negative limit switch is inactive (here shown as low). The home position is at the first index pulse to the right of the position where the negative limit switch becomes inactive.
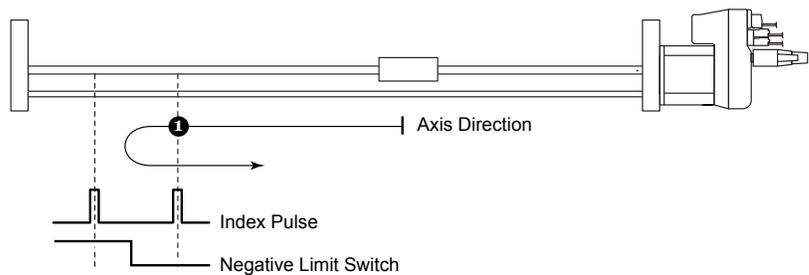
Figure 6.9: Homing on the negative limit switch and index pulse

Method 2: Homing on the positive limit switch and index pulse

Using this method the initial direction of movement is rightward if the positive limit switch is inactive (here shown as low). The position of home is at the first index pulse to the left of the position where the positive limit switch becomes inactive.
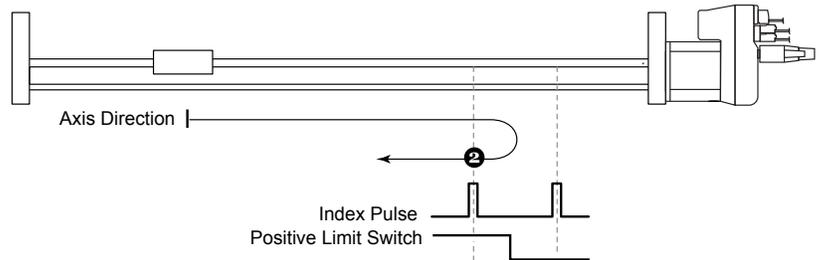
Figure 6.10: Homing on the positive limit switch and index pulse

Methods 3 and 4: Homing on the positive home switch and index pulse

Using methods 3 or 4 the initial direction of movement is dependent on the state of the home switch. The home position is at the index pulse to either to the left or the right of the point where the home switch changes state. If the initial position is sited so that the direction of movement must reverse during homing, the point at which the reversal takes place is anywhere after a change of state of the home switch.
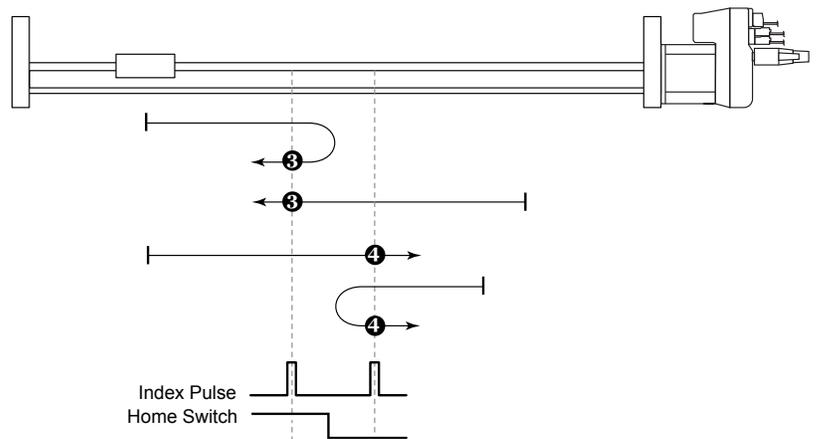
Figure 6.11: Homing on the positive home switch and index pulse

Methods 5 and 6: Homing on the negative home switch and index pulse

Using methods 5 or 6 the initial direction of movement is dependent on the state of the home switch. The home position is at the index pulse to either to the left or the right of the point where the home switch changes state. If the initial position is sited so that the direction of movement must reverse during homing, the point at which the reversal takes place is anywhere after a change of state of the home switch.
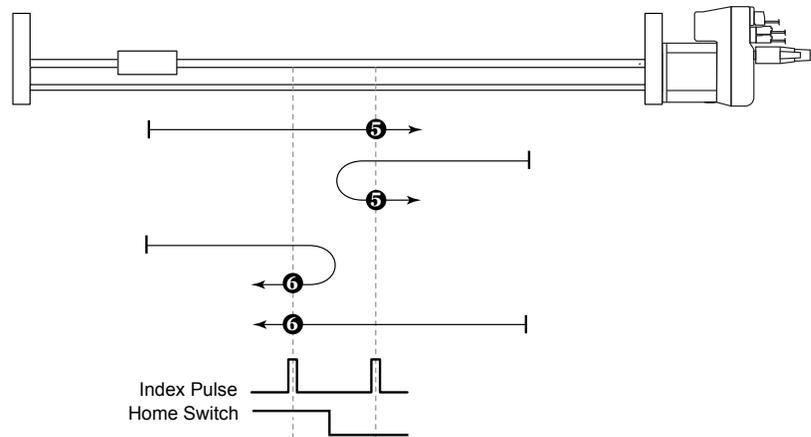
Figure 6.12: Homing on the negative home switch and index pulse

Methods 7 to 14: Homing on the Home Switch and Index Pulse

These methods use a home switch which is active over only portion of the travel, in effect the switch has a 'momentary' action as the axle`s position sweeps past the switch.

Using methods 7 to 10 the initial direction of movement is to the right, and using methods 11 to 14 the initial direction of movement is to the left except if the home switch is active at the start of the motion. In this case the initial direction of motion is Dependent on the edge being sought. The home position is at the index pulse on either side of the rising or falling edges of the home switch, as shown in the following two diagrams. If the initial direction of movement
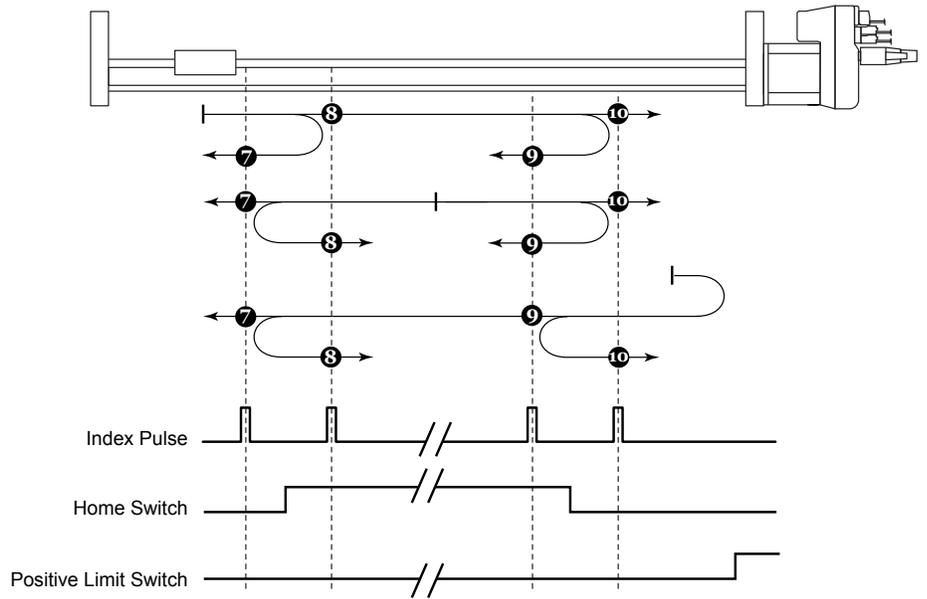
Figure 6.13: Homing on the home switch and index pulse - positive initial move
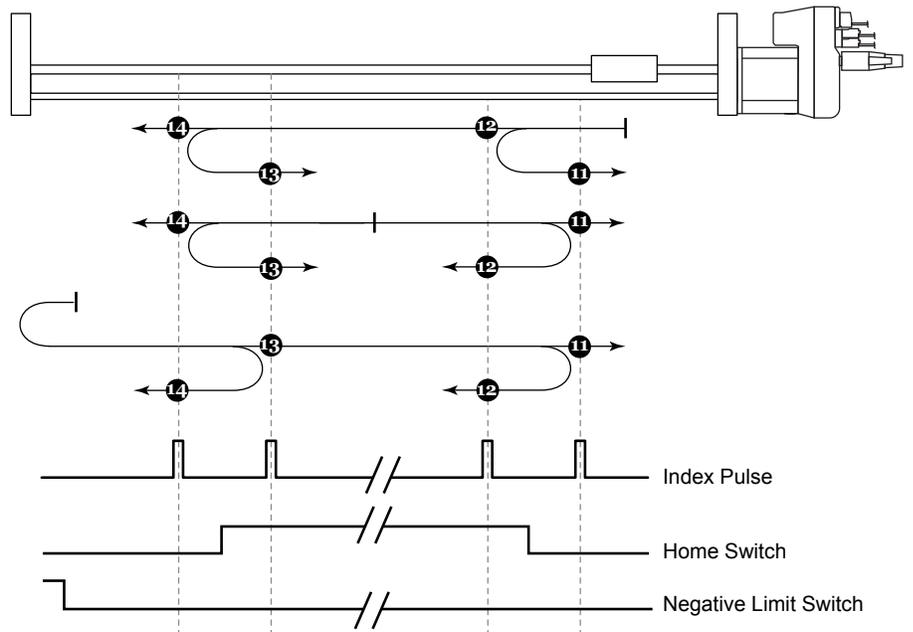
Figure 6.14: Homing on the home switch and index pulse - negative initial move

Methods 15 and 16: Reserved

These methods are reserved for future expansion of the homing mode.

Methods 17 to 30: Homing without an index pulse

These methods are similar to methods 1 to 14 except that the home position is not dependent on the index pulse but only Dependent on the relevant home or limit switch transitions. For example methods 19 and 20 are similar to methods 3 and 4 as shown in the following diagram.
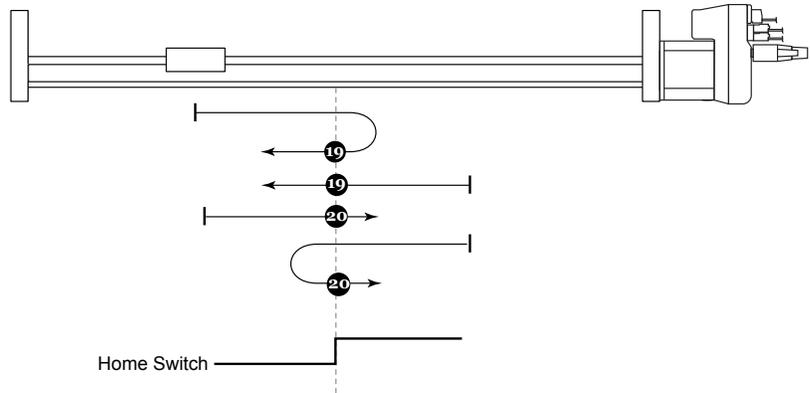
Figure 6.15: Homing without and index pulse

Methods 31 and 32: Reserved

These methods are reserved for future expansion of the homing mode.

Methods 33 and 34: Homing on an index pulse

Using methods 33 or 34 the direction of homing is negative or positive respectively. The home position is at the index pulse found in the selected direction.
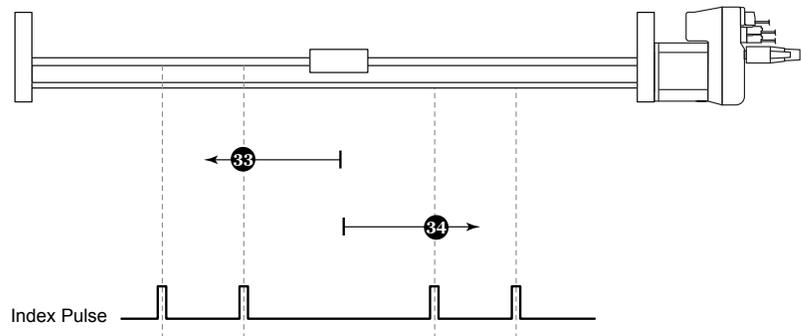
Figure 6.16: Homing on an index pulse

Method 35: Homing on the current position

In method 35 the current position is taken to be the home position.

6.7.5　6099$_h$ Homing speeds

The homing speeds object determines the fast and slow speeds that will be used during homing.

Object description

Index	6098$_h$
Name	Homing speeds
Object code	ARRAY
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Name	Number of entries
Access	ro
PDO mapping	—
Value range	—
Default value	02$_h$

Sub-index	01$_h$
Name	Homing speed fast
Access	rw
PDO mapping	—
Value range	0000 0000$_h$ to 7FFF FFFF$_h$
Default value	102400$_d$

Sub-index	02$_h$
Name	Homing speed slow
Access	rw
PDO mapping	—
Value range	0000 0000$_h$ to 7FFF FFFF$_h$
Default value	6400$_d$

6.7.6   Homing mode application example

Homing Mode – demonstrates home method 18 decimal using Service Data Objects (SDOs).

Devices represented by this manual support the ability to move in homing mode.

The below example sets typical motion profile commands a system would configure[1], enabling the motor power[2] and executing a homing function using SDOs with Node ID 41_h.

[1] Typical motion profile commands could be set each time on power up from host or set using a configuration file and stored to NVM once.

[2] Enabling the motor power only has to be done once on power up.

All values shown are hexadecimal.

ID	RTR	Data String	Action
Typical motion parameters			
0641	00	2F 04 22 00 50 00 00 00	Set run current to 80%
0641	00	23 84 60 00 40 42 0F 00	Set deceleration to 1M steps/sec^2
0641	00	23 83 60 00 40 42 0F 00	Set acceleration to 1M steps/sec^2
0641	00	23 81 60 00 00 D0 07 00	Set max velocity to 512000 steps/sec
Enable motor power - DSP402 state machine			
0641	00	2B 40 60 00 06 00 00 00	Ready to switch on
0641	00	2B 40 60 00 07 00 00 00	Switched on
0641	00	2B 40 60 00 0F 00 00 00	Operation enable
Set to homing mode			
0641	00	2F 60 60 00 06 00 00 00	Set to homing mode
Configure I/O and homing method			
0641	00	22 00 20 01 00 00 00 00	Set I/O as inputs
0641	00	22 00 20 02 00 00 00 00	Set I/O as sinking
0641	00	22 00 20 04 01 00 00 00	Set I1 as polarity
0641	00	22 02 20 01 01 00 00 00	Set I1 as home switch
0641	00	22 06 20 01 0A 00 00 00	Set I1 filter to 10ms
Set homing method, offset and speeds			
0641	00	22 98 60 00 13 00 00 00	Homing Method 19 decimal
0641	00	2F 98 20 00 01 00 00 00	Apply home offset to pos counter
0641	00	22 7C 60 00 00 00 00 00	Home offset = 0
0641	00	22 99 60 01 00 C8 00 00	Home speed fast
0641	00	22 99 60 02 00 14 00 00	Home speed slow
Start homing			
0641	00	2B 40 60 00 1F 00 00 00	Start homing
After home switch toggles			
0641	00	2B 40 60 00 00 00 00 00	Stop homing

Table 6.23: Homing mode application example

6.8   Torque mode (Closed loop models only)

6.8.1   Overview

The profile torque mode allows the Lexium MDrive to transmit the target torque value, which is processed via the trajectory generator and the hM-Technology logic.

The axis will ramp to the `TARGET_TORQUE (6071h)` as specified by the `TORQUE_RAMP (6087h)`.

The manufacturer-specific `TORQUE_SPEED (2704h)` specifies the axis velocity at which the `TARGET_TORQUE` will be maintained.
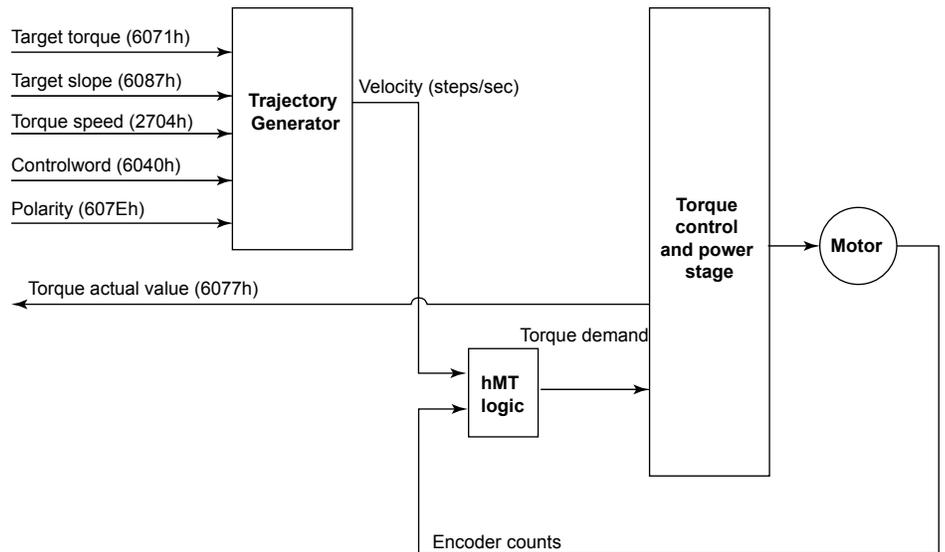
Figure 6:17: Profile torque mode

6.8.2   Control word definition for profile torque mode

15 ... 9	8	7	6	4	3	0
See 6.2.1	halt	See 6.2.1	reserved		See 6.2.1	

Bit	Value	Meaning
8	0	The motion shall be executed 8 or continued
	1	1 Axis shall be stopped according to the halt option code (605Dh)

Table 6.24: Profile torque mode control word (6040h) bit state meanings

6.8.2 Control word definition for profile torque mode

15	14	13	12	11	10	9	0
See 6.2.1		reserved		See 6.2.1	Target reached	See 6.2.1	

Bit	Value	Meaning
10	0	Halt (Bit 8 in controlword) = 0: Target torque not reached
		Halt (Bit 8 in controlword) = 1: Axis decelerates
	1	Halt (Bit 8 in controlword) = 0: Target torque reached
		Halt (Bit 8 in controlword) = 1: Velocity of axis is 0

Table 6.25: Profile torque mode status word (6041h) bit state meanings

Torque Mode objects

6.8.3 Object 6071$_h$: Target torque

This object shall indicate the configured input value for the torque controller in profile torque mode. The value shall be given per thousand of rated torque.

Object description

Index	6071$_h$
Name	Target torque
Object code	VAR
Data type	Integer16
Category	Mandatory

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	-1000$_d$ to 1000$_d$
Default value	0$_d$

6.8.4 Object 6077$_h$: Torque actual value

Provides the actual value of the torque. It shall correspond to the instantaneous torque in the motor. The value shall be given per thousand of rated torque.

Object description

Index	6077$_h$
Name	Torque actual value
Object code	VAR
Data type	Integer16
Category	Optional

Entry description

Sub-index	00$_h$
Access	ro
PDO mapping	Yes
Value range	-1000$_d$ to 1000$_d$
Default value	0$_d$

6.8.5 Object 6087$_h$: Torque slope

This object shall indicate the configured rate of change of torque. The value shall be given in units of per thousand of rated torque per second.

Object description

Index	6087$_h$
Name	Torque slope
Object code	VAR
Data type	Unsigned32
Category	Mandatory

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	Yes
Value range	Unsigned32
Default value	1000$_d$

6.8.6 Profile torque example

This example will enter profile torque, set the target torque, speed and torque ramp.

ID	RTR	Data String	Action
0641	00	2F 60 60 00 04 00 00 00	Enable profile torque mode
0641	00	23 87 60 00 E8 03 00 00	Set torque ramp to 100%/sec
0641	00	2F 04 27 00 62 00 00 00	Set max torque speed to 120000 usteps/sec
0641	00	2B 71 60 00 F4 01 00 00	Set torque to 50%

Table 6.25: Profile torque application example

6.9 Cyclic synchronous position

6.9.1 Overview

With this mode, the trajectory generator is located in the control device, not in the drive device. In cyclic synchronous manner it provides a target position to the drive device, which performs position control and velocity control. Optionally, additive velocity values can be provided by the control system in order to allow for velocity feed-forward.

The behavior of the control function is influenced by control parameters, which are externally applicable. The interpolation time period defines the time period between two updates of the target position and/or additive position and shall be used for inter-cycle interpolation.

The target position is interpreted as absolute value. The position actual value is used as mandatory output to the control device. Additionally, the velocity actual value is output.

The following error is used to set an error bit in the Status word (6041h).
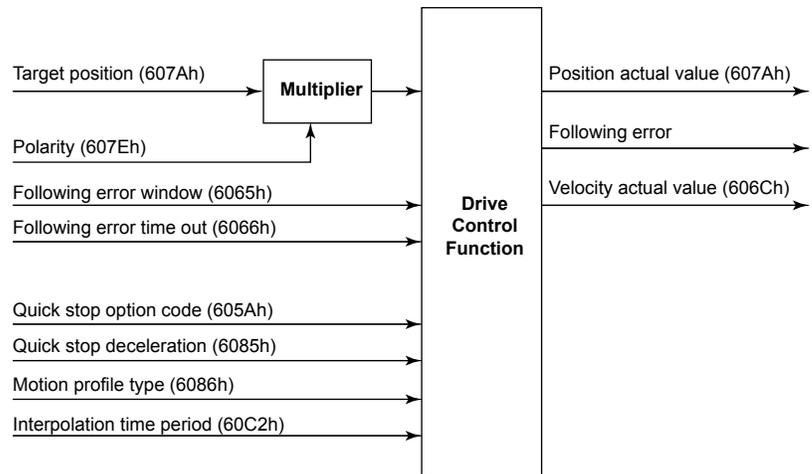
Figure 6.18: Cyclic synchronous position control function

6.9.2   Controlword and statusword for cyclic synch position

The cyclic synchronous position mode uses no mode specific bits of the controlword and three bits of the statusword for mode-specific purposes.

15 ... 14	13	12	11	10	9 0
See 6.2.1	following error	See 6.2.1	Target position ignored	reserved	See 6.2.1

Bit	Value	Meaning
10	0	reserved
	1	reserved
11	0	Target position ignored
	1	Target position shall be used as input to position control loop
13	0	No following error
	1	Following error

Table 6.26: Statusword bits for Cyclic Synchronous Position mode

6.10   Position control function

6.10.1   Overview

In this section, all parameters are described which are necessary for a closed loop position control. The control loop is fed with the `position_de-mand_value` as one of the outputs of the Trajectory Generator and with the output of the position detection unit (`position_actual_value`) like a resolver or encoder as input parameters.

6.10.2   6062$_h$ Position demand actual value

This object shall provide the demanded position value. The value shall be given in motor steps.

Object description

Index	6062$_h$
Name	Position demand actual value
Object code	VAR
Data type	Integer32
Category	Optional

Entry description

Sub-index	00$_h$
Access	ro
PDO mapping	Yes
Value range	Integer32
Default value	0000 0000$_h$

6.10.3   6063$_h$ Position actual value internal

This object shall provide the actual value of the position measurement device, which shall be one of the two input values of the closed-loop position control.

Object description

Index	6063$_h$
Name	Position actual value internal
Object code	VAR
Data type	Integer32
Category	Optional

Entry description

Sub-index	00$_h$
Access	ro
PDO mapping	Yes
Value range	Integer32
Default value	0000 0000$_h$

6.10.4   6064$_h$ Position actual value

This object represents the actual value of the position measurement device microsteps.

Object description

Index	6064$_h$
Name	Position actual value
Object code	VAR
Data type	Integer32
Category	Optional

Entry description

Sub-index	00$_h$
Access	ro
PDO mapping	Yes
Value range	Integer32
Default value	0000 0000$_h$

6.10.5   6065$_h$ Following error window

This object shall indicate the configured range of tolerated position values symmetrically to the position demand value. If the position actual value is out of the following error window, a following error occurs. A following error may occur when a drive is blocked, unreachable profile velocity occurs, or at wrong closed-loop coefficients. The value shall be given in user defined position units. If the value of the following error window is FFFF FFFF$_h$, the following control shall be switched off.

Object description

Index	6065$_h$
Name	Following error window
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	No
Value range	Unsigned32
Default value	0000 0512$_h$

6.10.6   6066$_h$ Following error timeout

This object shall indicate the configured time for a following error condition, after that the bit 13 of the statusword shall be set to 1. The reaction of the drive when a following error occurs is manufacturer-specific. The value shall be given in milliseconds.

Object description

Index	6066$_h$
Name	Following error timeout
Object code	VAR
Data type	Unsigned16
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	No
Value range	Unsigned16
Default value	0000$_h$

6.10.7   6067$_h$ Position window

This object shall indicate the configured symmetrical range of accepted positions relatively to the target position. If the actual value of the position encoder is within the position window, this target position shall be regarded as reached. The target position shall be handled in the same manner as in the trajectory generator concerning limiting functions and transformation into internal machine units before it may be used with this function. The value shall be given in user-defined position units. If the value of the position window is FFFF FFFF$_h$, the position window control shall be switched off.

Object description

Index	6067$_h$
Name	Position window
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	rw
PDO mapping	No
Value range	Unsigned32
Default value	0FFF FFFF$_h$

6.10.8   6068ₕ Position window time

This object shall indicate the configured time, during which the actual position within the position window is measured. The value shall be given in milliseconds.

Object description

Index	6066ₕ
Name	Position window time
Object code	VAR
Data type	Unsigned16
Category	Optional

Entry description

Sub-index	00ₕ
Access	rw
PDO mapping	No
Value range	Unsigned16
Default value	0000ₕ

6.11 Factors

6.11.1 608F_h Position encoder resolution

This object defines the ratio of encoder increments per motor revolution:

$$position\ encoder\ resolution = \frac{encoder\ increments\ (sub\ 01_h)}{motor\ revolutions\ (sub\ 02_h)}$$

The default values assume a 512 line (2048 edges) encoder.

Object description

Index	608F_h
Name	Position encoder resolution
Object code	ARRAY
Data type	—
Category	Optional

Entry description

Sub-index	00_h
Name	Number of entries
Access	ro
PDO mapping	—
Value range	—
Default value	02_h

Sub-index	01_h
Name	Resolution numerator
Access	rw
PDO mapping	—
Value range	Unsigned32 (1 to 65535_d)
Default value	2048_d

Sub-index	02_h
Name	Resolution denominator
Access	rw
PDO mapping	—
Value range	Unsigned32 (1 to 65535_d)
Default value	1_d

6.11.2   6092$_h$ Feed and drive shaft resolution

This object defines the ratio of user units per shaft revolution:

$$feed\ and\ drive\ shaft\ resolution = \frac{feed\ user\ units\ (sub\ 01_h)}{shaft\ revolutions\ (sub\ 02_h)}$$

The default values assume a 200 step/rev motor at a step resolution of 256, or 51200 steps per rev as the factor.

Object description

Index	6092$_h$
Name	Feed and drive shaft resolution
Object code	ARRAY
Data type	—
Category	Optional

Entry description

Sub-index	00$_h$
Name	Number of entries
Access	ro
PDO mapping	—
Value range	—
Default value	02$_h$

Sub-index	01$_h$
Name	Feed user units numerator
Access	rw
PDO mapping	—
Value range	Unsigned32
Default value	51200$_d$

Sub-index	02$_h$
Name	Shaft revolutions denominator
Access	rw
PDO mapping	—
Value range	Unsigned32
Default value	1$_d$

6.12   Optional application FE (general I/O)

6.12.1   60FD$_h$ Digital inputs

This object reads the digital inputs.

31 ... 24	23	22	21	20	19	18	17	16	15 ... 4	3	2	1	0
X	X	X	X	X	4	3	2	1	reserved	X	home switch	+ limit	– limit

MSB   Input points   LSb

Object description

Index	60FD$_h$
Name	Digital inputs
Object code	VAR
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Access	ro
PDO mapping	Yes
Value range	Unsigned32
Default value	0000 0000$_h$

6.12.1 60FE_h Digital outputs

This object reads the digital inputs.

31 ... 24	23	22	21	20	19	18	17	16	15 ... 1	0
X	X	X	X	X	X	3	2	1	reserved	brake
MSB	Output points									LSb

Object description

Index	60FE$_h$
Name	Digital outputs
Object code	ARRAY
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Name	Number of entries
Access	ro
PDO mapping	No
Value range	01$_h$
Default value	01$_h$

Sub-index	01$_h$
Name	Digital outputs
Access	rw
PDO mapping	Yes
Value range	Unsigned32
Default value	0000 0000$_h$

Page intentionally left blank

7 Diagnostics and Troubleshooting

7.1 Fieldbus communication error diagnostics

A properly operating fieldbus is essential for evaluating operating and error messages.

Connections for fieldbus mode   If the product cannot be addressed via the fieldbus, first check the connections. The product manual contains the technical data of the device and information on network and device installation. Check the following:

- Power connections to the device
- Fieldbus cable and fieldbus wiring

Baud rate and address   If it is impossible to connect to a device, check the baud rate and node address.

- The baud rate must be the same for all devices in the network.
- The node address of each device must be between 1 and 127 and unique for each device.

To set the baud rate and node address see Section 5: Commissioning.

Fieldbus function test   After correct configuration of the transmission data, test fieldbus mode. This requires installation of a CAN configuration tool that displays CAN messages. Feedback from the product is indicated in the form of a bootup message:

- Switch the power supply off and on again.
- Observe the network messages after switching on. After initialization of the bus, the device sends a boot-up message (COB ID 700h + node ID and 1 data byte with the content 00h).
- With the factory setting 65 (41h) for the node address, the boot-up message is sent via the bus . The device can then be put into operation via NMT services.

7.2   Error diagnostics via fieldbus

7.2.1   Message objects

A number of objects provide information on the operating state and on errors:

- Object `Statusword (6041h)` Operating states, see Section 6.2.2.
- Object `EMCY (80h+ Node-ID)` Error message from a device with error and error code.
- Object `Error register (1001h)` Error
- Object `Error code (603Fh)` Error code of the most recent error. See Section 6.3.2.

7.2.2   Messages on device status

Synchronous and asynchronous errors are distinguished in terms of evaluation and handling of errors.

Synchronous errors   The device signals a synchronous error directly as a response to a message that cannot be evaluated. Possible causes comprise transmission errors or invalid data. See Section 7.4.1 "Error register" for a list of synchronous errors.

Asynchronous errors   Asynchronous errors are signaled by the monitoring units in the device as soon as a device error occurs. An asynchronous error is signaled via bit 3, Fault, of the object `statusword (6041h).` In the case of errors that cause an interruption of the movement, the device transmits an EMCY message.

7.3   CANopen error messages

CANopen error messages are signaled in the form of EMCY messages. They are evaluated via the objects `Error register (1001h)` and `Error code (603Fh).`

CANopen signals errors that occur during data exchange via SDO with the special SDO error message ABORT.

7.3.1 Error register (1001$_h$)

This object is an error register for the device. The device can map internal errors in this byte. This entry is mandatory for all devices. It is a part of an Emergency object.

Object description

Index	1001$_h$
Name	Error register
Object code	VAR
Data type	Unsigned8
Category	Mandatory

Entry description

Sub-index	00$_h$
Access	ro
PDO mapping	Optional
Value range	Unsigned8
Default value	—

Bit	M/O	Meaning
0	M	Generic Error
1	O	Current
2	O	Voltage
3	O	Temperature
4	O	Communication error (Overrun, Error State)
5	O	Device profile specific
6	O	Reserved (always 0)
7	O	Manufacturer specific

Table 7.1: Abort Connection Option Code

7.3.2 Pre-defined error (1003$_h$)

The object at index 1003h holds the errors that have occurred on the device and have been signaled via the Emergency Object. In doing so it provides an error history.

1) The entry at sub-index 0 contains the number of actual errors that are recorded in the array starting at sub-index 1.

2) Every new error is stored at sub-index 1, the older ones move down the list.

3) Writing a „0" to sub-index 0 deletes the entire error history (empties the array). Values higher than 0 are not allowed to write. This have to lead to an abort message (error code: 0609 0030h).

4) The error numbers are of type UNSIGNED32 and are composed of a 16 bit error code and a 16 bit additional error information field which is manufacturer specific. The error code is contained in the lower 2 bytes (LSB) and the additional information is included in the upper 2 bytes (MSB). If this object is supported it must consist of two entries at least. The length entry on subindex 0h and at least one error entry at sub-index 1H.

Object description

Index	1003$_h$
Name	Pre-defined error field
Object code	ARRAY
Data type	Unsigned32
Category	Optional

Entry description

Sub-index	00$_h$
Description	Number of errors
Access	rw
Entry category	Mandatory
PDO mapping	No
Value range	0 – 254
Default value	0

Sub-index	01$_h$
Description	Standard error field
Access	ro
Entry category	Optional
PDO mapping	No
Value range	Unsigned32
Default value	—

Sub-index	02$_h$ – FE$_h$
Description	Standard error field
Access	ro
Entry category	Optional
PDO mapping	No
Value range	Unsigned32
Default value	—

Error description	Add'tl info byte	Error code byte
No error	0$_h$	0000$_h$
CAN overrun	0$_h$	8110$_h$
Can in error passive mode	0$_h$	8120$_h$
Lifeguard or heartbeat error	0$_h$	8130$_h$
Recovered from "bus off" state	0$_h$	8140$_h$
Bus off state occurred	0$_h$	8141$_h$
PDO not processed - length error	0$_h$	8210$_h$
Over temperature error	8$_h$	4210$_h$
Pending over temperature warning	16$_h$	4210$_h$
Motor idle during commanded move	1$_h$	FF01$_h$
Motor should be idle	2 – 8$_h$	FF01$_h$
Undershot warning	9$_h$	FF01$_h$

Table 7.2: Description of the error codes

7.4   Status LED

he Lexium MDrive has two LEDs for status indication.

- LED 1: Status of the power supply
- LED 2: The CANopen status LED shows the states as specified in CiA DR-303-3 Indicator specification.

LED 1 power indication

Color	Status
Off	No Power
Green	+VDC supply in range
Flashing green	+VDC off, drive on AUX power
Red	+VDC supply out of range
Flashing red	+VDC off, AUX power out of range

LED 2 CANopen status

Color	Status
Red - single Flash	At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames).
Red - double flash	A guard event (NMT-Slave or NMT-master) or a heartbeat event (Heartbeat consumer) has occurred.
Red - triple flash	The SYNC message has not been received within the configured communication cycle period time out (see Object Dictionary Entry 0x1006).
Red - on	The CAN controller is bus off
Green - single flash	The Device is in STOPPED state
Green - blinking	The Device is in the PREOPERATIONAL state
Green - on	The Device is in the OPERATIONAL state

Page intentionally left blank

8   Object Dictionary

8.1   Specification for the objects

Index The index specifies the position of the object in the object dictionary. The index value is specified as a hexadecimal value.

Object code & data types The object code specifies the data structure of the object.

Object code	Meaning	Coding
VAR	A simple value, for example of the type Integer8 or Unsigned32	7
ARR (ARRAY)	A data field in which the entries have the same data type.	8
REC (RECORD)	A data field that contains entries that are a combination of simple data types.	9

Table 8.1: CANopen object codes

Data type	Value range	Data length	DS301 coding
Boolean	0 = false, 1 = true	1 byte	0001
Integer8	–128 ... +127	1 byte	0002
Integer16	–32768 ... +32767	2 byte	0003
Integer32	–2147483648 ... 2147483647	4 byte	0004
Unsigned8	0 ... 255	1 byte	0005
Unsigned16	0 ... 65535	2 byte	0006
Unsigned32	0 ... 4294967295	4 byte	0007
Visible String8	ASCII chars	8 byte	0009
Visible String168	ASCII chars	16 byte	0010

Table 8.2: CANopen data types

RO/RW Indicates read and/or write values
RO: values can only be read
RW: values can be read and written.

PDO R_PDO: Mapping for R_PDO possible
T_PDO: Mapping for T_PDO possible
No specification: PDO mapping not possible with the object

Min/max values Specifies the permissible range in which the object value is defined and valid.

Factory default Factory default settings when the product is shipped

8.2 Overview of object group 1000_h

No objects from object group 1000_h are PDO mappable.

Index	Sub-index	Name	Obj. code	Data type	Access	Description
1000_h		Device type	VAR	Unsigned32	ro	Device type and profile
1001_h		Error register	VAR	Unsigned8	ro	Error register
1003_h		Predefined error field	ARR		rw	Error history, memory for error messages
	00_h	Number of errors	VAR	Unsigned8	rw	Number of error entries
	01_h - 04_h	Error field	VAR	Unsigned32	ro	Error number
1005_h		COB-ID SYNC	VAR	Unsigned32	rw	Identifier of the synchronization object
1007_h		Sync window length	VAR	Unsigned32	rw	Time window for synchronous PDOs in µS
1008_h		Mfg. device name	VAR	Vis String8	ro	Manufacturer's designation
1009_h		Mfg. hardware version	VAR	Vis String8	ro	Hardware version
100A_h		Mfg. software version	VAR	Vis String8	ro	Software version
100C_h		Guard time	VAR	Unsigned16	rw	Time span for Node Guarding [ms]
100D_h		Life time factor	VAR	Unsigned8	rw	Repeat factor for Node Guarding
1010_h		Store parameters	ARR	Unsigned32		Store parameters
	00_h	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 04_h
	01_h	Save all parameters	VAR	Unsigned32	rw	Save all parameters
	02_h	Save communication	VAR	Unsigned32	rw	Save Communication Parameters
	03_h	Save application	VAR	Unsigned32	rw	Save Application Parameters
	04_h	Save manufacturer	VAR	Unsigned32	rw	Save Manufacturer Parameters
1011_h		Restore defaults	ARR	Unsigned32		Restore defaults as group
	00_h	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 04_h
	01_h	Restore all defaults	VAR	Unsigned32	rw	Restore all defaults
	02_h	Restore communication	VAR	Unsigned32	rw	Restore Communication defaults
	03_h	Restore application	VAR	Unsigned32	rw	Restore Application defaults
	04_h	Restore manufacturer	VAR	Unsigned32	rw	Restore Manufacturer defaults
1012_h		COB-ID time stamp	VAR	Unsigned32	rw	COB-ID time stamp message
1014_h		COB-ID EMCY	VAR	Unsigned32	rw	80_h + Node ID
1015_h		Inhibit time EMCY	VAR	Unsigned16	rw	Wait time for the repeated transmission of EMCY x 100 µS
1017_h		Producer Heartbeat Time	VAR	Unsigned16	rw	Time interval for producer «Heartbeat»
1018_h		Identity	REC		ro	Identification object
	00_h	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 04_h
	01_h	Vendor ID	VAR	Unsigned32	ro	Vendor ID
	02_h	Product code	VAR	Unsigned32	ro	Product code
	03_h	Revision number	VAR	Unsigned32	ro	Revision number
	04_h	Serial number	VAR	Unsigned32	ro	Serial number
1400_h		1st R_PDO parameter	REC			1st receive PDO parameter
	00_h	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 05_h
	01_h	COB-ID used	VAR	Unsigned32	rw	COB-ID used: 200_h + Node ID
	02_h	Transmission type	VAR	Unsigned8	rw	Default type = 255 (asynchronous)
	03_h	Inhibit time	VAR	Unsigned16	rw	Default = 0
	05_h	Event timer	VAR	Unsigned16	rw	Default = 0

Index	Sub-index	Name	Obj. code	Data type	Access	Description
1401$_h$		2nd R_PDO parameter	REC			2nd receive PDO parameter
	00$_h$	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 05$_h$
	01$_h$	COB-ID used	VAR	Unsigned32	rw	COB-ID used: 300$_h$ + Node ID
	02$_h$	Transmission type	VAR	Unsigned8	rw	Default type = 255 (asynchronous)
	03$_h$	Inhibit time	VAR	Unsigned16	rw	Default = 0
	05$_h$	Event timer	VAR	Unsigned16	rw	Default = 0
1402$_h$		3rd R_PDO parameter	REC			3rd receive PDO parameter
	00$_h$	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 05$_h$
	01$_h$	COB-ID used	VAR	Unsigned32	rw	COB-ID used: 400$_h$ + Node ID
	02$_h$	Transmission type	VAR	Unsigned8	rw	Default type = 255 (asynchronous)
	03$_h$	Inhibit time	VAR	Unsigned16	rw	Default = 0
	05$_h$	Event timer	VAR	Unsigned16	rw	Default = 0
1403$_h$		4th R_PDO parameter	REC			4th receive PDO parameter
	00$_h$	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 05$_h$
	01$_h$	COB-ID used	VAR	Unsigned32	rw	COB-ID used: 500$_h$ + Node ID
	02$_h$	Transmission type	VAR	Unsigned8	rw	Default type = 255 (asynchronous)
	03$_h$	Inhibit time	VAR	Unsigned16	rw	Default = 0
	05$_h$	Event timer	VAR	Unsigned16	rw	Default = 0
1600$_h$		1st R_PDO mapping	REC		ro	PDO mapping for R_PDO1, settings
	00$_h$	# of mapped objects	VAR	Unsigned8	rw	Number of mapped objects, range 1 – 64
	01$_h$ - 08$_h$	Application Objects	VAR	Unsigned32	rw	R_PDO1 mapping application objects
1601$_h$		2nd R_PDO mapping	REC		ro	PDO mapping for R_PDO2, settings
	00$_h$	# of mapped objects	VAR	Unsigned8	rw	Number of mapped objects, range 1 – 64
	01$_h$ - 08$_h$	Application Objects	VAR	Unsigned32	rw	R_PDO2 mapping application objects
1602$_h$		3rd R_PDO mapping	REC		ro	PDO mapping for R_PDO3, settings
	00$_h$	# of mapped objects	VAR	Unsigned8	rw	Number of mapped objects, range 1 – 64
	01$_h$ - 08$_h$	Application Objects	VAR	Unsigned32	rw	R_PDO3 mapping application objects
1603$_h$		4th R_PDO mapping	REC		ro	PDO mapping for R_PDO4, settings
	00$_h$	# of mapped objects	VAR	Unsigned8	rw	Number of mapped objects, range 1 – 64
	01$_h$ - 08$_h$	Application Objects	VAR	Unsigned32	rw	R_PDO4 mapping application objects
1800$_h$		1st T_PDO parameter	REC			1st transmit PDO parameter
	00$_h$	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 05$_h$
	01$_h$	COB-ID used	VAR	Unsigned32	rw	COB-ID used: 180$_h$ + Node ID
	02$_h$	Transmission type	VAR	Unsigned8	rw	Default type = 255 (asynchronous)
	03$_h$	Inhibit time	VAR	Unsigned16	rw	Default = 0
	05$_h$	Event timer	VAR	Unsigned16	rw	Default = 0
1801$_h$		2nd T_PDO parameter	REC			2nd transmit PDO parameter
	00$_h$	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 05$_h$
	01$_h$	COB-ID used	VAR	Unsigned32	rw	COB-ID used: 280$_h$ + Node ID
	02$_h$	Transmission type	VAR	Unsigned8	rw	Default type = 255 (asynchronous)
	03$_h$	Inhibit time	VAR	Unsigned16	rw	Default = 0
	05$_h$	Event timer	VAR	Unsigned16	rw	Default = 0

Index	Sub-index	Name	Obj. code	Data type	Access	Description
1802_h		3rd T_PDO parameter	REC			3rd transmit PDO parameter
	00_h	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 05_h
	01_h	COB-ID used	VAR	Unsigned32	rw	COB-ID used: 380_h + Node ID
	02_h	Transmission type	VAR	Unsigned8	rw	Default type = 255 (asynchronous)
	03_h	Inhibit time	VAR	Unsigned16	rw	Default = 0
	05_h	Event timer	VAR	Unsigned16	rw	Default = 0
1803_h		4th T_PDO parameter	REC			4th transmit PDO parameter
	00_h	Largest sub-index	VAR	Unsigned8	ro	Largest sub-index supported» 05_h
	01_h	COB-ID used	VAR	Unsigned32	rw	COB-ID used: 480_h + Node ID
	02_h	Transmission type	VAR	Unsigned8	rw	Default type = 255 (asynchronous)
	03_h	Inhibit time	VAR	Unsigned16	rw	Default = 0
	05_h	Event timer	VAR	Unsigned16	rw	Default = 0
$1A00_h$		1st T_PDO mapping	REC		ro	PDO mapping for T_PDO1, settings
	00_h	# of mapped objects	VAR	Unsigned8	rw	Number of mapped objects, range 1 – 64
	01_h - 08_h	Application Objects	VAR	Unsigned32	rw	T_PDO1 mapping application objects
$1A01_h$		2nd T_PDO mapping	REC		ro	PDO mapping for T_PDO2, settings
	00_h	# of mapped objects	VAR	Unsigned8	rw	Number of mapped objects, range 1 – 64
	01_h - 08_h	Application Objects	VAR	Unsigned32	rw	T_PDO2 mapping application objects
$1A02_h$		3rd T_PDO mapping	REC		ro	PDO mapping for T_PDO3, settings
	00_h	# of mapped objects	VAR	Unsigned8	rw	Number of mapped objects, range 1 – 64
	01_h - 08_h	Application Objects	VAR	Unsigned32	rw	T_PDO3 mapping application objects
$1A03_h$		4th T_PDO mapping	REC		ro	PDO mapping for T_PDO4, settings
	00_h	# of mapped objects	VAR	Unsigned8	rw	Number of mapped objects, range 1 – 64
	01_h - 08_h	Application Objects	VAR	Unsigned32	rw	T_PDO4 mapping application objects

8.3 Overview of manufacturer specific objects group 2000$_h$

Index	Sub	Name	Obj. code	Data type	Access	PDO	Description
2000$_h$		Configure GPIO	ARR				Configure the general purpose I/O points
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 05$_h$
	01$_h$	Reserved	VAR	Unsigned8	rw		Configure the I/O point as an output or input
	02$_h$	Reserved	VAR	Unsigned8	rw		Configure the I/O point as sourcing or sinking
	03$_h$	Reserved	VAR	Unsigned8	rw		Configure as both sink/source or source only
	04$_h$	Configure polarity in	VAR	Unsigned8	rw		Configure input logic polarity
	05$_h$	Reserved	VAR	Unsigned8	rw		Configure output logic polarity
2002$_h$		Configure digital inputs	ARR				Configure the functions of inputs
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 04$_h$
	01$_h$	Define as Home switch	VAR	Unsigned8	rw		Configure input as homing switch
	02$_h$	Define as positive limit	VAR	Unsigned8	rw		Configure input as positive limit switch
	03$_h$	Define as negative limit	VAR	Unsigned8	rw		Configure input as negative limit switch
	04$_h$	Define as inhibit switch	VAR	Unsigned8	rw		Configure input as inhibit switch (see 2007$_h$)
2004$_h$		Input mask	ARR				Configure input filter mask
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 08$_h$
	01$_h$ - 08$_h$	Input filter mask	VAR	Unsigned8	rw		Defines inputs to apply filtering (see 2006$_h$)
2006$_h$		Input filter time	ARR				Configure input filter time
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 08$_h$
	01$_h$ - 08$_h$	Input filter time	VAR	Unsigned8	rw		Defines inputs filter time in ms
2007$_h$		Inhibit switch	ARR				Configure inhibit switch
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 01$_h$
	01$_h$	Input switch action	VAR	Unsigned8	rw		Defines the action of the inhibit switch
2008$_h$		Configure digital outputs	ARR				Configure output functions
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 02$_h$
	01$_h$	Brake output defined	VAR	Unsigned8	rw		Defines the output(s) used for braking
	02$_h$	Target reached output	VAR	Unsigned8	rw		Defines the output used to indicate target reached
2010$_h$		Analog input configuration	ARR				Configure the analog input
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 03$_h$
	01$_h$	Analog input read value	VAR	Unsigned16	ro	T_PDO	Analog input value
	02$_h$	Analog input scale	VAR	Unsigned8	rw		Sets the mode as 0 - 5V, 0 - 10V or 0 - 20mA
	03$_h$	Analog input filter	VAR	Unsigned8	rw		Defines the filtering for the analog input
2014$_h$		Aux-power monitoring	ARR				Monitor the level of the Aux-power input
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 03$_h$
	01$_h$	Aux-power voltage	VAR	Unsigned8	ro	T_PDO	Read Aux-power voltage value
	02$_h$	Aux-power low warning	VAR	Unsigned8	ro		Read Aux-power low level warning
	03$_h$	Aux-power high warning	VAR	Unsigned8	ro		Read Aux-power high level warning
2015$_h$		Input voltage monitoring	ARR				Monitors the level of the +VDC input voltage
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 03$_h$
	01$_h$	Input +V	VAR	Unsigned8	ro	T_PDO	Read Input +V voltage value
	02$_h$	Input +V low warning	VAR	Unsigned8	ro		Read Input +V low level warning
	03$_h$	Input +V high warning	VAR	Unsigned8	ro		Read Input +V high level warning

Index	Sub	Name	Obj. code	Data type	Access	PDO	Description
2018$_h$		Board Temperature	ARR				Set the board temperature parameters
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 03$_h$
	01$_h$	Board temperature	VAR	Integer8	ro	T_PDO	Board temperature reading
	02$_h$	Set temperature warning	VAR	Integer8	rw		Set board temperature warning threshold
	03$_h$	Set temperature fault	VAR	Integer8	rw		Set board temperature fault threshold
2019$_h$		Bridge Temperature	ARR				Set the bridge temperature parameters
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 03$_h$
	01$_h$	Bridge temperature	VAR	Integer8	ro	T_PDO	Bridge temperature reading
	02$_h$	Set temperature warning	VAR	Integer8	rw		Set bridge temperature warning threshold
	03$_h$	Set temperature fault	VAR	Integer8	rw		Set board temperature fault threshold
2020$_h$		Set temperature fault	VAR	Integer8	rw		Set bridge temperature fault threshold
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 02$_h$
	01$_h$	Limit reached flag	VAR	Unsigned8	rw		Indicates type of limit reached
	02$_h$	Limit reached mask	VAR	Unsigned8	rw		Defines limits to act upon
2022$_h$		Software limit	ARR				Actual position softwre limit
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 02$_h$
	01$_h$	Actual negative limit	VAR	Interger32	rw		Actual negative limit
	02$_h$	Actual positive limit	VAR	Interger32	rw		Actual positive limit
2030$_h$		Output bridge polarity	VAR	Integer8	rw		Defines the polarity of the output bridge
2031$_h$		Unit options	VAR	Unsigned8	rw		Enable encoder, capture/trip functions
2033$_h$		Capture input parameters	REC				Capture input parameters
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 04$_h$
	01$_h$	Capture input control	VAR	Unsigned8	rw		Enables the capture input
	02$_h$	Capture input flag	VAR	Unsigned8	rw		Displays the status of a position capture
	03$_h$	Capture input filter	VAR	Unsigned8	rw		Sets the filtering for the capture input
	04$_h$	Captured position	VAR	Integer32	ro	T_PDO	Stores the captured position
2034$_h$		Bridge on settle time	ARR				Settling time after bridge power on
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 01$_h$
	01$_h$	Bridge settle time	VAR	Unsigned8	rw		Bidge settling time in msec
2035$_h$		Brake settle allow time					Settling time after brake on/off
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 02$_h$
	01$_h$	Brake on settle time	VAR	Unsigned8	rw		Brake on settling time in msec
	02$_h$	Brake off settle time	VAR	Unsigned8	rw		Brake off settling time in msec
2036$_h$		Hold current delay time	VAR	Unsigned16	rw		Defines time in msec to transition to hold current following cessation of motion
2037$_h$		Bridge on to encoder settle time	VAR	Unsigned16	rw		Time between switching into operation enable to resynching the encoder position
2038$_h$		Trip output configuration	REC				Trip output parameters
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 03$_h$
	01$_h$	Trip output control	VAR	Unsigned16	rw		Controls logic and trip points
	02$_h$	1st position of a series	VAR	Integer32	rw		First trip position
	03$_h$	Multiple trip point spacing	VAR	Integer32	rw		Defines the spacing between subsequent trip points

Index	Sub	Name	Obj. code	Data type	Access	PDO	Description
2098_h		Homing configuration	VAR	Unsigned8	rw		Defines the counter status following a home
2204_h		Run current	VAR	Unsigned8	rw		Sets the motor run current percent
2205_h		Hold current	VAR	Unsigned8	rw		Sets the motor hold current percent
2211_h		Position present point target	VAR	Integer32	ro		Position present point target
2212_h		Position final point target	VAR	Integer32	ro		Position final point target
2221_h		Following Error	ARR				Following Error reaction code
	00_h	Number of entries	VAR	Unsigned8	ro		Number of entries = 01_h
	01_h	Reaction code	VAR	Unsigned8	rw		Following error reaction code
2401_h		Gen Purpose user variable	VAR	Unsigned8	rw		May be used to store 8 bits of data
2504_h		SEM options	VAR	Unsigned8	rw		SEM control options for compatibility
2701_h		hMTechnology enable	VAR	Unsigned8	rw		Enable/disable hMT control (closed loop only)
2702_h		Make up mode	VAR	Unsigned8	rw		Defines hMT make up mode (closed loop only)
2703_h		Make up velocity	VAR	Unsigned32	rw		Defines the velocity for make up (closed loop only)
2704_h		Torque Speed	VAR	Unsigned8	rw		Defines the velocity for torque mode (closed loop only)
2710_h		Locked Rotor Timeout	VAR	Unsigned8	rw		Defines the time period for a locked rotor condition to assert. (closed loop only)
2740_h		hMTechnology control	VAR	Unsigned8	rw	R_PDO	Defines the control parameters for hMTechnology. (closed loop only)
2741_h		hMTechnology status (filtered)	VAR	Unsigned8	ro	T_PDO	Reads the hMT status as filtered by 2743_h (closed loop only)
2742_h		hMTechnology status (unfiltered)	VAR	Unsigned8	ro		Reads the hMT status unfiltered (closed loop only)
2743_h		hMTechnology status filter	VAR	Unsigned8	rw		Determines which bits of the status byte are filtered in 2741_h

8.4   Overview of assignment objects group 6000$_h$

Index	Sub	Name	Obj. code	Data type	Access	PDO	Description
6007$_h$		Abort connection opcode	VAR	Integer16	rw		Controls the process for abort connection
603F$_h$		Error code	VAR	Unsigned16	ro	T_PDO	Stores the last error
6040$_h$		Control word	VAR	Unsigned16	rw	R_PDO	Control word
6041$_h$		Status word	VAR	Unsigned16	ro	T_PDO	Status word
605A$_h$		Quick stop option code	VAR	Integer16	rw		Defines the method for quick stop
605B$_h$		Shutdown option code	VAR	Integer16	rw		Defines the method for shutdown
605C$_h$		Disable operation opcode	VAR	Integer16	rw		Defines the method for disable operation
605D$_h$		Halt operation opcode	VAR	Integer16	rw		Defines the method for halt operation
605E$_h$		Fault reaction opcode	VAR	Integer16	rw		Defines the reaction to a fault
6060$_h$		Modes of operation	VAR	Integer8	rw	R_PDO	Set the mode of operation
6061$_h$		Modes of operation display	VAR	Integer8	ro	T_PDO	read the mode of operation
6062$_h$		Position demand value	VAR	Integer32	ro	T_PDO	Read the motor position in user units
6063$_h$		Position actual value	VAR	Integer32	ro	T_PDO	Read the motor position
6064$_h$		Position actual value	VAR	Integer32	ro	T_PDO	Read the motor position
6065$_h$		Following error window	VAR	Unsigned32	rw		Defines range of tolerated positions symetrical to 6062$_h$
6066$_h$		Following error window time	VAR	Unsigned16	rw		Defines the timeout for the next error window
6067$_h$		Position window	VAR	Unsigned32	rw		Defines accepted positions relative to target
6068$_h$		Position window timeout	VAR	Unsigned16	rw		Defines time to indicate target reached
606C$_h$		Velocity actual value	VAR	Integer32	ro	T_PDO	Actual velocity of the motor
6071$_h$		Target torque	VAR	Integer16	rw	R_PDO	Defines the target torque for Profile Torque mode
6077$_h$		Torque actual value	VAR	Integer16	ro	T_PDO	Actual torque value for Profile Torque
607A$_h$		Profiled target position	VAR	Integer32	rw	R_PDO	Defines target position for absolute or relative move
607C$_h$		Homing offset	VAR	Integer32	rw		Defines offset from homing zero position
607E$_h$		Polarity	VAR	Unsigned8	rw	R_PDO	Sets polarity for position/speed commands
6081$_h$		Profile velocity	VAR	Unsigned32	rw	R_PDO	Sets the velocity for the profile position motion
6082$_h$		End velocity	VAR	Unsigned32	rw	R_PDO	Sets the terminal (max) velocity
6083$_h$		Profile acceleration	VAR	Unsigned32	rw	R_PDO	Sets the acceleration for profile position and profile velocity motion
6084$_h$		Profile deceleration	VAR	Unsigned32	rw	R_PDO	Sets the deceleration for profile position and profile velocity motion
6085$_h$		Quick stop deceleration	VAR	Unsigned32	rw		Sets the deceleration for a quick stop state
6086$_h$		Motion profile type	VAR	Integer16	rw	R_PDO	Defines method by which profile motion is eva-luated
6087$_h$		Torque slope	VAR	Unsigned32	rw		Defines the torque ramp for Profile Torque mode
608F$_h$		Position encoder reso-lution	ARR				Defines relation between motor revolution and position increments
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 02$_h$
	01$_h$	Resolution numerator	VAR	Unsigned32	rw		# of encoder increments
	02$_h$	Resolution denominator	VAR	Unsigned32	rw		# of motor revolutions

Index	Sub	Name	Obj. code	Data type	Access	PDO	Description
6092$_h$		Factor group feed and driveshaft	ARR				Defines relation between feed user units and pdrive shaft revolutions
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 02$_h$
	01$_h$	Feed numerator	VAR	Unsigned32	rw		# of feed increments
	02$_h$	Driveshaft denominator	VAR	Unsigned32	rw		# of driveshaft revolutions
6098$_h$		Homing method	VAR	Integer8	rw		Defines the method for homing operation
6099$_h$		Homing speed	ARR				Defines the high and low speeds for homing
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 02$_h$
	01$_h$	Homing speed fast	VAR	Unsigned32	rw		Defines the high speed for homing
	02$_h$	Homing speed slow	VAR	Unsigned32	rw		Defines the low speed for homing
60C2$_h$		Interpolated position time period	REC				Defines time for interpolation position trajectory.
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 02$_h$
	01$_h$	Interpoltion period	VAR	Unsigned8	rw		Interpoltion time period
	02$_h$	Interpolation factor	VAR	Integer8	rw		Interpolation factor
60F8$_h$		Maximum slippage	VAR	Integer32	rw		Maximum slippage
60FD$_h$		Digital inputs	VAR	Unsigned32	ro	T_PDO	Reads the state of digital inputs
60FE$_h$		Digital outputs	ARR				Sets the state of digital outputs
	00$_h$	Number of entries	VAR	Unsigned8	ro		Number of entries = 01$_h$
	01$_h$	Digital outputs	VAR	Unsigned32	rw	R_PDO	Reads the state of digital inputs
60FF$_h$		Target velocity	VAR	Integer32	rw	R_PDO	Defines the target velocity
6402$_h$		Motor types	VAR	Unsigned16	ro		Motor type = 9: Stepper motor
6502$_h$		Supported drive modes	VAR	Unsigned32	ro		Profile position, profile velocity, homing

8.5 Details of object group 1000$_h$

8.5.1 1000$_h$ Device type

The object specifies the device profile used as well as the device type.

Object description

Index	1000$_h$
Name	Device type
Object code	VAR
Data type	Unsigned32

Value description

Sub-index	00$_h$, device type
Meaning	Device type and profile
Access	Read only
PDO mapping	—
Value range	—
Default value	0044 0192$_h$
Category	—

Bit coding sub-index 00$_h$

Bit	Access	Value	Meaning
31-16	ro	0044$_h$	Stepper motor
15-0	ro	0192$_h$	Device profile DSP402

8.5.2 1001$_h$ Error register

The object specifies the error of the device. The detailed cause of error can be determined with the object `predefined error field (1003h)` and - for reasons of compatibility with devices with other fieldbus profiles - with the object `error code (603Fh)`.

Errors are signaled by an EMCY message as soon as they occur.

Object description

Index	1001$_h$
Name	Error register
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, error register
Meaning	Error register
Access	Read only
PDO mapping	—
Value range	—
Default value	—
Category	—

Bit coding sub-index 00$_h$

Bit	Access	Value	Meaning
0	ro	—	Error (generic error)
1	ro	—	Reserved
2	ro	—	Reserved
3	ro	—	Temperature
4	ro	—	Communication profile (communication error)
5	ro	—	Reserved
6	ro	—	Reserved
7	ro	—	Manufacturer specific

8.5.3   1003$_h$ Pre-defined error field

The object contains the latest error messages that were shown as EMCY messages.

- The sub-index 00h entry contains the number of saved error messages.
- The current error message is stored at sub-index 01h, older messages are moved to higher sub-index entries.
- Writing 0 to sub-index 00h resets the error list.

Object description

Index	1003$_h$
Name	Pre-defined error field
Object code	ARRAY
Data type	Unsigned32

Value description

Sub-index	00$_h$, number of errors
Meaning	Number of error entries
Access	Read-write
PDO mapping	—
Value range	0 ... 4
Default value	0
Category	—

Sub-index	01$_h$ – 04$_h$, error field
Meaning	Error number
Access	Read only
PDO mapping	—
Value range	—
Default value	0
Category	—

Bit coding sub-index 00$_h$ ... 04$_h$ Bytes 0 ... 15: error code. Bytes 16 ... 31 additional error information, not assigned in the device.

8.5.4 1005$_h$ COB ID SYNC message

The object specifies the COB ID of the SYNC object and determines whether a device sends or receives SYNC messages.

The device can only receive SYNC messages.

For synchronization, a device in the network must send SYNC objects.

The COB ID can be changed in the NMT state "Pre-Operational"

Object description

Index	1005$_h$
Name	COB ID SYNC
Object code	VAR
Data type	Unsigned32

Value description

Sub-index	00$_h$, COB ID SYNC
Meaning	Identifier of the synchronization object
Access	Read-write
PDO mapping	—
Value range	0...4294967295
Default value	0000 0080$_h$
Category	Yes

Bit coding sub-index 00$_h$

Bit	Access	Value	Meaning
31	ro	0$_b$	
30	ro	0$_b$	
29	ro	0$_b$	
28-11	ro	0000$_h$	
10-7	rw	0001$_b$	
6-0	ro	7F$_h$	

8.5.5   1007$_h$ Sync window length

Contains the length of the time window for synchronous PDOs in microseconds.

Object description

Index	1007$_h$
Name	Sync window length
Object code	VAR
Data type	Unsigned32

Value description

Sub-index	00$_h$, Sync window length
Meaning	Timing for sync PDOs
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	0000 0000$_h$
Category	Yes

8.5.6   1008$_h$ Mfg. device name

Provides the name of the device as given by the manufacturer.

Object description

Index	1008$_h$
Name	Manufacturer device name
Object code	VAR
Data type	Visible String

Value description

Sub-index	00$_h$, Manufacturer device name
Meaning	Manufacturer device name
Access	Read only
PDO mapping	—
Value range	Visible String
Default value	CANopen Lexium MDrive Motion Control Node
Category	—

8.5.7   1009$_h$ Mfg. hardware version

Provides the hardware version of the device as given by the manufacturer.

Object description

Index	1009$_h$
Name	Manufacturer hardware version
Object code	VAR
Data type	Visible String

Value description

Sub-index	00$_h$, Manufacturer hardware version
Meaning	Manufacturer hardware version
Access	Read only
PDO mapping	—
Value range	Visible String
Default value	V1.00
Category	—

8.5.8   100A$_h$ Mfg. software version

Provides the software version of the device as given by the manufacturer.

Object description

Index	100A$_h$
Name	Manufacturer software version
Object code	VAR
Data type	Visible String

Value description

Sub-index	00$_h$, Manufacturer software version
Meaning	Manufacturer software version
Access	Read only
PDO mapping	—
Value range	Visible String
Default value	V5.48
Category	—

8.5.9   100C$_h$ Guard time

The object specifies the time span for connection monitoring (Node Guarding) of an NMT slave.

The time span for connection monitoring of an NMT master results from the time span "guard time" multiplied by the factor "life time", object `Life time factor(100Dh)`.

The time span can be changed in the NMT state "Pre-Operational".

Object description

Index	100C$_h$
Name	Guard time
Object code	VAR
Data type	Unsigned16

Value description

Sub-index	00$_h$, Guard time
Meaning	Guard time
Access	Read-write
PDO mapping	—
Value range	0...65535
Default value	0000$_h$
Category	Yes

8.5.10   100D$_h$ Life time factor

The object specifies the factor that, together with the time span "guard time", results in the time interval for connection monitoring of an NMT master. Within this period, the NMT slave device expects a monitoring request via Node Guarding from the NMT master.

life time = guard time * life time factor

The value "0" deactivates monitoring of the NMT master.

If there is no connection monitoring through the NMT master during the time interval "life time", the device signals an error and switches to the operating state Fault.

The time factor can be changed in the NMT state "Pre-Operational". The time span "guard time" is set with the object `Guard time (100Ch)`.

Object description

Index	100D$_h$
Name	Life time factor
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, Life time factor
Meaning	Life time factor
Access	Read-write
PDO mapping	—
Value range	0...255
Default value	00$_h$
Category	Yes

8.5.10   1010$_h$ Store parameters

This object supports the saving of parameters in non volatile memory. By read access the device provides information about its saving capabilities. Several parameter groups are distinguished:

- Sub-Index 0 contains the largest Sub-Index that is supported.

- Sub-Index 1 refers to all parameters that can be stored on the device.

- Sub-Index 2 refers to communication related parameters (`Index 1000`$_h$ `- 1FFF`$_h$ manufacturer specific communication parameters).

- Sub-Index 3 refers to application related parameters (`Index 6000`$_h$ `- 9FFF`$_h$ manufacturer specific application parameters).

- Sub-index 4 refers to manufacturer specific parameters.

In order to avoid storage of parameters by mistake, storage is only executed when a specific signature is written to the appropriate Sub-Index. The signature is "save".

Signature ISO 8859	MSB			LSB
ASCII	e	v	a	s
hex	65h	76h	61h	73h

Figure 8.1: Storage write access signature

On reception of the correct signature in the appropriate sub-index the device stores the parameter and then confirms the SDO transmission (initiate download response). If the storing failed, the device responds with an Abort SDO Transfer (`abort code: 0606 0000`$_h$).

If a wrong signature is written, the device refuses to store and responds with Abort SDO Transfer (`abort code: 0800 002x`$_h$).

On read access to the appropriate Sub-Index the device provides information about its storage functionality with the following format:

Unsigned32		
MSB		LSB
bits 31 ... 2	1	0
Reserved (=0)	0/1	0/1

Figure 8.2: Storage read access structure

Bit	Value	Meaning
31 ... 2	0	Reserved (=0)
1	0	Device does not save the parameters autonomously
	1	Device does save the parameters autonomously
2	0	Device does not save the parameters on command
	1	Device does save the parameters on command

Table 8.3: Structure of read access

Autonomous saving means that a device stores the storable parameters in a non-volatile manner without user request.

Object description

Index	1010h
Name	Store parameters
Object code	Array
Data type	Unsigned32

Value description

Sub-index	00h, Largest supported sub-index
Meaning	Largest supported sub-index
Access	Read only
PDO mapping	—
Value range	1h - 4h
Default value	4h
Category	—

Sub-index	01h, Save all parameters
Meaning	Save all parameters
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	—
Category	—

Sub-index	02h, Save communication parameters
Meaning	Save communication parameters
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	—
Category	—

Sub-index	03h, Save application parameters
Meaning	Save application parameters
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	—
Category	—

Sub-index	04h, Save manufacturer parameters
Meaning	Save manufacturer parameters
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	—
Category	—

8.5.11   1011$_h$ Restore default parameters

With this object the default values of parameters according to the communication or device profile are restored. By read access the device provides information about its capabilities to restore these values. Several parameter groups are distinguished:

1) Sub-Index 0 contains the largest Sub-Index that is supported.

2) Sub-Index 1 refers to all parameters that can be restored.Sub-Index 2 refers to communication related parameters (Index 1000$_h$ – 1FFF$_h$ manufacturer specific communication parameters).

3) Sub-Index 3 refers to application related parameters (Index 6000$_h$ – 9FFF$_h$ manufacturer specific application parameters).

4) At Sub-Index 4 - 127 manufacturers may restore their individual choice of parameters.

5) Sub-Index 128 - 254 are reserved for future use.

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

Signature ISO 8859 ASCII	MSB			LSB
	d	a	o	l
hex	64h	61h	6Fh	6Ch

Figure 8.3: Restore default parameters write access signature.

On reception of the correct signature in the appropriate sub-index the device restores the default parameters and then confirms the SDO transmission (initiate download response). If the restoring failed, the device responds with an Abort SDO Transfer (abort code: 0606 0000$_h$). If a wrong signature is written, the device refuses to restore the defaults and responds with an Abort SDO Transfer (abort code: 0800 002x$_h$).

The default values are set valid after the device is reset (reset node for sub-index 1$_h$ – 4$_h$, reset communication for sub-index 2h) or power cycled.

On read access to the appropriate sub-index the device provides parameter restoring capability with the following format:

Unsigned32

bits	MSB	LSB
	31 ... 1	0
	Reserved (=0)	0/1

Figure 8.4: Restore default parameters write access structure.

Bit	Value	Meaning
31 ... 1	0	Reserved (=0)
1	0	Device does not restore the default parameters
	1	Device does restore the default parameters

Table 8.4: Structure of write access

Object description	Index	1011$_h$
	Name	Restore default parameters
	Object code	Array
	Data type	Unsigned32

Value description	Sub-index	00h, Largest supported sub-index
	Meaning	Largest supported sub-index
	Access	Read only
	PDO mapping	—
	Value range	1h - 4h
	Default value	4h
	Category	—

	Sub-index	01$_h$, Restore all parameters
	Meaning	Restore all parameters
	Access	Read-write
	PDO mapping	—
	Value range	Unsigned32
	Default value	—
	Category	—

	Sub-index	02$_h$, Restore communication parameters
	Meaning	Restore communication parameters
	Access	Read-write
	PDO mapping	—
	Value range	Unsigned32
	Default value	—
	Category	—

	Sub-index	03$_h$, Restore application parameters
	Meaning	Restore application parameters
	Access	Read-write
	PDO mapping	—
	Value range	Unsigned32
	Default value	—
	Category	—

	Sub-index	04$_h$, Restore manufacturer parameters
	Meaning	Restore manufacturer parameters
	Access	Read-write
	PDO mapping	—
	Value range	Unsigned32
	Default value	—
	Category	—

8.5.12 1012$_h$ COB-ID time stamp object

Index 1012$_h$ defines the COB-ID of the time-stamp object (TIME). Further, it defines whether the device consumes the TIME or whether the device generates the TIME.

	MSB				LSB
Bits:	31	30	29	28 ... 11	10 ... 0
11-bit ID	0/1	0/1	0	000000000000000000	11-bit Indentifier
29-bit ID	0/1	0/1	1	29-bit Identifier	
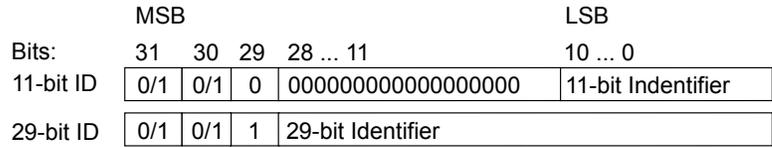
Figure 8.5: Structure of the COB-ID TIME entry

Bit	Value	Meaning
31 (MSB)	0	Device does not consume the TIME message
	1	Device consumes the TIME message
30	0	Device does not produce the TIME message
	1	Device produces the TIME message
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 ... 11	0	If bit 29=0
	X	if bit 29=1: bits 28 ... 11 of 29 bit TIME-COB-ID
10 ... 0 (LSB)	X	Bits 10 ... 0 of TIME-COB-ID

Table 8.5: Description of the TIME COB-ID entry

Bits 29, 30 may be static (not changeable). If a device is not able to generate TIME messages, an attempt to set bit 30 is responded with an abort message (`abort code: 0609 0030`$_h$). Devices supporting the standard CAN frame type only, an attempt to set bit 29 is responded with an abort message (`abort code: 0609 0030`$_h$). It is not allowed to change Bits 0-29, while the object exists (Bit 30=1).

Object description

Index	1012$_h$
Name	COB-ID time stamp message
Object code	VAR
Data type	Unsigned32

Value description

Sub-index	00h, COB-ID time stamp message
Meaning	COB-ID time stamp message
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	0000 0100$_h$
Category	—

8.5.13   1014$_h$ COB-ID emergency error object

Index 1014h defines the COB-ID of the Emergency Object (EMCY).

	MSB				LSB
Bits:	31	30	29	28 ... 11	10 ... 0
11-bit ID	0/1	0	0	000000000000000000	11-bit Indentifier
29-bit ID	0/1	0	1	29-bit Identifier	

Figure 8.6: Structure of the EMCY identifier entry

Bit	Value	Meaning
31 (MSB)	0	EMCY exists / is valid
	1	EMCY does not exist / is not valid
30	0	Reserved (always 0)
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 ... 11	0	If bit 29=0
	X	if bit 29=1: bits 28 ... 11 of 29 bit COB-ID
10 ... 0 (LSB)	X	Bits 10 ... 0 of COB-ID

Table 8.6: Description of the COB-ID entry

Devices supporting the standard CAN frame type only, an attempt to set bit 29 is responded with an abort message (`abort code: 0609 0030`$_h$). It is not allowed to change Bits 0-29, while the object exists (Bit 31=0).

Object description

Index	10142$_h$
Name	COB-ID emergency message
Object code	VAR
Data type	Unsigned32

Value description

Sub-index	00h, COB-ID emergency message
Meaning	COB-ID emergency message
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	0080$_h$ + node ID
Category	—

8.5.14   1015$_h$ Inhibit time EMCY object

The inhibit time for the EMCY message can be adjusted via this entry. If this entry exists it must be writable in the object dictionary. The time has to be a multiple of 100µs

Object description

Index	1015$_h$
Name	Inhibit tme EMCY
Object code	VAR
Data type	Unsigned16

Value description

Sub-index	00h, Inhibit tme EMCY
Meaning	Inhibit tme EMCY
Access	Read-write
PDO mapping	—
Value range	Unsigned16
Default value	0000$_h$
Category	—

8.5.15   1017$_h$ Producer heartbeat time

The producer heartbeat time defines the cycle time of the heartbeat. The producer heartbeat time is 0 if it not used. The time has to be a multiple of 1ms.

Object description

Index	1017$_h$
Name	Producer heartbeat time
Object code	VAR
Data type	Unsigned16

Value description

Sub-index	00h, Producer heartbeat time
Meaning	Producer heartbeat time
Access	Read-write
PDO mapping	—
Value range	Unsigned16
Default value	0000$_h$
Category	—

8.5.16   1018_h Identity object

The object at index 1018$_h$ contains general information about the device. The Vendor ID (sub-index 1$_h$) contains a unique value allocated to each manufacturer.

The manufacturer-specific Product code (sub-index 2$_h$) identifies a specific device version. The manufacturer-specific Revision number (sub-index 3$_h$) consists of a major revision number and a minor revision number. The major revision number identifies a specific CANopen behaviour. If the CANopen functionality is expanded, the major revision has to be incremented. The minor revision number identifies different versions with the same CANopen behaviour.

Byte:	MSB		LSB
	31	16 15	0
	Major Revision Number	Minor Revision Number	
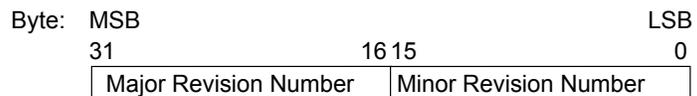
Figure 8.7: Structure of the revision number

The manufacturer-specific serial number (sub-index 4h) identifies a specific device.

Object description

Index	1018$_h$
Name	Identity object
Object code	Array
Data type	Unsigned32

Value description

Sub-index	00$_h$, Largest supported sub-index
Meaning	Largest supported sub-index
Access	Read only
PDO mapping	—
Value range	1$_h$ - 4$_h$
Default value	4$_h$
Category	—

Sub-index	01$_h$, Vendor ID
Meaning	Vendor ID
Access	Read-only
PDO mapping	—
Value range	Unsigned32
Default value	0000 021B$_h$
Category	—

Sub-index	02$_h$, Product code
Meaning	Product code
Access	Read only
PDO mapping	—
Value range	Unsigned32
Default value	0000 0000$_h$
Category	—

Sub-index	03$_h$, Revision number
Meaning	Revision number
Access	Read only
PDO mapping	—
Value range	Unsigned32
Default value	0000 0507$_h$
Category	—

Sub-index	04$_h$, Serial number
Meaning	Serial number
Access	Read only
PDO mapping	—
Value range	Unsigned32
Default value	—
Category	—

8.5.18   1400 – 1403$_h$ Receive PDO communications parameter

Contains the communication parameters for the PDOs the device is able to receive. The type of the PDO communication parameter (20h) is described in Section 9.5.4 of CiA DS-301: CANopen Application Layer and Communications Profile. The sub-index 0h contains the number of valid entries within the communication record. Its value is at least 2. If inhibit time supported the value is 3. At sub-index 1h resides the COB-ID of the PDO. This entry has been defined as UNSIGNED32 in order to cater for 11-bit CAN Identifiers (CAN 2.0A) as well as for 29-bit CAN identifiers (CAN 2.0B).

	MSB				LSB
Bits:	31	30	29	28 ... 11	10 ... 0
11-bit ID	0/1	0/1	0	000000000000000000	11-bit Indentifier
29-bit ID	0/1	0/1	1	29-bit Identifier	
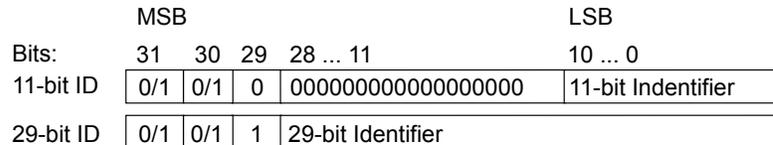
Figure 8.8: Structure of the PDO COB-ID entry

Bit	Value	Meaning
31 (MSB)	0	PDO Exists/Is Valid
	1	PDO Does Not Exist/Is Not Valid
30	0	RTR is Allowed on this PDO
	1	RTR is Not Allowed on this PDO
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 ... 11	0	If bit 29=0
	X	if bit 29=1: bits 28 ... 11 of 29 bit COB-ID
10 ... 0 (LSB)	X	Bits 10 ... 0 of COB-ID

Table 8.7: Description of the PDO COB-ID entry

The PDO valid/not valid allows to select which PDOs are used in the operational state. There may be PDOs fully configured (e.g. by default) but not used, and therefore set to "not valid" (deleted). The feature is necessary for devices supporting more than 4 RPDOs or 4 TPDOs, because each device has only default identifiers for the first four RPDOs/TPDOs. Devices supporting the standard CAN frame type only or do not support Remote Frames, an attempt to set bit 29 to 1 or bit 30 to 0 is responded with an abort message (`abort code: 0609 0030`$_h$). It is not allowed to change bit 0-29 while the PDO exists (Bit 31=0).

The transmission type (sub-index 2) defines the transmission/reception character of the PDO. On an attempt to change the value of the transmission type to a value that is not supported by the device an abort message (`abort code: 0609 0030`$_h$) is generated.

Transmission type	PDO transmission				
	cyclic	acyclic	sync	async	RTR only
0		X	X		
1 – 240	X		X		
241 – 251	Reserved				
252			X		X
253				X	X
254				X	
255				X	

Table 8.8: Description of the PDO COB-ID entry

Synchronous (transmission types 0-240 and 252) means that the transmission of the PDO shall be related to the SYNC object. Preferably the devices use the SYNC as a trigger to output or actuate based on the previous synchronous Receive PDO respectively to update the data transmitted at the following synchronous Transmit PDO. Details of this mechanism depend on the device type and are defined in the device profile if applicable.

Asynchronous means that the transmission of the PDO is not related to the SYNC object. A transmission type of zero means that the message shall be transmitted synchronously with the SYNC object but not periodically. A value between 1 and 240 means that the PDO is transferred synchronously and cyclically. The transmission type indicating the number of SYNC which are necessary to trigger PDO transmissions.

Receive PDOs are always triggered by the following SYNC upon receiption of data independent of the transmission types 0 - 240. The transmission types 252 and 253 mean that the PDO is only transmitted on remote transmission request. At transmission type 252, the data is updated (but not sent) immediately after reception of the SYNC object.

At transmission type 253 the data is updated at the reception of the remote transmission request (hardware and software restrictions may apply). These value are only possible for T_PDOs.

For T_PDOs transmission type 254 means, the application event is manufacturer specific (manufacturer specific part of the Object Dictionary), transmission type 255 means, the application event is defined in the device profile. R_PDOs with that type trigger the update of the mapped data with the reception.

Sub-index 3_h contains the inhibit time. This time is a minimum interval for PDO transmission. The value is defined as multiple of 100µs. It is not allowed to change the value while the PDO exists (Bit 31 of sub-index 1 is 0).

Sub-index 4h is reserved. It does not have to be implemented, in this case read or write access leads to Abort SDO Transfer (`abort code: 0609 0011`$_h$).

In mode 254/255 additionally an event time can be used for T_PDO. If an event timer exists for a T_PDO (value not equal to 0) the elapsed timer is considered to be an event. The event timer elapses as multiple of 1 ms of the entry in sub-index 5_h of the T_PDO. This event will cause the transmission of this T_PDO in addition to otherwise defined events. The occurrence of the events set the timer. Independent of the transmission type the R_PDO event timer is used recognize the expiration of the R_PDO.

Object description

Index	1400 – 1403$_h$
Name	1st, 2nd, 3rd and 4th receive PDO parameters
Object code	Record
Data type	—

Value description

Sub-index	00$_h$, Largest supported sub-index
Meaning	Largest supported sub-index
Access	Read only
PDO mapping	—
Value range	Unsigned8
Default value	05$_h$
Category	—

Sub-index	01$_h$, COB-ID used by PDO
Meaning	COB-ID used by PDO
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1400$_h$= 0200$_h$+ node ID 1401$_h$= 0300$_h$+ node ID 1402$_h$= 0400$_h$+ node ID 1403$_h$= 0500$_h$+ node ID
Category	Yes

Sub-index	02$_h$, Transmission type
Meaning	Transmission type
Access	Read-write
PDO mapping	—
Value range	Unsigned8
Default value	255 (asynchronous)
Category	Yes

Sub-index	03$_h$, Inhibit time
Meaning	Inhibit time
Access	Read-write
PDO mapping	—
Value range	Unsigned16
Default value	0000$_h$
Category	Yes

Sub-index	05$_h$, Event timer
Meaning	Event timer
Access	Read-write
PDO mapping	—
Value range	Unsigned16
Default value	0000$_h$
Category	Yes

8.5.19   1600 – 1603$_h$ Receive PDO mapping parameter

Contains the mapping for the PDOs the device is able to receive. The type of the PDO mapping parameter (21$_h$) is described in Section 9.5.4 of CiA DS-301: CANopen Application Layer and Communications Profile.. The sub-index 0$_h$ contains the number of valid entries within the mapping record. This number of entries is also the number of the application variables which shall be transmitted/received with the corresponding PDO. The sub-indices from 1h to number of entries contain the information about the mapped application variables. These entries describe the PDO contents by their index, sub-index and length. All three values are hexadecimal coded. The length entry contains the length of the object in bit (1..40$_h$).

This parameter can be used to verify the overall mapping length. It is mandatory.

The structure of the entries from sub-index 1$_h$ – 40$_h$ is as follows:

Byte: MSB		LSB
Index (16-bit)	Sub-Index (8-Bit)	Object Length (8-Bit)

Figure 8.9: Structure of the PDO mapping entry

If the change of the PDO mapping cannot be executed (e.g. the PDO length is exceeded or the SDO client attempts to map an object that cannot be mapped) the device responds with an Abort SDO Transfer Service.

Sub-index 0 determines the valid number of objects that have been mapped. For changing the PDO mapping first the PDO has to be deleted, the sub-index 0 must be set to 0 (mapping is deactivated). Then the objects can be remapped. When a new object is mapped by writing a sub-index between 1 and 64, the device may check whether the object specified by index / sub-index exists. If the object does not exist or the object cannot be mapped, the SDO transfer must be aborted with the Abort SDO Transfer Service with one of the abort codes 0602 0000h or 0604 0041h.

After all objects are mapped sub-index 0 is set to the valid number of mapped objects. Finally the PDO will be created by writing to its communication parameter COB-ID. When sub-index 0 is set to a value >0 the device may validate the new PDO mapping before transmitting the response of the SDO service. If an error is detected the device has to transmit the Abort SDO Transfer Service with one of the abort codes 0602 0000h, 0604 0041h or 0604 0042h.

When sub-index 0 is read the actual number of valid mapped objects is returned. If data types (Index 1h-7h) are mapped they serve as „dummy entries". The corresponding data in the PDO is not evaluated by the device. This optional feature is useful e.g. to transmit data to several devices using one PDO, each device only utilising a part of the PDO. It is not possible to create a dummy mapping for a T_PDO.

A device that supports dynamic mapping of PDOs must support this during the state PREOPERATIONAL state. If dynamic mapping during the state OPERATIONAL is supported, the SDO client is responsible for data consistency.
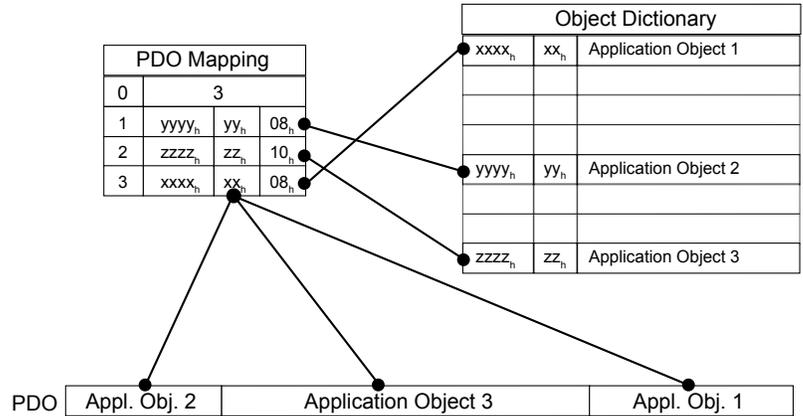
Figure 8.10: Principle of PDO mapping

Object description

Index	1600 – 1603$_h$
Name	1st, 2nd, 3rd and 4th receive PDO mapping
Object code	Record
Data type	—

Value description

Sub-index	00$_h$, Number of mapped application objects
Meaning	Number of mapped application objects
Access	Read-write
PDO mapping	—
Value range	1 – 64
Default value	1600$_h$= 1 1601$_h$= 2 1602$_h$= 2 1603$_h$= 2
Category	Yes

Sub-index	01$_h$, PDO mapping 1st application object
Meaning	PDO mapping 1st application object
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1600$_h$= 6040 0010$_h$ 1601$_h$= 6040 0010$_h$ 1602$_h$= 6040 0010$_h$ 1603$_h$= 6040 0010$_h$
Category	Yes

Sub-index	02$_h$, PDO mapping 2nd application object
Meaning	PDO mapping 2nd application object
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1600$_h$= 0000 0000$_h$ 1601$_h$= 607A 0020$_h$ (Profile position – target position) 1602$_h$= 60FF 0020$_h$ (Profile velocity – target velocity) 1603$_h$= 6071 0020$_h$ (Profile torque – target torque)
Category	Yes

Sub-index	03 – 08$_h$, PDO mapping *n*th application object
Meaning	PDO mapping *n*th application object
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1600$_h$= 0000 0000$_h$ 1601$_h$= 0000 0000$_h$ 1602$_h$= 0000 0000$_h$ 1603$_h$= 0000 0000$_h$
Category	Yes

8.5.20   1800 – 1803$_h$ Receive PDO mapping parameter

Contains the communication parameters for the PDOs the device is able to transmit. The type of the PDO communication parameter (20$_h$) is described in 9.5.4 of CiA DS-301: CANopen Application Layer and Communications Profile. A detailed description of the entries is done in the section for the Receive PDO Communication Parameter (1400$_h$ – 1403$_h$).

Object description

Index	1800 – 1803$_h$
Name	1st, 2nd, 3rd and 4th transmit PDO parameters
Object code	Record
Data type	—

Value description

Sub-index	00$_h$, Largest supported sub-index
Meaning	Largest supported sub-index
Access	Read only
PDO mapping	—
Value range	Unsigned8
Default value	05$_h$
Category	—

Sub-index	01$_h$, COB-ID used by PDO
Meaning	COB-ID used by PDO
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1800$_h$= 0180$_h$+ node ID 1801$_h$= 0280$_h$+ node ID 1802$_h$= 0380$_h$+ node ID 1803$_h$= 0480$_h$+ node ID
Category	Yes

Sub-index	02$_h$, Transmission type
Meaning	Transmission type
Access	Read-write
PDO mapping	—
Value range	Unsigned8
Default value	255 (asynchronous)
Category	Yes

Sub-index	03$_h$, Inhibit time
Meaning	Inhibit time
Access	Read-write
PDO mapping	—
Value range	Unsigned16
Default value	0000$_h$
Category	Yes

Sub-index	05$_h$, Event timer
Meaning	Event timer
Access	Read-write
PDO mapping	—
Value range	Unsigned16
Default value	1800$_h$= 0 1801$_h$= 100 1802$_h$= 100
Category	Yes

8.5.21   1A00 – 1A02$_h$ Transmit PDO mapping parameter

Contains the mapping for the PDOs the device is able to transmit. The type of the PDO mapping parameter (21$_h$) is described in 9.5.4 of CiA DS-301: CANopen Application Layer and Communications Profile. A detailed description of the entries is done in the Section 8.5.19 for the Receive PDO Mapping Parameter (1600$_h$ – 1602$_h$).

Object description

Index	1A00 – 1A02$_h$
Name	1st, 2nd and 3rd transmit PDO mapping
Object code	Record
Data type	—

Value description

Sub-index	00$_h$, Number of mapped application objects
Meaning	Number of mapped application objects
Access	Read-write
PDO mapping	—
Value range	1 – 64
Default value	1A00$_h$= 1 1A01$_h$= 2 1A02$_h$= 2 1A03$_h$= 1
Category	Yes

Sub-index	01$_h$, PDO mapping 1st application object
Meaning	PDO mapping 1st application object
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1A00$_h$= 6041 0010$_h$ (Status Word) 1A01$_h$= 6041 0010$_h$ (Status Word) 1A02$_h$= 6041 0010$_h$ (Status Word) 1A03$_h$= 6041 0010$_h$ (Status Word)

Category	Yes

Sub-index	02_h, PDO mapping 2nd application object
Meaning	PDO mapping 2nd application object
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1800_h= 0000 0000$_h$ 1801_h= 6064 0020$_h$ (Profile position – position actual value) 1802_h= 606C 0020$_h$ (Profile velocity – velocity actual value) 1803_h= 0000 0000$_h$
Category	Yes

Sub-index	03 – 08_h, PDO mapping nth application object
Meaning	PDO mapping nth application object
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1600_h= 0000 0000$_h$ 1601_h= 0000 0000$_h$ 1602_h= 0000 0000$_h$ 1603_h= 0000 0000$_h$
Category	Yes

8.6 Details of object group 2000$_h$ (Mfg specific)

The objects detailed in this section are Schneider Electric Motion USA manufacturer specific configuration objects to configure the manufacturer object specific to the Lexium MDrive CANopen node.

8.6.1 2000$_h$ I/O configuration

This object facilitates the configuration of the input and output points available on Lexium MDrive CANopen devices. The sub-indexes 4 and 5, which may be configured as sinking or sourcing inputs or outputs. Each bit of the sub-indices are mapped to a particular I/O point with I/O 12 being the Most Significant bit (MSb) and I/O 1 being the Least Significant bit (LSb). The configuration options are:

1)   Sub-Index 01$_h$: Reserved

2)   Sub-Index 02$_h$: Reserved

3)   Sub-Index 03$_h$: Reserved

4)   Sub-Index 04$_h$: This allows the configuration of inputs to read inverted polarity. By default this is deactivated and setting the bit to a 1 will change the invert polarity read by the input. See Optional Application FE, Object 60DF.

5)   Sub-Index 05$_h$: This allows the configuration of outputs to invert the output polarity. See Optional Application FE, Object 60FE.

The I/O configuration is saved using the Store Parameters Object (1010h).

Object description

Index	2000$_h$
Name	I/O configuration
Object code	ARRAY
Data type	Unsigned8

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	05$_h$
Category	—

Sub-index	01$_h$ - 03$_h$
Meaning	Reserved
Access	—
PDO mapping	—
Value range	—
Default value	—
Category	Yes

Sub-index	04$_h$, Configure as polarity in
Meaning	Configure as polarity in (See object 60FD$_h$ sub-index 01$_h$)
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	05$_h$, Configure as polarity out
Meaning	Configure as polarity out
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$ (See object 60FE$_h$ sub-index 01$_h$)
Category	Yes

8.6.2   2003$_h$ Configure input switches

Object 2002 facilitates the configuration of input switches. Input switches may be configured as the following types:

1)   Home
2)   Positive Limit
3)   Negative Limit
4)   Inhibit (Inhibit Switch function is configured by Object 2007h)

Object description

Index	2002$_h$
Name	Configure input switches
Object code	ARRAY
Data type	Unsigned8

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	04$_h$
Category	—

Sub-index	01$_h$, Configure input as home
Meaning	Configure input as home switch (I/O point bit(s) = 1$_b$ to select as home)
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	02$_h$, Configure input as positive limit
Meaning	Configure input as + Limit (I/O point bit(s) = 1$_b$ to select as + limit)
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	03$_h$, Configure input as negative limit
Meaning	Configure input as – Limit (I/O point bit(s) = 1$_b$ to select as – limit)
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	04$_h$, Configure inhibit switch
Meaning	Configure inhibit switch (I/O point bit(s) = 1$_b$ to select as inhibit). Use object 2007$_h$ to define inhibit function.
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

8.6.3   2004$_h$ Configure input filter mask

The Input filter mask object configure s the device to filter the selected inputs. Sub-indices 01$_h$ through 04$_h$ define the inputs to which filtering will be applied. Object 2006$_h$ defines the filter time applied to each input.
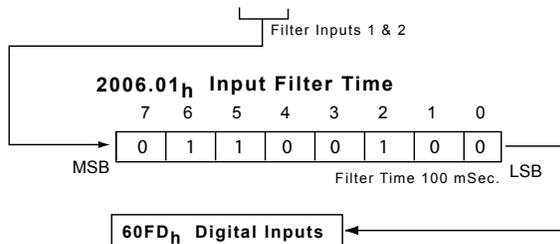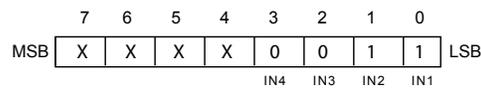
Figure 8.12: Input filter mask

Object description

Index	2004$_h$
Name	Configure input mask
Object code	ARRAY
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	08$_h$
Category	—

Sub-index	01$_h$, Configure mask for Input 1
Meaning	Configure mask for Input 1
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	01$_h$
Category	Yes

Sub-index	02$_h$, Configure mask for Input 2
Meaning	Configure mask for Input 2
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	02$_h$
Category	Yes

Sub-index	03$_h$, Configure mask for Input 3
Meaning	Configure mask for Input 3
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	04$_h$
Category	Yes

Sub-index	04$_h$, Configure mask for Input 4
Meaning	Configure mask for Input 4
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	08$_h$
Category	Yes

Sub-index	05ₕ - 08ₕ, reserved
Meaning	reserved
Access	reserved
PDO mapping	—
Value range	—
Default value	—
Category	Yes

8.6.4   2006$_h$ Configure input filter time

This object sets the input filter time in milliseconds. Each sub-index applies to a specific input where sub-index 01h applies to input 1, sub-index 02h applies to input 2 and etc.

Object description

Index	2006$_h$
Name	Configure input filter
Object code	ARRAY
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	08$_h$
Category	—

Sub-index	01$_h$, Configure filter for Input 1
Meaning	Configure filter for Input 1
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	02$_h$, Configure filter for Input 2
Meaning	Configure filter for Input 2
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	03$_h$, Configure filter for Input 3
Meaning	Configure filter for Input 3
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

V1.00, 09.2014

Sub-index	05$_h$, Configure filter for Input 4
Meaning	Configure filter for Input 4
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	05$_h$, Configure filter for Input 9
Meaning	Configure filter for Input 9
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	06$_h$, Configure filter for Input 10
Meaning	Configure filter for Input 10
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	07$_h$, Configure filter for Input 11
Meaning	Configure filter for Input 11
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

Sub-index	08$_h$, Configure filter for Input 12
Meaning	Configure filter for Input 12
Access	Read-write
PDO mapping	—
Value range	00 – FF$_h$
Default value	00$_h$
Category	Yes

8.6.7 2007$_h$ Inhibit switch reaction

This object allows the user to configure different actions for an inhibit switch (See `Object 2002`$_h$`, Sub-index 4`$_h$).

This object will function through Control word overwrites and overrides. The inhibit switch reaction is set using sub-index 01$_h$.

Object description

Index	2007$_h$
Name	Inhibit switch reaction
Object code	ARRAY
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	01$_h$
Category	—

Sub-index	01$_h$, Inhibit switch reaction
Meaning	Inhibit switch reaction
Access	Read-write
PDO mapping	—
Value range	1 – 12
Default value	12
Category	Yes

Value	Meaning
0	No action
1	Fault signal - Control word overwrite
2	Fault signal - Control word override
3	Disable voltage command - Control word overwrite
4	Disable voltage command - Control word override
5	Quick stop command - Control word overwrite
6	Quick stop command - Control word override
7	Shutdown command - Control word overwrite
8	Shutdown command - Control word override
9	Disable operation command - Control word overwrite
10	Disable operation command - Control word override
11	Halt command - Control word overwrite
12	Halt command - Control word override

Table 8.9: Inhibit switch reactions

8.6.8   2008$_h$ Output definition

This object allows the user to configure one or more outputs as brake outputs. For an explanation of brake functions, see Object 2035$_h$: Brake Timers

Object description

Index	2008$_h$
Name	Output definition
Object code	ARRAY
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	02$_h$
Category	—

Sub-index	01$_h$, Brake output defined
Meaning	Brake output defined
Access	Read-write
PDO mapping	—
Value range	Unsigned8
Default value	00$_h$
Category	Yes

Sub-index	02$_h$, Target reached output defined
Meaning	Target reached output defined
Access	Read-write
PDO mapping	—
Value range	Unsigned8
Default value	00$_h$
Category	Yes

Output selected	Setting
1	Sub-index = 01$_h$
2	Sub-index = 02$_h$
3	Sub-index = 04$_h$

Table 8.10: Brake and target reached output definition

8.6.9   2010ₕ Analog input configuration

This object allows the user to configure the 12-bit Analog Input. There are 3 sub-indices that set the configuration properties for the input:

1) Sub-Index 01$_h$: Analog Full Scale. This has a range of 0 to 4095 and sets the full scale of the analog input.

2) Sub-Index 02$_h$: This sets the type of device the Analog Input will read. It can be set for two modes, Voltage with ranges of 0 to 5V or 0 to 10V, or Current with an input range of 0 to 20 mA.

3) Sub-Index 03$_h$: This sets the filtering for the Analog Input. In the 0 (default) setting the filtering is off.

Object description

Index	2010$_h$
Name	Analog input configuration
Object code	ARRAY
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	03$_h$
Category	—

Sub-index	01$_h$, Analog input reading
Meaning	Analog input reading
Access	Read-only
PDO mapping	Yes – T_PDO
Value range	0 – 4095$_d$ (0000 – FFFF$_h$ – Unsigned16)
Default value	—
Category	—

Sub-index	02$_h$, Analog input configuration
Meaning	Analog input configuration
Access	Read-write
PDO mapping	—
Value range	00$_h$, 02$_h$, or 08$_h$ (Unsigned8)
Default value	00$_h$
Category	Yes

Sub-index 02$_h$	Analog input mode
00$_h$	0 to 5 V scale
02$_h$	4 to 20 mA scale
08$_h$	0 to 10 V scale

Sub-index	03$_h$, Analog filter level
Meaning	Analog filter level
Access	Read-write
PDO mapping	—
Value range	00 – 31$_h$ (Unsigned8)
Default value	00$_h$
Category	Yes

8.6.10   2014$_h$ Auxiliary power input monitoring

This object allows for the user to monitor the voltage level of the aux-power input, as well as establish low and high level warning limits. This input provides power to logic and feedback circuitry in the event of a main power loss.

The reading is given as voltage X 10 i.e. 244 = 24.4 VDC.

Object description

Index	2014$_h$
Name	Auxiliary power input monitoring
Object code	ARRAY
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	03$_h$
Category	—

Sub-index	01$_h$, Aux-power input reading
Meaning	Aux-power input reading
Access	Read-only
PDO mapping	Yes – T_PDO
Value range	0 – 280$_d$ (0000 – 0118$_h$ – Unsigned16)
Default value	—
Category	—

Sub-index	02$_h$, Aux-power input low-level warning
Meaning	Aux-power input low-level warning
Access	Read-write
PDO mapping	—
Value range	0 – 280$_d$ (0000 – 0118$_h$ – Unsigned16)
Default value	00$_h$
Category	Yes

Sub-index	03$_h$, Aux-power input high-level warning
Meaning	Aux-power input high-level warning
Access	Read-write
PDO mapping	—
Value range	0 – 280$_d$ (0000 – 0118$_h$ – Unsigned16)
Default value	280$_d$ (0118h)
Category	Yes

8.6.11   2015$_h$ +VDC input monitoring

This object allows for the user to monitor the voltage level of the motor power input (+VDC), as well as establish low and high level warning limits.

The reading is given as voltage X 10 i.e. 448 = 44.8 VDC.

Object description

Index	2015$_h$
Name	+VDC input monitoring
Object code	ARRAY
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	03$_h$
Category	—

Sub-index	01$_h$, +VDC input monitoring
Meaning	+VDC input monitoring
Access	Read-only
PDO mapping	Yes – T_PDO
Value range	0 – 800$_d$ (0000 – 0320$_h$ – Unsigned16)
Default value	—
Category	—

Sub-index	02$_h$, +VDC input low-level warning
Meaning	+VDC input low-level warning
Access	Read-write
PDO mapping	—
Value range	0 – 800$_d$ (0000 – 0320$_h$ – Unsigned16)
Default value	100$_d$ (64$_h$)
Category	Yes

Sub-index	03$_h$, +VDC input high-level warning
Meaning	+VDC input high-level warning
Access	Read-write
PDO mapping	—
Value range	0 – 800$_d$ (0000 – 0320$_h$ – Unsigned16)
Default value	580$_d$ (244$_h$)
Category	Yes

8.6.12   2018$_h$ PCB temperature options

This object reflects the temperature of the PCB as measured at the micro-processor. There are 3 sub-indices:

1) Sub-Index 01$_h$: Read-only sub-index that reads the internal temperature of the device. May be mapped to a PDO

1) Sub-Index 02$_h$: Temperature Warning parameter allows for the setting of a parameter that will generate an error (`Index 1003 – 0016 4210`$_h$) if the warning threshold is reached. the default is 80°C.

1) Sub-Index 03$_h$: This sub-index sets the threshold for a temperature fault. Note that the outputs of the device will disable at 85°C regardless of the setting for this parameter. if reached, the error message will be locates at `Index 1003`$_h$. The error code is byte will read `0008 4210`$_h$.

The units for this object are degrees Celsius (°C).

Object description

Index	2018$_h$
Name	PCB temperature options
Object code	ARRAY
Data type	Signed8

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	03$_h$
Category	—

Sub-index	01$_h$, PCB temperature reading
Meaning	PCB temperature reading
Access	Read-only
PDO mapping	Yes – T_PDO
Value range	—
Default value	—
Category	—

Sub-index	02$_h$, PCB temperature warning threshold
Meaning	PCB temperature warning threshold
Access	Read-write
PDO mapping	—
Value range	-50 to +120$_d$ (Signed8)
Default value	80$_d$
Category	Yes

Sub-index	03$_h$, PCB temperature fault
Meaning	PCB temperature fault
Access	Read-write
PDO mapping	—
Value range	-50 to +120$_d$ (Signed8)
Default value	85$_d$
Category	Yes

8.6.13   2019$_h$ Bridge temperature options

This object reflects the temperature of the output bridge as measured at the bridge. There are 3 sub-indices:

1) Sub-Index 01$_h$: Read-only sub-index that reads the bridge temperature of the device. May be mapped to a PDO

1) Sub-Index 02$_h$: Temperature Warning parameter allows for the setting of a parameter that will generate an error (`Index 1003 - 0016 4210`$_h$) if the warning threshold is reached. the default is 80°C.

1) Sub-Index 03$_h$: This sub-index sets the threshold for a temperature fault. Note that the outputs of the device will disable at 85°C regardless of the setting for this parameter. if reached, the error message will be locates at `Index 1003`$_h$. The error code is byte will read `0008 4210`$_h$.

The units for this object are degrees Celsius (°C).

Object description

Index	2018$_h$
Name	Bridge temperature options
Object code	ARRAY
Data type	Signed8

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	03$_h$
Category	—

Sub-index	01$_h$, Bridge temperature reading
Meaning	Bridge temperature reading
Access	Read-only
PDO mapping	Yes – T_PDO
Value range	—
Default value	—
Category	—

Sub-index	02$_h$, Bridge temperature warning threshold
Meaning	Bridge temperature warning threshold
Access	Read-write
PDO mapping	—
Value range	-50 to +120$_d$ (Signed8)
Default value	80$_d$
Category	Yes

Sub-index	03$_h$, Bridge temperature fault
Meaning	Bridge temperature fault
Access	Read-write
PDO mapping	—
Value range	-50 to +120$_d$ (Signed8)
Default value	85$_d$
Category	Yes

8.6.14   2020$_h$ Software limits as hardware

This object defines the actions taken when the Position Software Limit for Object 2022$_h$ is reached. It consists of 2 sub-indices.

2)   Sub-index 01$_h$:Limit reached flag. This flag will be set based upon the status of a limit, it will register whether the limit reached is a hardware limit or a software limit. The statusword (6041$_h$), bit 11, internal limit active will set whenever the flag is not 0. The action taken for a limit reached condition will be determined by the limit mask sub-index.

Status	Bit 3	Bit 4	Bit 1	Bit 0
Negative hardware limit reached	0	0	0	1
Positive hardware limit reached	0	0	1	0
Negative software limit reached	0	1	0	0
Positive software limit reached	1	0	0	0

Table 8.11: Description of limit reached flag 2020.01$_h$

3)   Sub-index 02$_h$: Limit reached mask
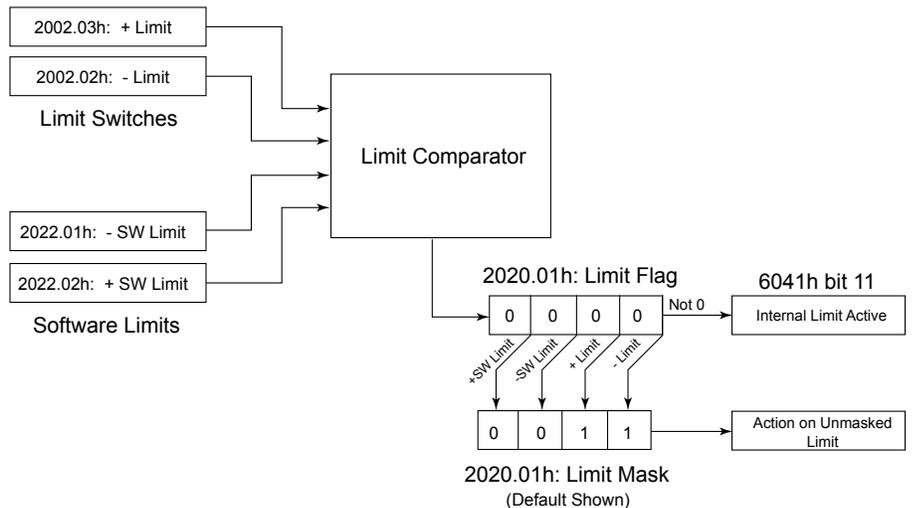
Figure 8.13: Software limits as hardware functions

Object description

Index	2020$_h$
Name	Software limits as hardware
Object code	ARRAY
Data type	Unsigned8

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	02$_h$
Category	—

Sub-index	01$_h$, Limit reached flag
Meaning	Limit reached flag
Access	Read-write
PDO mapping	—
Value range	00 – 0F$_h$
Default value	00$_h$
Category	—

Sub-index	02$_h$, Limit reached mask
Meaning	Limit reached mask
Access	Read-write
PDO mapping	—
Value range	00 – 0F$_h$
Default value	03$_h$
Category	Yes

8.6.15   2022$_h$ Actual position software limit

This object defines the software limit based on set negative and positive limits set in actual position counts. See object 2020$_h$ for a description of software limit functionality and configuration.

1)   Sub-index 01$_h$: Actual negative limit

2)   Sub-index 02$_h$: Actual positive limit

Object description

Index	2022$_h$
Name	Actual position software limit
Object code	ARRAY
Data type	Signed32

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	02$_h$
Category	—

Sub-index	01$_h$, Actual negative limit
Meaning	Actual negative limit
Access	Read-write
PDO mapping	—
Value range	Signed32
Default value	8000 0000$_h$
Category	—

Sub-index	02$_h$, Actual positive limit
Meaning	Actual positive limit
Access	Read-write
PDO mapping	—
Value range	Signed32
Default value	7FFF FFFF$_h$
Category	Yes

8.6.16   2030_h Output bridge polarity

This object defines the polarity of the output bridge where positive=clockwise and negative=counter clockwise (default). By changing this to a negative integer between -128 to -1 the polarity of the bridge, thus the default CW/CCW motor direction can be swapped.

There is no value assigned to the number, any negative integer from -128 to -1 will reverse the bridge polarity, any positive integer from 0 to 127 will reset the polarity to the default CW/CCW motor direction configuration.

Object description

Index	2030$_h$
Name	Output bridge polarity
Object code	VAR
Data type	Signed8

Value description

Sub-index	00$_h$, Output bridge polarity
Meaning	Output bridge polarity
Access	Read-write
PDO mapping	—
Value range	-128 to 127$_d$
Default value	0$_d$
Category	Yes

8.6.17   2031_h Unit options (encoder enable, trip/capture enable)

This object defines the configuration of the following unit options:

1) Encoder sync actions (bits 5, 4). Encoder sync will determine whether the encoder counter, position counter or -home offset will be the master counter for synchronizing the counters. If -home offset is used it will function as homing 35.

2) Encoder enable (bit 3)

3) Trip output/capture input (bit 2)

Note: Encoder functions only apply to the Lexium MDrive products. The MForce products do not have closed loop capability.

Object description

Index	2031$_h$
Name	Unit options (encoder enable, trip/capture enable)
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, Unit options (encoder enable, trip/capture enable)
Meaning	Unit options (encoder enable, trip/capture enable)
Access	Read-write
PDO mapping	—
Value range	Unsigned8
Default value	00$_h$
Category	Yes

Bits		7	6	5	4	3	2	1	0
Function		X	X	sync_action		e/e	c/t	X	X
Encoder auto-sync action (sync_action)									
Bit 5	Bit 4	Bits 5 and 4 control encoder sync action							
0	0	No action							
0	1	Position synced to encoder master							
1	0	Encoder synced to position master							
1	1	Position and encoder synced to –home offset							
Encoder enable (e/e)									
Bit 3	0	Encoder not enabled							
	1	Encoder operation enabled							
Capture/trip select (c/t)									
Bit 2	0	Will operate as a capture input (configure using 2033_h)							
	1	Will operate as a trip output (configure using 2038_h)							

Table 8.12: Description of unit options object 2031.00_h

8.6.18   2033$_h$ Capture input parameters

This object configures the functionality of the capture input.

1)   Sub-index 01$_h$: Capture input control: sets a bit that will enable the capture input. Note that the capture input must also be selected using `object 2031`$_h$.

2)   Sub-index 02$_h$: Position captured flag. Displays the status of a position capture by setting the least significant bit (lsb) of an 8-bit unsigned integer. Write all ones to clear the flag.

3)   Sub-index 03$_h$: Capture input filter time. This sub-index configures the filtering for the capture input.

4)   Sub-index 04$_h$: Captured position. This sub-index holds the captured position.

Object description

Index	2033$_h$
Name	Capture input parameters
Object code	REC
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	04$_h$
Category	—

Sub-index	01$_h$, Capture input enable
Meaning	Capture input enable
Access	Read-write
PDO mapping	—
Value range	0/1 (Unsigned8)
Default value	0 (disabled)
Category	Yes

Sub-index	02$_h$, Position captured flag
Meaning	Position captured flag
Access	Read-write
PDO mapping	—
Value range	0/1 (Unsigned8)
Default value	0 (no position captured)
Category	Yes

Sub-index	03_h, Capture input filter time
Meaning	Capture input filter time
Access	Read-write
PDO mapping	—
Value range	0 – 9 (Unsigned8)
Default value	0 (50ns)
Category	Yes

Value	Filter		Value	Filter		Value	Filter
0	50ns		4	500ns		8	6.5µs
1	150ns		5	900ns		9	12.9µs
2	200ns		6	1.7µs			
3	300ns		7	3.3µs			

Sub-index	04_h, Captured position
Meaning	Captured position
Access	Read-only
PDO mapping	—
Value range	Integer32
Default value	0 (no position)
Category	—

8.6.19   2034_h Bridge on settle time

Establishes the time in milliseconds that current in the bridge is allowed to stabilize after power on. This index will delay the device entering operation enabled mode by the time set (0 to 1000ms). It is also a factor in the brake logic block. See `object 2035`_h for functional block diagram.

Object description

Index	2034_h
Name	Bridge on settle time
Object code	Array
Data type	Unsigned16

Value description

Sub-index	00_h, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	01_h
Category	—

Sub-index	01_h, Bridge on settle time
Meaning	Bridge on settle time
Access	Read-write
PDO mapping	—
Value range	$0 - 1000_d$
Default value	0
Category	Yes

8.6.20 2035_h Brake settle allow time

Establishes the time in milliseconds that the brake is allowed to settle. This object works in cooperation with `object 2034.01`$_h$, Bridge on settle allow time. The sequence of events follows for a braking operation:

1) Bridge power turns on, `object 2034.01`$_h$ begins timing the amount of milliseconds specified in sub-index 01_h. This will be the time between bridge power enabled and brake off. This time will also allow time for settling before initial synchronizing with encoder counts.

2) Break discrete turns off, `object 2035.02`$_h$ specifies the time delay from set brake off to allow for motor movement.

3) Device enters the operation enabled state of the state machine. Motion occurs.

4) Motion ceases, the brake engages, `object 2035.01`$_h$ specifies the time delay from set brake on to removal of bridge power.
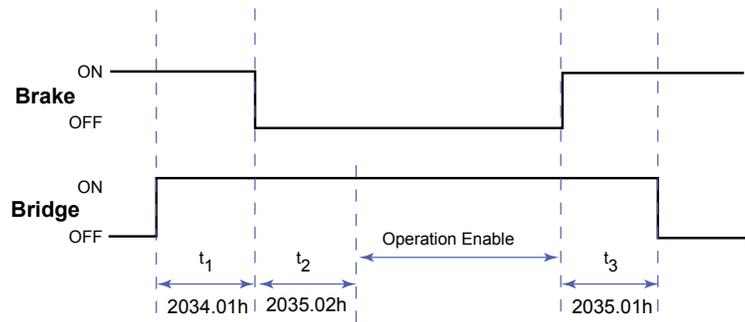
Figure 8.14: Bridge to brake timing
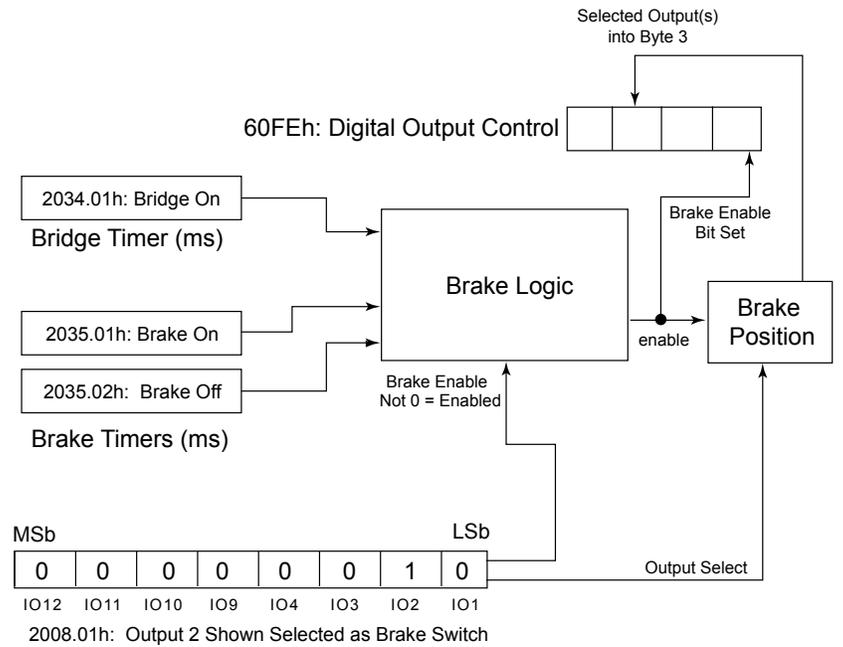
Figure 8.15: Brake functions block diagram

Object description		
	Index	2035$_h$
	Name	Brake settle allow time
	Object code	ARRAY
	Data type	—

Value description		
	Sub-index	00$_h$, Number of entries
	Meaning	Number of entries
	Access	Read only
	PDO mapping	—
	Value range	—
	Default value	02$_h$
	Category	—
	Sub-index	01$_h$, Brake on settle time
	Meaning	Brake on settle time
	Access	Read-write
	PDO mapping	—
	Value range	0 – 2000$_d$ (ms)
	Default value	0
	Category	Yes

Sub-index	02$_h$, Brake off settle time
Meaning	Brake off settle time
Access	Read-write
PDO mapping	—
Value range	0 – 1000$_d$ (ms)
Default value	0
Category	Yes

8.6.21   2036$_h$ Hold current delay time

Defines the delay tim in milliseconds between the device switching from the run current (`object 2204`$_h$) to the holding current (`object 2205`$_h$)

Object description

Index	2036$_h$
Name	Hold current delay time
Object code	VAR
Data type	Unsigned16

Value description

Sub-index	00$_h$, Hold current delay time
Meaning	Hold current delay time
Access	Read-write
PDO mapping	—
Value range	0$_d$ (off) or 2 to 65535$_d$ (ms)
Default value	500$_d$ (ms)
Category	Yes

8.6.22   2037$_h$ Bridge on to encoder settle time

Defines the delay tim in milliseconds between the device switching into operation enable to resynching the encoder position. Only applicable on Lexium MDrive models equipped with an encoder.

Object description

Index	2037$_h$
Name	Bridge on to encoder settle time
Object code	VAR
Data type	Unsigned16

Value description

Sub-index	00$_h$, Bridge on to encoder settle time
Meaning	Bridge on to encoder settle time
Access	Read-write
PDO mapping	—
Value range	0$_d$ to 3000$_d$ (ms)
Default value	300$_d$ (ms)
Category	Yes

8.6.23   2038$_h$ Trip output configuration

This object configures the functionality of the trip output which will pulse the output upon reaching particular position_demand_effort_position(s).

Values are in Internal Units, based on 51200/rev and not necessarily scaled to user units. Also note trip is activated on position demand effort, and not actual position or encoder position.

This definition of point(s) begins with sub-index 2038.02$_h$ (1st position of a series) then add sub-index 2038.03$_h$ to form the specified number of trip points.

1)   Sub-index 01$_h$: Trip output control: controls the logic and trip points and is used to set up one or multiple trip positions. Note that the trip output must also be selected using `object 2031`$_h$.

2)   Sub-index 02$_h$: 1st trip point of a series. Defines the first trip point of a series of points.

3)   Sub-index 03$_h$: Capture input filter time. This sub-index configures the filtering for the capture input.

4)   Sub-index 04$_h$: Multiple trip point spacing. This sub-index defines the modulus, or spacing between the trip points. Scaled at 51200 steps/rev.

Object description

Index	2038$_h$
Name	Trip output configuration
Object code	REC
Data type	—

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	03$_h$
Category	—

Sub-index	01$_h$, Trip output control
Meaning	Trip output control
Access	Read-write
PDO mapping	—
Value range	Unsigned16
Default value	0000$_h$
Category	Yes

Sub-index 01$_h$ value range		
Bit: 15 (trip enable)	14 ... 12	Bits: 11 ... 0 (number of trip points)
0= disabled, 1=enabled	0 0 0	0=infinite, 1 – 4095=# trip points from start

Sub-index	02$_h$, First trip point
Meaning	First trip point
Access	Read-write
PDO mapping	—
Value range	Integer32
Default value	0000 0000$_h$
Category	Yes

Sub-index	03$_h$, Multiple trip point spacing
Meaning	Multiple trip point spacing
Access	Read-write
PDO mapping	—
Value range	Integer32
Default value	5120$_d$
Category	Yes

8.6.24   2098$_h$ Homing configuration

Determines the position or encoder counter status following a home. If 0 the position or encoder counter will clear following a home. If 0, the position and encoder counter will NOT be assigned, following a homing attained. The exception to this rule is when performing homing method 35.

If 1 (default) the device will subtract the homing offset (`object 607C`$_h$) from the counter and set the counter to the difference.

Object description

Index	2098$_h$
Name	Homing configuration
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, Homing configuration
Meaning	Homing configuration
Access	Read-write
PDO mapping	—
Value range	0/1
Default value	1
Category	Yes

8.6.25　2204$_h$ Run current

This object sets the percentage of full current at which the device will operate.

Object description

Index	2204$_h$
Name	Run current
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, Run current
Meaning	Run current
Access	Read-write
PDO mapping	—
Value range	1 – 100$_d$ (%)
Default value	50 (%)

8.6.26　2205$_h$ Hold current

This object sets the percentage of full current at which the device will transition to when motion ceases.

Object description

Index	2204$_h$
Name	Hold current
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, Hold current
Meaning	Hold current
Access	Read-write
PDO mapping	—
Value range	0 – 100$_d$ (%)
Default value	5 (%)
Category	Yes

8.6.27   2211$_h$ Position present point target

This object contains the position present point target

Object description

Index	2211$_h$
Name	Position present point target
Object code	VAR
Data type	Integer32

Value description

Sub-index	00$_h$, Position present point target
Meaning	Position present point target
Access	Read-only
PDO mapping	—
Value range	$\pm 2^{31}$
Default value	0
Category	—

8.6.28   2212$_h$ Position final point target

This object contains the position final point target

Object description

Index	2212$_h$
Name	Position final point target
Object code	VAR
Data type	Integer32

Value description

Sub-index	00$_h$, Position final point target
Meaning	Position final point target
Access	Read-only
PDO mapping	—
Value range	$\pm 2^{31}$
Default value	0
Category	—

8.6.29   2221$_h$ Following error

This object defines an action to take in the event of a following error (`Idx 6041 bit 13`).

Object description

Index	2221$_h$
Name	Following error
Object code	VAR
Data type	Integer32

Value description

Sub-index	00$_h$, Number of entries
Meaning	Number of entries
Access	Read only
PDO mapping	—
Value range	—
Default value	01$_h$
Category	—

Sub-index	01$_h$, Following error reaction code
Meaning	Following error reaction code
Access	Read-only
PDO mapping	—
Value range	—
Default value	0 - 2
Category	—

Sub-index 01$_h$ value range	
Value	**Action**
0	No action taken
1	Post-move to correct for error after initial move
2	Post-move to correct for error after initial move, the adjust for encoder position

8.6.30   2401$_h$ General purpose user variable

This object is a general purpose user variable which can be used to store 8-bits of data.

Object description

Index	2401$_h$
Name	General purpose user variable
Object code	VAR
Data type	Integer32

Value description	Sub-index	00_h, General purpose user variable
	Meaning	General purpose user variable
	Access	Read-write
	PDO mapping	—
	Value range	$00 - FF_h$
	Default value	0
	Category	—

8.6.31 2504_h SEM options

This object allows interoperability with Schneider Electric Twido PLC's and will not ordinarily be used.

Object description	Index	2504_h
	Name	SEM options
	Object code	VAR
	Data type	Integer32

Value description	Sub-index	00_h, SEM options
	Meaning	SEM options
	Access	Read-write
	PDO mapping	—
	Value range	$00 - FF_h$
	Default value	0
	Category	—

8.6.32 2701_h hMTechnology enable

This object controls the enable/disable state of the hMTechnology control circuitry.

- Disabled = 00_h
- Enabled = 80_h

`Object 2701`$_h$ is only available on Lexium MDrive closed loop models.

Object description	Index	2701_h
	Name	Hybrid enable
	Object code	VAR
	Data type	Unsigned8

Value description	Sub-index	00_h, Hybrid enable
	Meaning	Hybrid enable
	Access	Read-write
	PDO mapping	—
	Value range	00_h or 80_h
	Default value	00_h
	Category	Yes

8.6.33   2702$_h$ hMTechnology configuration

This object sets the parameters of the hMTechnology (hMT)operation. The following parameters are set:

1) **Fixed/variable current (bit 7):** hMT can operate using two current modes: fixed and variable. In fixed current mode the device will operate using the settings of the run (`object 2204`$_h$) and hold (`object 2205`$_h$) current set parameters. In variable current mode the device will adjust the current between the run and hold current settings to what is required to move the load at the desired velocity.

2) **Control boundaries (bits 6 ... 5):** HMT functions by closely monitoring the relationship between the rotor and stator of the motor to within set boundaries from 1.1 to 1.7 motor full steps where a lower setting gives enhanced torque performance and a higher setting deliver batter speed performance.

3) **Make-up mode (bits 1 ... 0):** If enabled, hMT control will make-up for lost steps in the move profile. This can occur at a fixed velocity of 2.5 MHz or at a speed preset using `object 2703`$_h$.

`Object 2701`$_h$ is only avialable on Lexium MDrive closed loop models.

Object description

Index	2702$_h$
Name	hMTechnology configuration
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, hMTechnology configuration
Meaning	hMTechnology configuration
Access	Read-write
PDO mapping	—
Value range	See Table 8.13 (Unsigned8)
Default value	A2$_h$
Category	Yes

Bits	7	6	5	4	3	2	1	0

Function	cur_fv	c_bnds	X	X	X	mu
Current mode (cur_fv)						
Bit 7	0	Fixed current (default)				
	1	Variable current				
Control bounds (c_bnds)						
Bit 6	Bit 5					
0	0	1.1 full steps (best torque performance)				
0	1	1.3 full steps (default)	(Best overall performance)			
1	0	1.5 ful steps				
1	1	1.7 full steps (best speed performance)				
Make-up mode (mu)						
Bit 1	Bit 0					
0	0	Make up disabled				
0	1	reserved				
1	0	Make-up velocity = 2.5 MHz (default)				
1	1	Make-up velocity = 2703_h setting				

Table 8.11: Description of clock options object 2032.01_h

8.6.34 2703_h Make-up velocity

Defines the velocity for hybrid make-up (`Object 2702`$_h$`, bits 1 ... 0`) if selected. The make-up period is determind using the equation:

Frequency = (x+2) * 50ns where x is the setting of 2703_h

e.g. (1998+2) * 50 = 100000 steps/sec

`Object 2703`$_h$ is only avialable on Lexium MDrive closed loop models.

Object description

Index	2703_h
Name	Make-up velocity
Object code	VAR
Data type	Unsigned32

Value description

Sub-index	00_h, Make-up velocity
Meaning	Make-up velocity
Access	Read-write
PDO mapping	—
Value range	Unsigned32
Default value	1998_d (100000 steps/sec)
Category	Yes

8.6.35 2704$_h$ Torque velocity

Defines the maximum velocity for Profile Torqeu mode.

Velocity = 12 Mhz (x+2) where x is the setting of `2703`$_h$

e.g. 12 x 10^6 / (98+2) = 120000 steps/sec @ torque (`Idx 6071`)

`Idx 2704`$_h$ is only avialable on Lexium MDrive closed loop models.

Object description

Index	2704$_h$
Name	Torque velocity
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, Torque velocity
Meaning	Torque velocity
Access	Read-write
PDO mapping	—
Value range	Unsigned8
Default value	98$_d$ (120000 steps/sec)
Category	Yes

8.6.36 2740$_h$ Clear locked rotor

Used to clear the locked rotor status flag. Sending 04$_h$ will clear a locked rotor indication.

Object description

Index	2740$_h$
Name	Clear locked rotor
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, Clear locked rotor
Meaning	Clear locked rotor
Access	Read-write
PDO mapping	Yes, R_PDO
Value range	Unsigned8
Default value	—
Category	Yes

8.6.37   2741$_h$ hMTechnology status (filtered)

Indicates the status of the hMTechnology logic. This object may be filtered using `Idx 2743`. The purpose behing the filter is that the object is PDO mappable. Bit 4, hMT intervening may cycle rapidly between states, if mapped to a PDO this can flood the buss with messages.

7	6	5	4	3	2	1	0
Factory	Reserved	Reserved	hMT intervening	Locked rotor	Reserved	Reserved	Fault

Bit	Value	Meaning
1	0	No error
	1	hMTechnology error state exists
3	0	Rotor normal
	1	Rotor is locked
4	0	hMT not intervening
	1	hMT intervening
7	0	N/A
	1	Factory alignment - This bit will always be 1

`Object 2741`$_h$ is only avialable on Lexium MDrive closed loop models.

Object description

Index	2741$_h$
Name	hMT status
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, hMT status
Meaning	hMT status
Access	Read-only
PDO mapping	Yes, T_PDO
Value range	Unsigned8
Default value	80$_h$
Category	—

8.6.38   2742ₕ hMTechnology status (unfiltered)

Indicates the status of the hMTechnology logic.

7	6	5	4	3	2	1	0
Factory	Reserved	Reserved	hMT intervening	Locked rotor	Reserved	Reserved	Fault

Bit	Value	Meaning
1	0	No error
	1	hMTechnology error state exists
3	0	Rotor normal
	1	Rotor is locked
4	0	hMT not intervening
	1	hMT intervening
7	0	N/A
	1	Factory alignment - This bit will always be 1

`Object 2742`$_h$ is only avialable on Lexium MDrive closed loop models.

Object description

Index	2742$_h$
Name	hMT status
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, hMT status
Meaning	hMT status
Access	Read-only
PDO mapping	—
Value range	Unsigned8
Default value	80$_h$
Category	—

8.6.39   2743$_h$ hMTechnology status byte filter

Filters the status byte of the hMTechnology logic.

7	6	5	4	3	2	1	0
Factory	Reserved	Reserved	hMT intervening	Locked rotor	Reserved	Reserved	Fault
1	1	1	0	1	1	1	1

`Object 2743`$_h$ is only avialable on Lexium MDrive closed loop models.

Object description

Index	2743$_h$
Name	hMT status filter
Object code	VAR
Data type	Unsigned8

Value description

Sub-index	00$_h$, hMT status filter
Meaning	hMT status filter
Access	Read-write
PDO mapping	—
Value range	Unsigned8
Default value	EF$_h$
Category	—

8.7   Details of object group 5000$_h$ (Mfg factory specific)

Object group 5000$_h$ contains objects factory configuration and are not for end use.

8.8   Details of assignment objects group 6000$_h$

The objects in group 6000$_h$ are operation specific. See Section 6 for detailed information on these objects.

9 CANopen configurator

9

The software associated with Lexium MDrive CANopen products is contained within the Lexium MDrive Software Suite. This software package is available for download at the Schneider Electric Motion USA web site at http://motion.schneider-electric.com.

The modules applicable are:

1. CANopen configuration interface: Graphic User Interface (GUI) for commissioning the Lexium MDrive

See the Lexium MDrive Software Suite manual for installation and usage instructions.

WARRANTY

Reference the web site at www.motion.schneider-electric.com for the latest warranty and product information.

USA SALES OFFICES

East Region
Tel. 610-573-9655
e-mail: e.region@imshome.com

Northeast Region
Tel. 860-368-9703
e-mail: n.region@imshome.com

Central Region
Tel. 630-267-3302
e-mail: c.region@imshome.com

Western Region
Tel. 602-578-7201
e-mail: w.region@imshome.com

EUROPEAN SALES MANAGEMENT

Tel. +33/4 7256 5113 – Fax +33/4 7838 1537
e-mail: europe.sales@imshome.com

TECHNICAL SUPPORT

Tel. +00 (1) 860-295-6102 – Fax +00 (1) 860-295-6107
e-mail: etech@imshome.com