CANOpen Reference Manual

Ver.2 Rel.01/10

Release	Note
Version 1 Release 05/05	First edition.
Version 2 Release 06/06	First release.
Version 2 Release 07/06	New chapters. Corrections.
Version 2 Release 09/06	Insert chapter 6: "Appendix". Corrections.
Version 2 Release 05/09	New objects inserted. Emcy message modified. Interpolated Position Mode inserted. Corrections.
Version 2 Release 06/09	Homing methods 33 and 34 inserted. Corrections.
Version 2 Release 01/10	Corrections.

Print in Italy 01/2010

AXOR INDUSTRIES
Via Stazione N°5
36054 Montebello Vi.no (VI)
Italy
Tel. +39 0444 440441
Fax. +39 0444 440418
E-mail sales@axorindustries.com
Internet: http://www.axorindustries.com

Index

1. CanOpen Protocol

2. Communication Objects

3. Network Management (NMT)

4. Object Dictionary

5. Device Protocol

6. Appendix

Chapter 1
CanOpen Protocol

1.1 - Introduction

CanOpen protocol is one of the most common CAN protocols. Since 1985 the CanOpen specifications are managed by the international users and constructors group called CAN in Automation (CiA). European Standards Authorities have accepted the CanOpen specification 4.01 as EN 50325-4.
This chapter describes the basic services and Communication Objects of the CanOpen communication profile DS301, which are used in the Axor's drives.

CAN-based networks use the following *reference model*, *device model* and *communication model*.

1.2 - Reference model

The communication concept can be described similar to the *ISO-OSI Reference Model*:

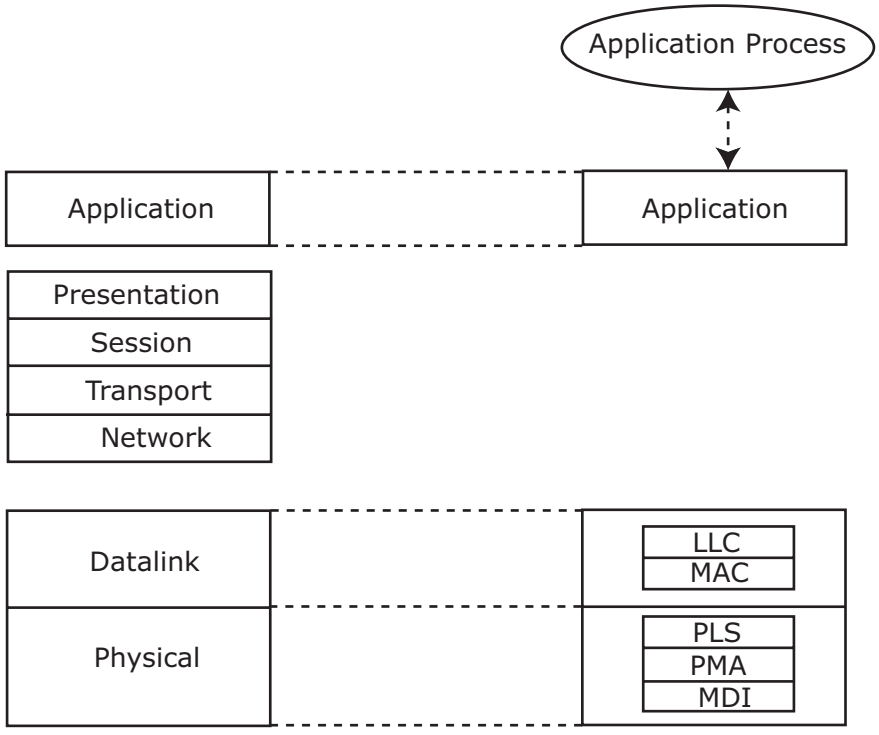


Figure1.1: Reference Model

This reference model is based on seven Layer protocols.
The Layer-1/2 protocol (Physical Layer/Data Link Layer) that is implemented in all CAN modules provides, amongst other things, the requirements for data.

Data transport or data request is made by means of a data telegram (*Data Frame*) with up to 8 bytes of user data, or by a data request telegram (*Remote Frame*).

Communication Objects are labeled by an 11-bit Identifier (ID) that olso determines the priority of Objects.

A Layer-7 protocol (*Application Layer*) was developed to decouple the application from the communication.

Applications interact by invoking *services* of a *service object* in the application layer. To realise these services, this object exchanges data via the CAN Network with a peer service object(s) via a protocol.
A *service type* defines the primitives that are exchenged between the application layer and the cooperating applications for a particular service of a service object. There are four possible service types:

- *Local Service*: the appplication isses a request to its local service object that executes the requested service without communicating with the (a) peer service object(s).
- *Unconfirmed Service*: the appplication isses a request to its local service object. The request is transferred to the peer service object(s) that each pass it to their application as an indication. The result is not confirmed back.
- *Confirmed Service*: the appplication isses a request to its local service object. The request is transferred to the peer service object that each passes it to the other application as an indication. To other application issues a response that is transferred to the originating service object that passes it as a confirmation to the requesting application.
- *Provider Initiated*: the service object detects an event not solicited by a request service. This event is then indicated to the application.

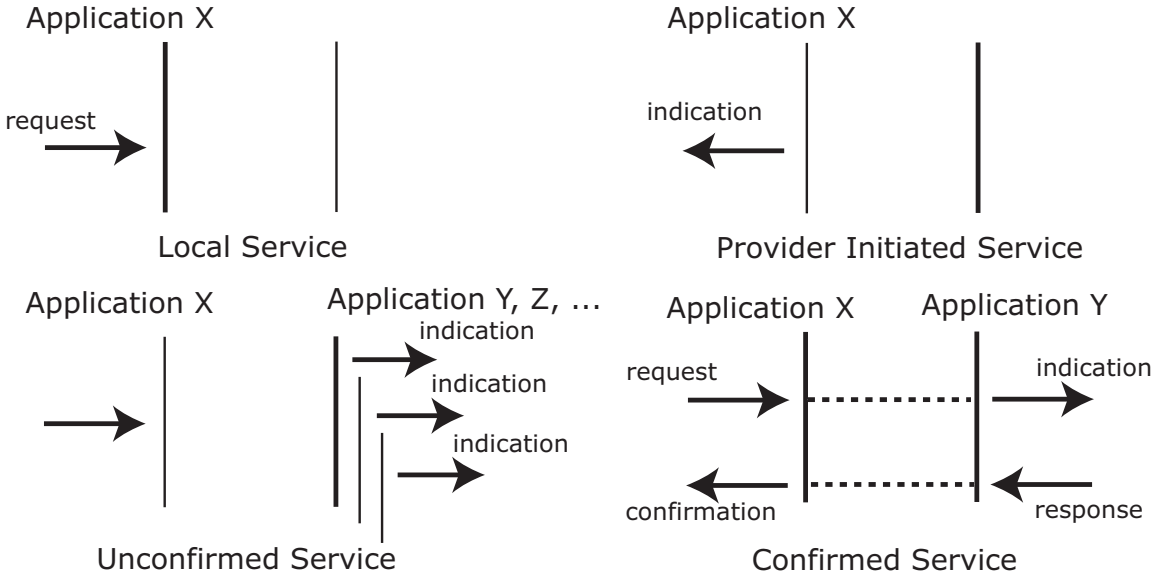


Figure1.2: Service Types

1.3 - Device Model

1.3.1 - General

A *device* is structured as follows (see Figure 1.3):

• *Communication* – This function unit provides the communication objects and the appropriate functionality to transport data items via the underlying network structure.

• *Object Dictionary* – The Object Dictionary is a collection of all the data items which have an influence on the behaviour of the application objects, the communication objects and the state machine used on this device.

• *Application* – The application comprises the functionality of the device with respect to the interaction with the process environment.

Thus the Object Dictionary serves as an interface between the communication and the application. The complete description of a device's application with respect to the data items in the Object Dictionary is named *device profile*.

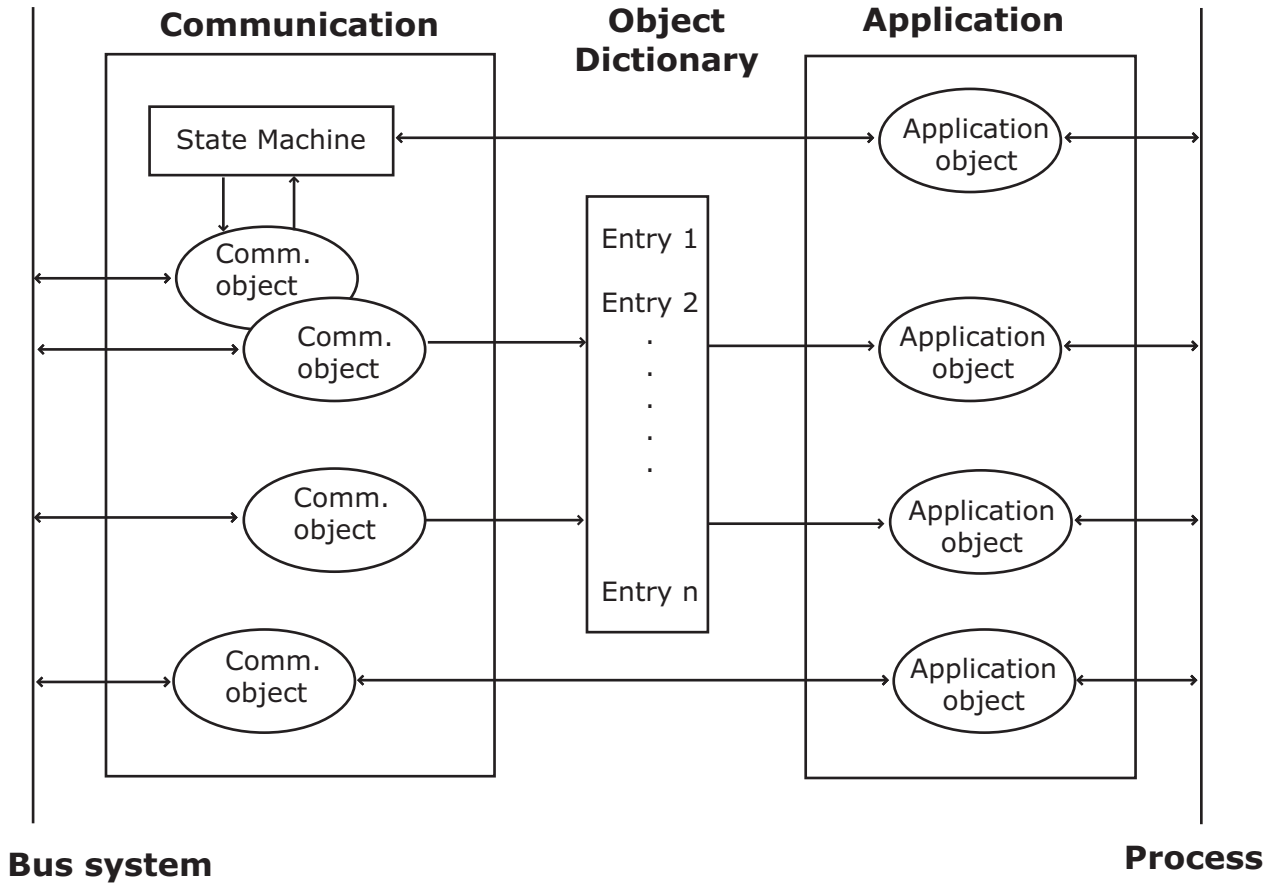


Figure1.3: Device Model

1.3.2 - Object Dictionary

The most important part of a device profile is the *Object Dictionary* description. The Object Dictionary is essentially a grouping of objects accessible via the network in an ordered pre-defined fashion. Each object within the dictionary is addressed using a **16-bit index**.
The overall layout of the standard Object Dictionary is shown below. This layout closely conforms with other industrial serial bus system concepts:

INDEX	Object
0000	Not used
0001 - 001F	Static Data Types
0020 - 003F	Complex Data Typed
0040 - 005F	Manufacturer Specific Complex Data Types
0060 - 007F	Device Profile Specific Static Data Types
0080 - 009F	Device Profile Specific Complex Data Types
00A0 - 0FFF	Reserved for further use
1000 - 1FFF	Communication Profile Area
2000 - 5FFF	Manufacturer Specific Profile Area
6000 - 9FFF	Standardised Device Profile Area
A000 - BFFF	Standardised Interface Profile Area
C000 - FFFF	Reserved for further use

Table1.1: Object Dictionary Structure

In case of a *simple variable* the index references the value of this variable directly. In case of *records* and *arrays* however, the index addresses the whole data structure. To allow individual elements of data structures to be accessed via the network a **sub-index** is defined. For single Object Dictionary entries, such as an UNSIGNED8, BOOLEAN, INTEGER 32 etc, the value for the sub-index is always zero. For complex Object Dictionary entries, such as arrays or records with multiple data fields, the sub-index references fields within a data-structure pointed to by the main index. The fields accessed by the sub-index can be of differing data types.

Data Types used by the Object Dictionary are:

- INTEGER8
- UNSIGNED8
- INTEGER16
- UNSIGNED16
- INTEGER32
- UNSIGNED32

To transmit numerical values, a bit sequence is ordered by byte and then transmitted starting from the LSB.

Examples:

Numerical Value 15432 = $3C48_h$ (INTEGER16)

B0	B1
$3C48_h$	

⇨

B0	B1
48_h	$3C_h$

Numerical Value 894989336 = 35587418h (INTEGER32)

B0	B1	B2	B3
35587418			

⇨

B0	B1	B2	B3
18	74	58	35

1.4 - Communication Model

The communication model specifies the different communication objects and services and the available modes of messagge transmissioning triggering.

The communication model supports the transmission of synchronous and asynchronous messages. Synchronous messages are transmetted with respect to a pre-defined synchronisation message, while asynchronous messages may be transmetted at any time.

It is possible define *inhibit times* for the communication. The inhibit time of a data object defines the minimum time that has to elapse between consecutive invocations of a transmission services for that data object.

It is possible to distinguished three types of communication relationships:

- *Master/Slave relationship*;
- *Client/Server relationship*;
- Producer/Consumer relationship.

1.4.1 - Master/Slave relationship

At any time there is exactly one device in the network serving as a master, while all others devices are considered as slaves. The master issues a request and the addressed slave(s) reapond(s) if the protocol requires this behavior.

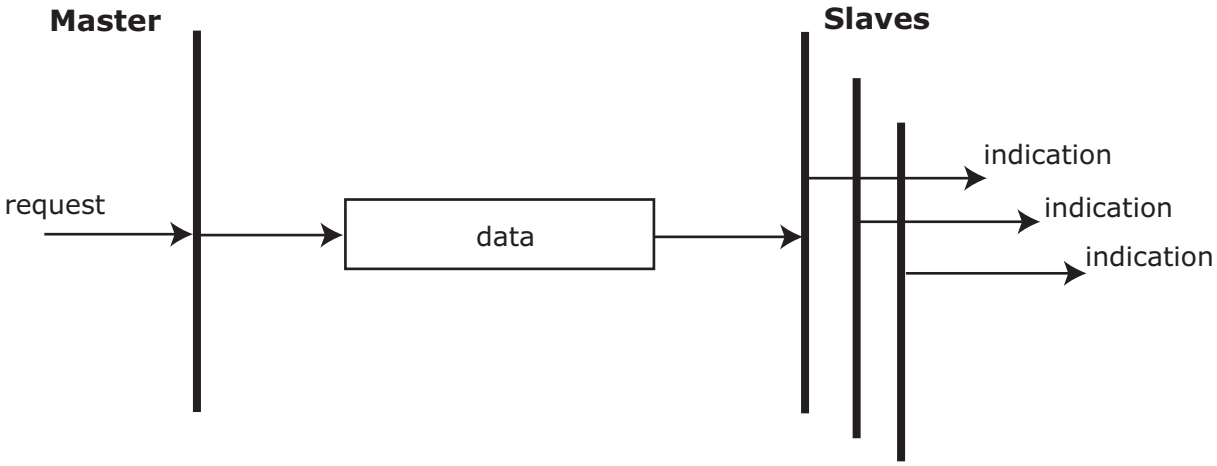


Figure1.3: Unconfirmed Master Slave Communication

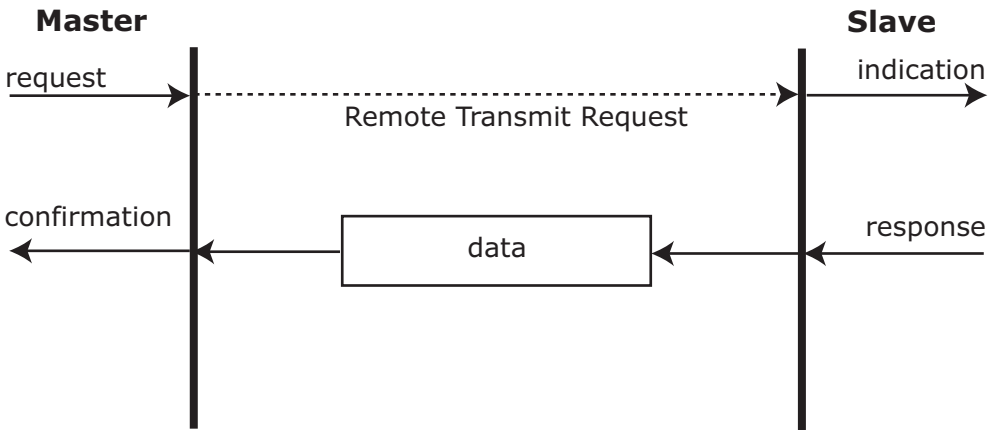


Figure1.5: Confirmed Master Slave Communication

1.4.2 - Client/Server relationship

This is a relationship between a single client and a single server. A client issues a request (upload/download) thus triggering the server to perform a certain task. After finishing the task the server answer the request.

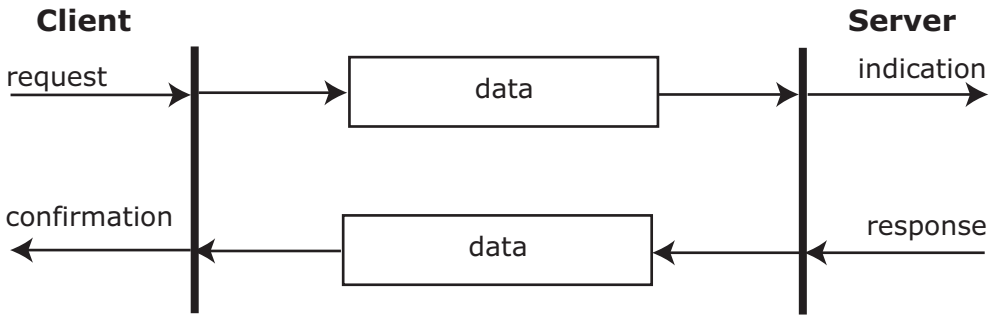


Figure1.6: Client/Server Communication

1.4.3 - Producer/Consumer relationship - Pull/Push model

The producer/consumer relationship model involves a producer and zero or more consumer(s). The push model is characterized by an unconfirmed services request by the producer. The pull model is characterized by a confirmed service requested by the consumer.

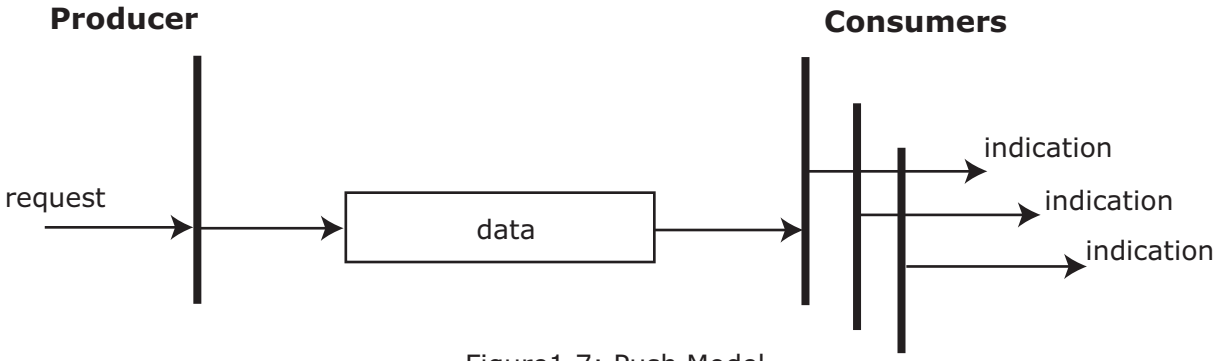


Figure1.7: Push Model

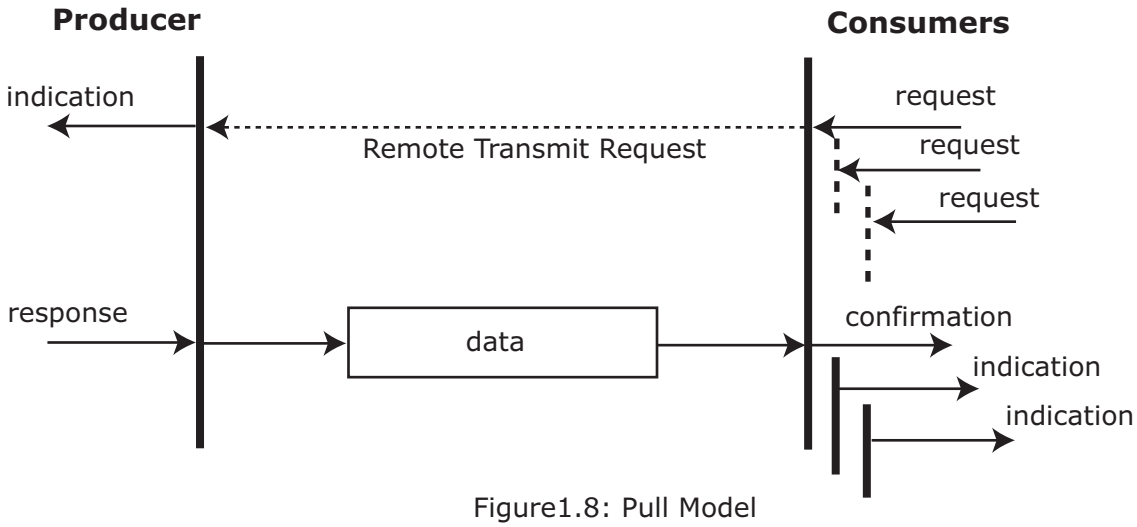


Figure1.8: Pull Model

1.5 - Communication Objects

Communication Objects are described with the help of *service elements* and *protocols*. Axor's drive supports Communication Objects which are described in the next chapters:

• ***SDO*** stands for *Service Data Object* and is a protocol with confirmed transaction for exchanging Object Dictionary dates between master and slave. Usually a slave device is a SDO server while a master device is a SDO client. That means that a Slave can only answer to a Masters interrogation. Often this protocol is used to configure device internal parameters. Being a confirmed service, this service generates a big quantity of traffic on the network. That is the reason for which it is not recommended for high speed and real time communication.

• ***PDO*** stands for *Process Data Object* and is a highly configurable protocol with unconfirmed transaction for exchanging dates in high-velocity and real time applications. This is the protocol that maximises CAN architecture's advantages. PDO transfer is made without overloading the net. PDOs correspond to specific Object Dictionary entries of the device and supply the interface for the application parameters. PDO mapping is determined by the corresponding PDO Map Object in the Object Dictionary. PDOs can be synchronous or asynchronous.

• The ***SYNC*** *generator* (usually the master) periodically sends the synchronisation object. This SYNC object supplies the network base timer. There could be a temporary uncertainty in transmission on the part of the SYNC that corresponds approximately to the latency due to another COB transmitted slightly before the SYNC. In order to guarantee access at the proper instance on CAN-bus, a very high priority identification is attributed to the SYNC.

• *Emergency Objects* (***EMCY***) are transmitted by the Slave Device when an internal error has occurred. EMCY are useful to signal occasional errors or alarms.

• The *Network Management* (***NMT***) is built following a Master-Slave structure. NMT parameters are only used to execute NMT services. Using NMT nodes can be initialised, stopped, monitored, reset, or stopped. Each node has its own ID included in the range [1, …, 127]. One and only one node can be the NMT master. All other nodes are NMT slaves and are univocally identified by their own ID.

• *Time Stamp Object (**TIME**)* provides to devices a common time frame reference.

Basic Structure

The *basic structure* of a Communication Object is the followg:

S O M	COB-ID	R T R	CTRL	Data Segment	CRC	A C K	EOM

SOM	Start message
COB-ID	COB Identifier (11-bit)
RTR	Remote Transmission Request
CTRL	Control Field
Data Segment	0...8 byte (Data-CBO)
	0 byte (Data-CBO)
CRC	Cyclic Redundancy Check
ACK	Acknowlenge slot
EOM	End of message

Pre-defined connection set

In order to reduce configuration effort for simple networks a mandatory default *identifier allocation scheme* is defined. It consists of a *functional part*, which determines the object priority and a *Node-ID-part*, which allows to distinguish between devices of the same functionality (see Figura 1.9).

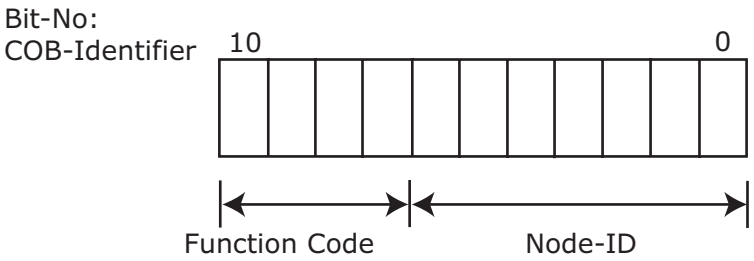


Figure 1.9: Identifier allocation scheme for the pre-defined connection set

This allows a peer-to-peer communication between a single master device and up to 127 slave devices. It also supports the broadcasting of non-confirmed NMT-objects, SYNC- and TIME STAMP-objects. Broadcasting is indicated by a Node-ID of zero.
The pre-defined connection set supports one emergency object, one SDO, at maximum 4 Receive-PDOs and 4 Transmit-PDO and the NMT objects.
The following tables show the supported objects and their allocated COB-IDs.

Object	Function Code (binary)	resulting COB-ID	Communication Parameters at Index
NMT	0000	0	-
SYNC	001	128 (80h)	1006h
TIME STAMP	0010	256 (100h)	-

Table 1.2: Broadcast Objects of the Pre-defined Connection Set

Object	Function Code (binary)	resulting COB-IDs	Communication Parameters at Index
EMERGENCY	0001	129 (81h) - 255 (FFh)	-
PDO1 (tx)	0011	385 (181h) - 511(1FFh)	1800h
PDO1 (rx)	0100	513 (201h) - 639 (27Fh)	1400h
PDO2 (tx)	0101	641 (281h) - 767 (2FFh)	1801h
PDO2 (rx)	0110	769 (301h) - 895 (37Fh)	1401h
PDO3 (tx)	0111	897 (381h) - 1023 (3FFh)	1802h
PDO3 (rx)	1000	1025 (401h) - 1151 (47Fh)	1402h
PDO4 (tx)	1001	1153 (481h) - 1279 (4FFh)	1803h
PDO4 (rx)	1010	1281 (501h) - 1407 (57Fh)	1403h
SDO (tx)	1011	1409 (581h) - 1535 (5FFh)	1200h
SDO (rx)	1100	1537 (601h) - 1663 (67Fh)	1200h
NMT Error Control	1110	1793 (701h) - 1919 (77Fh)	1017h

Table 1.3: Peer-to-Peer Objects of the Pre-defined Connection Set

The following COB-ID are for restricted use and they have not to be used as COB-ID by any configurable communication object, neither for SYNC, TIME-STAMP, EMCY, PDO and SDO.

COB-ID	Used by object
0 (000h)	NMT
1 (001h)	reserved
257 (101h) - 384 (180h)	reserved
1409 (581h) - 1535 (5FFh)	default SDO (tx)
1537 (601h) - 1663 (67Fh)	default SDO (rx)
1760 (6E0h)	reserved
1793 (701h) - 1919 (77Fh)	NMT Error Control
2020 (780h) - 2047 (7FFh)	reserved

Table 1.4: Restricted COB-IDs

Chapter 2
Communication Objects

2.1 – Introduction
The Communication Objects are described by *services* and *protocols*.

2.2 – Process Data Object (PDO)
The PDOs are used for real-time data communication. The transfer of PDOs is performed with no protocol overhead, in fact these communication objects use the "unconfirmed communication service".
There are two types of PDOs, depending on the direction of transmission:
1- **Transmit-PDO** (TPDO): Axor's drive ⇨ Master;
2- **Receive-PDO** (RPDO): Master ⇨ Axor's drive.
PDOs are described by the *PDO communication parameter* (20h, which describes the communication capabilities of the PDO) and the *PDO mapping parameter* (21h, which contains information about the contents of the PDOs).
The indices of the corresponding Object Dictionary entries are computed by the following formulas:
- RPDO communication parameter index = 1400h + RPDO-number-1
- TPDO communication parameter index = 1800h + TPDO-number-1
- RPDO mapping parameter index = 1600h + RPDO-number-1
- TPDO mapping parameter index = 1A00h + TPDO-number-1

For each PDO the pair of communication and mapping parameter is mandatory.

2.2.1 - Transmission Modes
The following PDO *transmission modes* are distinguished:
- *Synchronous Transmission*
- *Asynchronous Transmission*

In Figure 2.1 the principle of the synchronous and asynchronous transmission is shown.

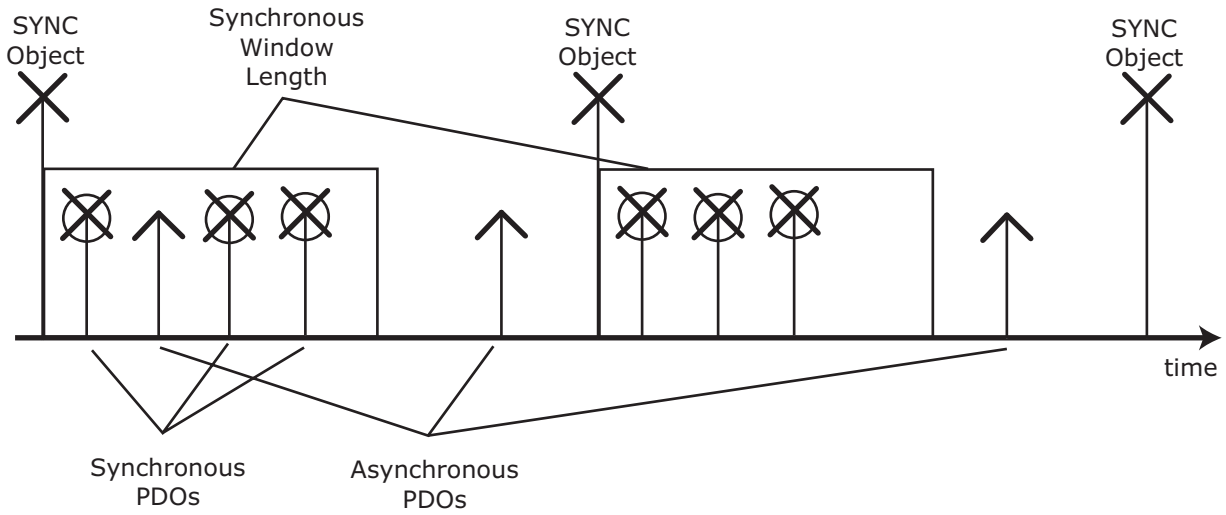


Figure 2.1: Synchronous and Asynchronous Transmission

Synchronous PDOs are transmitted within a pre-defined time-window immediately after the SYNC Object (which is a synchronisation object transmitted periodically by a synchronisation application).
The *transmission type parameter* of a PDO specifies the transmission mode. A transmission type of "0" means that the message shall be transmitted after occurrence of the SYNC but acyclic (not periodically), only if an event occurred before the SYNC. A transmission type of "1" means that the message is transmitted with every SYNC object. A transmission type of "n" means that the message is transmitted with every n-th SYNC object.
Asynchronous TPDOs are transmitted without any relation to a SYNC.
The *data of synchronous RPDOs* received after the occurrence of a SYNC is passed to the application with the occurrence of the following SYNC, independent of the transmission type parameter. The *data of asynchronous RPDOs* is passed directly to the application.

2.2.2 - PDO Services

The PDO transmission follows the *Producer/Consumer relationship*.
The PDOs have the following *attributes*:

- PDO number: PDO number [1...512] for every user type on the local device
- user type: one of the values {consumer, producer}
- data type: according to the PDO mapping
- inhibit time: n*100μs, n≥0

Write PDO Service

For the Write PDO service the *Push Model* is valid, in fact the service for a *PDO write* request is unconfirmed. A PDO has exactly one producer and zero or more consumers.
Through this service the producer of a PDO sends the data of the mapped application objects to the consumer(s).

Parameter	*Request/Indication*
Argument	**Mandatory**
PDO Number	mandatory
Data	mandatory

Table 2.1: Write PDO

Read PDO Service

For the Read PDO service the *Pull Model* is valid, in fact the service for a *PDO read* request is confirmed. PDO has exactly one producer and one or more consumers.
Through this service the consumer of a PDO requests the producer to supply the data of the mapped application objects.

Parameter	*Request/Indication*	*Response/Confirm*
Argument	**Mandatory**	
PDO Number	mandatory	
Remote Result		**Mandatory**
Data		mandatory

Table 2.2: Read PDO

2.2.3 - PDO Protocols

Write PDO Protocol

The Figure 2.2 shows the *Write PDO Protocol*: the PDO producer sends the process data within a PDO to the network.

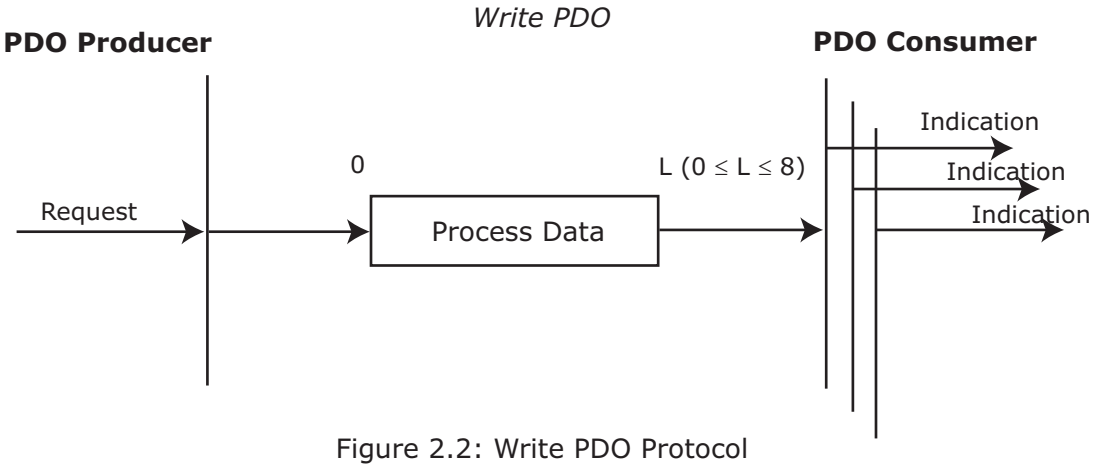


Figure 2.2: Write PDO Protocol

Process-Data: up to L bytes of application data according to the PDO mapping. If L exceeds the number of bytes "n" defined by the actual PDO mapping length, only the first "n" bytes are used by the consumer. If L is less than "n", the data of the received PDO is not processed and an Emergency message with error code 8210h has to be produced if Emergency is supported.

Read PDO Protocol

The Figure 2.3 shows the *Read PDO Protocol*: one or more PDO consumers transmit a remote transmission request frame (RTR) to the network. At the reception of the RTR frame the PDO producer for the requested PDO transmits the PDO.

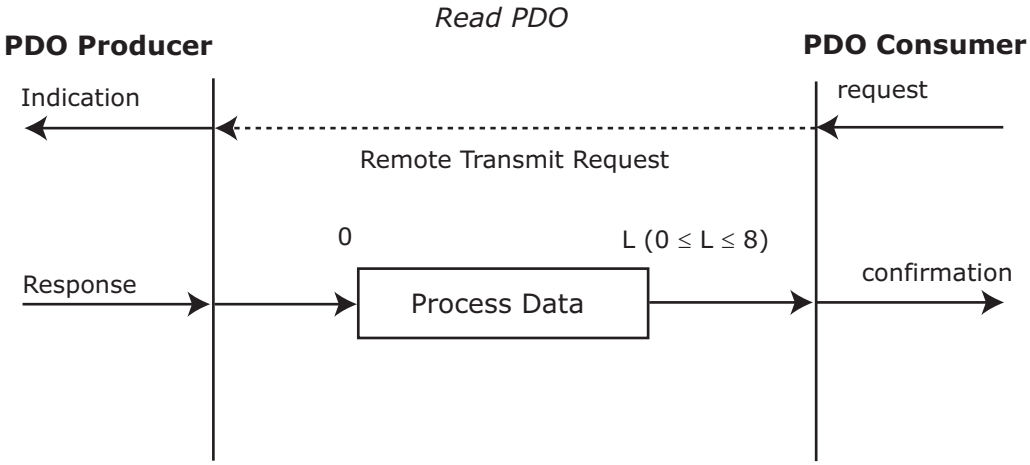


Figure 2.3: Read PDO Protocol

Process-Data: up to L bytes of application data according to the PDO mapping. If L exceeds the number of bytes "n" defined by the actual PDO mapping length, only the first "n" bytes are used by the consumer. If L is less than "n", the data of the received PDO is not processed and an Emergency message with error code 8210h has to be produced if Emergency is supported.

2.3 – Service Data Object (SDO)

The Service Data Objects (SDOs) provide the access to entries of a device Object Dictionary. The SDOs can be used to transfer *multiple data sets* (each containing an arbitrary large block of data) from a client to a server and vice-versa. The client can control via a *multiplexor* (index and sub-index of the Object Dictionary) which data set is to be transferred. The contents of the data set are defined within the Object Dictionary.
Basically a SDO is transferred as a *sequence of segments*. Prior to transferring the segments there is an *initialisation phase* where client and server prepare themselves for transferring the segments.

For SDOs, it is also possible to transfer a data set of up to four bytes during the initialisation phase. This mechanism is called an *expedited transfer.*

Optionally a SDO can be transferred as a *sequence of blocks* where each block is a sequence of up to 127 segments containing a sequence number and the data. Prior to transferring the blocks there is an initialisation phase. After transferring the blocks there is a finalisation phase where client and server can optionally verify the correctness of the previous data transfer by the comparing checksum derived from the data set. This transfer type mentioned above is called a *block transfer* which is faster than the segmented transfer for a large set of data.
In SDO Block Upload it is possible that the size of the data set does not justify the use of a block transfer because of the implied protocol overhead. In this cases a support for a fallback to the segmented or expedited transfer in initialisation phase can be implemented.
For the block transfer a Go-Back-n ARQ (Automatic Repeat Request) scheme is used to confirm each block.
After *block download* the server indicates the client the last successfully received segment of this block transfer by acknowledging this segment sequence number. The client has to start the following block transfer with the re-transmission of all not acknowledged data. Additionally the server has to indicate the number of segments per block for the next block transfer.
After *block upload* the server indicates the client the last successfully received segment of this block transfer by acknowledging this segment sequence number. The client has to start the following block transfer with the retransmission of all not acknowledged data. Additionally the client has to indicate the number of segments per block for the next block transfer.

For all transfer types it is the client that takes the initiative for a transfer. The owner of the accessed Object Dictionary is the server of the SDO. Both the client or the server can take the initiative to abort the transfer of a SDO.

By means of a SDO a peer-to-peer communication channel between two devices is established. A device may support more than one SDO.

SDOs are described by the SDO *communication parameter record* (22h, which describes the communication capabilities of the Server-SDOs and Client-SDOs).
The indices of the corresponding Object Dictionary entries are computed by the following formulas:

- SSDO communication parameter index
- CSDO communication parameter index

For each SDO the communication parameters are mandatory. If only one Server-SDO exists the communication parameters can be omitted.

2.3.1 - SDO Services

The SDO communication follows the *Client/Server Model*.
The SDOs have the following *attributes*:

- SDO number: SDO number [1...128] for every user type on the local device
- user type: one of the values {client, server}
- mux data type multiplexor containing index and sub-index of type STRUCTURE OF UNSIGNED(16), UNSIGNED(8), with index specifying an entry of the device Object Dictionary and "sub-index" specifying a component of a device object dictionary entry
- transfer type: depends on the length of data to transfer:
  - *expedited* for up to 4 data bytes
  - *segmented* or *block* for more than 4 data bytes
- data type: according to the referenced index and sub-index

The following *services* can be applied onto a SDO depending on the application requirements:

- **SDO Download**, which can be split up into:
  - Initiate SDO Download
  - Download SDO Segment

- **SDO Upload**, which can be split up into:
  - Initiate SDO Upload
  - Upload SDO Segment

- **Abort SDO Transfer**

When using the segmented SDO download and upload services, the communication software will be responsible for transferring the SDO as a sequence of segments.
Expedited transfer has to be supported. *Segmented transfer* has to be supported if objects larger than 4 bytes are supported.

Optionally the following SDO services for doing a block transfer with higher bus utilisation and performance for large data set size can be implemented:

- **SDO Block Download**, which can be split up into:
  - Initiate Block Download
  - Download Block
  - End Block Download

- **SDO Block Upload**, which can be split up into:
  - Initiate Block Upload
  - Upload Block
  - End Block Upload

When using the SDO block download and upload services, the communication software will be responsible for transferring the SDO as a sequence of blocks.
In SDO Block Upload Protocol a support for a switch to SDO Upload Protocol in "Initiate SDO Block Upload" can be implemented to increase transfer performance for data which size does not justify using the protocol overhead of the "SDO Block Upload" protocol.

For aborting a SDO block transfer the *Abort SDO Transfer Service* is used.

Axor's drives supports the following services:

SDO Download
Through this service the client of a SDO downloads data to the server (owner of the Object Dictionary).
The service is confirmed.

Parameter	*Request/Indication*	*Response/Confirm*
Argument	**Mandatory**	
SDO Number	mandatory	
Data	mandatory	
Size	optional	
Multiplexor	mandatory	
Remote Result		**Mandatory**
Data		selection
Failure		selection
Reason		optional

Table 2.3: SDO Download

Initiate SDO Download
Through this service the client of a SDO requests the server to prepare for downloading data to the server.
The service is confirmed.

Parameter	*Request/Indication*	*Response/Confirm*
Argument	**Mandatory**	
SDO Number	mandatory	
Size	optional	
Multiplexor	mandatory	
Transfer Type	mandatory	
Normal	selection	
Expedeted	selection	**Mandatory**
Data	mandatory	mandatory
Remote Result		
Succeess		

Table 2.4: Initiate SDO Download

SDO Upload
Through this service the client of a SDO uploads data from the server.
The service is confirmed.

Parameter	*Request/Indication*	*Response/Confirm*
Argument	**Mandatory**	
SDO Number	mandatory	
Multiplexor	mandatory	
Remote Result		**Mandatory**
Success		selection
Data		mandatory
Size		optional
Failure		selection
Reason		optional

Table 2.5: SDO Upload

Initiate SDO Upload
Through this service the client of a SDO requests the server to prepare for uploading data to the client.
The service is confirmed.

Parameter	*Request/Indication*	*Response/Confirm*
Argument	**Mandatory**	
SDO Number	mandatory	
Multiplexor	mandatory	
Remote Result		**Mandatory**
Succeess		mandatory
Size		optional
Multiplexor		mandatory
Transfer Type		mandatory
Normal		selection
Expedeted		selection
Data		mandatory

Table 2.6: Initiate SDO Upload

Abort SDO Transfer
This service aborts the up- or download of a SDO referenced by its number. Optionally the reason is indicated. The service is unconfirmed. The service may be execute at any time by both the client and the server of a SDO.

Parameter	*Request/Indication*
Argument	**Mandatory**
SDO Number	mandatory
Multiplexor	mandatory
Reason	mandatory

Table 2.7: Abort SDO Transfer

2.3.2 - SDO Protocols

Six confirmed services (SDO Download, SDO Upload, Initiate SDO Download, Initiate SDO Upload, Download SDO Segment, Upload SDO Segment) and one unconfirmed service (Abort SDO Transfer) are defined for Service Data Objects doing the standard segmented/ expedeted transfer.
Eight confirmed services (SDO Block Download, SDO Block Upload, Initiate SDO Upload, Initiate Block Download, Download SDO Segment, Upload SDO Segment, End SDO Upload and End SDO Block Download) and one unconfirmed service (Abort SDO Block Transfer) are defined for Service Data Objects doing the optional block transfer.

Axor's drives supports the following protocols:

Download SDO Protocols

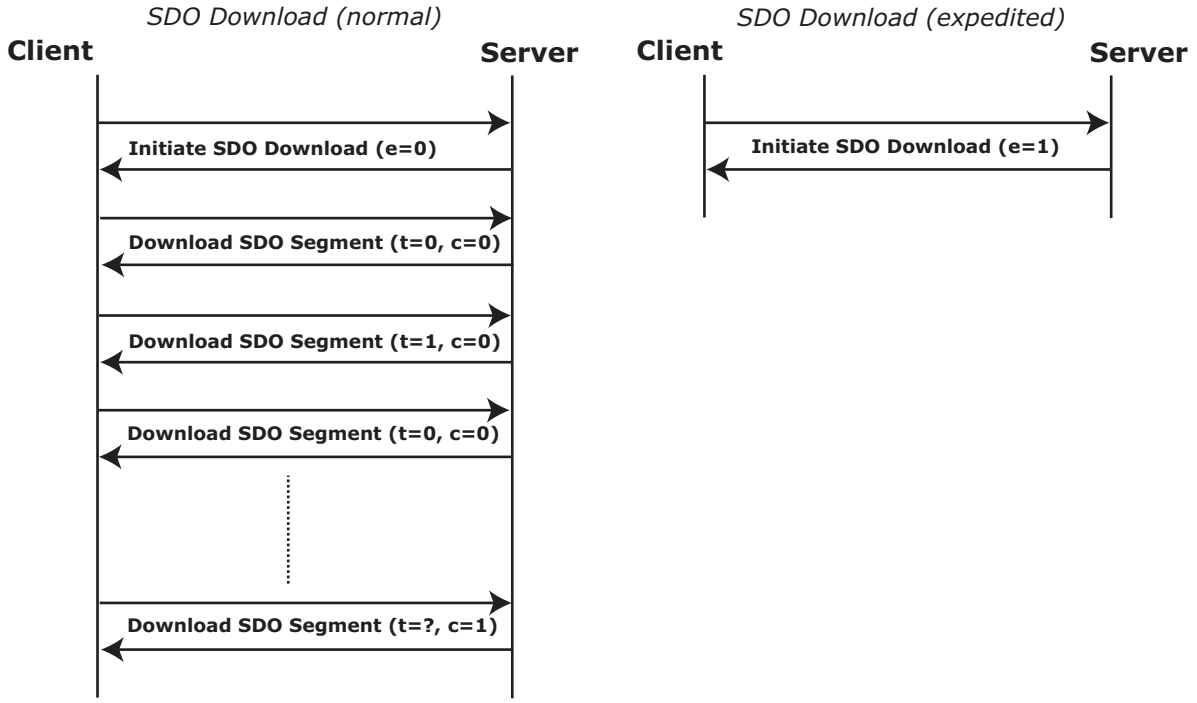


Figure 2.4: Download SDO Protocol

This protocol is used to implement the SDO Download service. SDOs are downloaded as a sequence of zero or more Download SDO Segment services preceded by an Initiate SDO Downloaded service.
The sequence is terminated by:

- an Initiate SDO Download request/indication with the e-bit set to 1 followed by an Initiate SDO Download response/confirm, indicating the successful completion of an expedited download sequence.
- a Download SDO Segment response/confirm with c-bit set to 1, indicating the successful completion of a normal download sequence.
- an Abort SDO Transfer request/indication, indicating the unsuccessful completion of the download sequence.
- a new Initiate Domain Download request/Indication, indicating the unsuccessful completion of the download sequence and the start of a new download sequence.

If in the download of two consecutive segments the toggle bit does not alter, the contents of the last segments has to be ignored. If such an error is reported to the application, the application may decide to abort the download.

Initiate SDO Download Protocol

This protocol is used to implement the Initiate SDO Download service for SDOs.

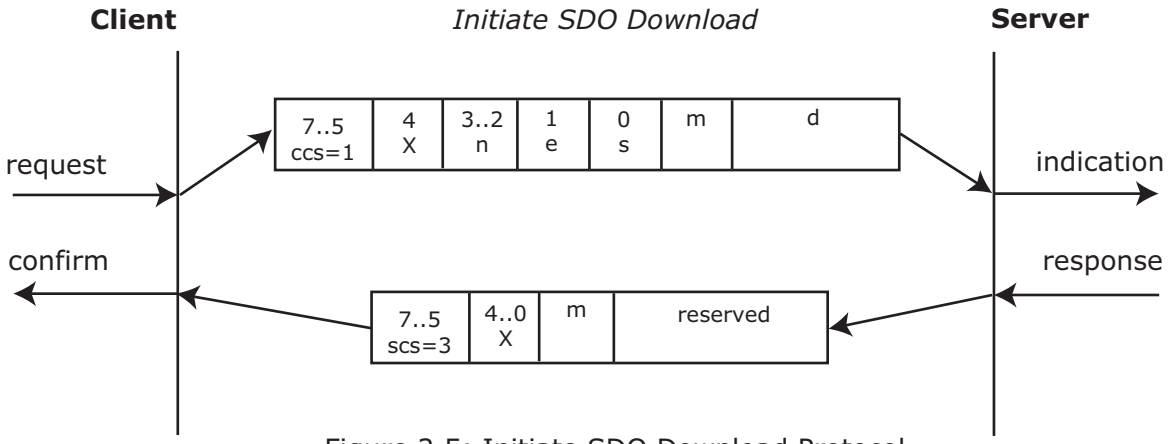


Figure 2.5: Initiate SDO Download Protocol

- **ccs**: client command specifier
  1: initiate download request
- **scs**: server command specifier
  3: initiate download response
- **n**: only valid if e =1 and s=1, otherwise 0. If valid, it indicates the number of bytes in d that do not contain data. Bytes [8-n, 7] do not contain data.
- **e**: transfer type
  0: normal transfer
  1: expedited transfer
- **s**: size indicator
  0: data set size is not indicated
  1: data set size is indicated
- **m**: multiplexor. It represents the index/sub-index of the data to be transfer by the SDO.
- **d**: data
  e=0, s=0: d is reserved for further use.
  e=0, s=1: d contains the number of bytes to be downloaded. Byte 4 contains the lsb and byte 7 contains the msb.
  e=1, s=1: d contains the data of length 4-n to be downloaded, the encoding depends on the type of the data referenced by index and sub-index.
  e=1, s=0: d contains unspecified number of bytes to be downloaded.
- **X**: not used, always 0
- **reserved**: reserved for further use, always 0

Upload SDO Protocol

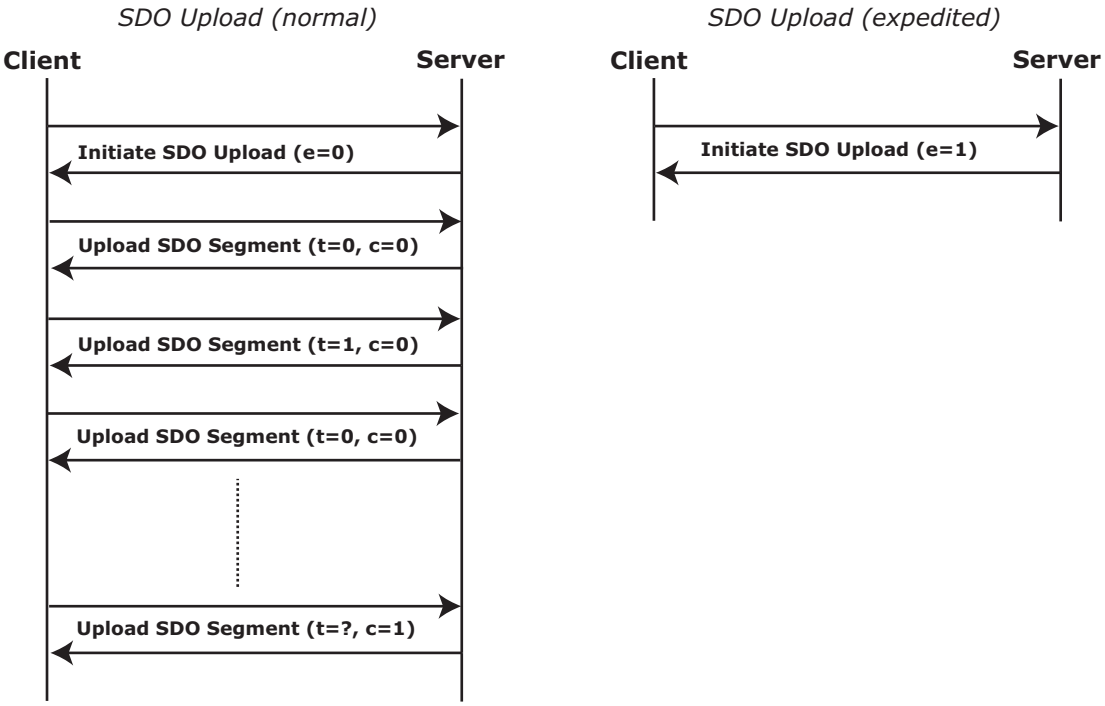


Figure 2.6: Upload SDO Protocol

This protocol is used to implement the SDO Upload service. SDOs are uploaded as a sequence of zero or more Upload SDO Segment services preceded by an Initiate SDO Uploaded service.
The sequence is terminated by:

- an Initiate SDO Upload response/confirm with the e-bit set to 1, indicating the successful completion of an expedited upload sequence.
- an Upload SDO Segment response/confirm with c-bit set to 1, indicating the successful completion of a normal upload sequence.
- an Abort SDO Transfer request/indication, indicating the unsuccessful completion of the upload sequence.
- a new Initiate SDO Upload request/indication, indicating the unsuccessful completion of the upload sequence and the start of a new download sequence.

If in the upload of two consecutive segments the toggle bit does not alter, the contents of the last segments has to be ignored. If such an error is reported to the application, the application may decide to abort the upload.

Initiate SDO Upload Protocol

This protocol is used to implement the Initiate SDO Upload service.

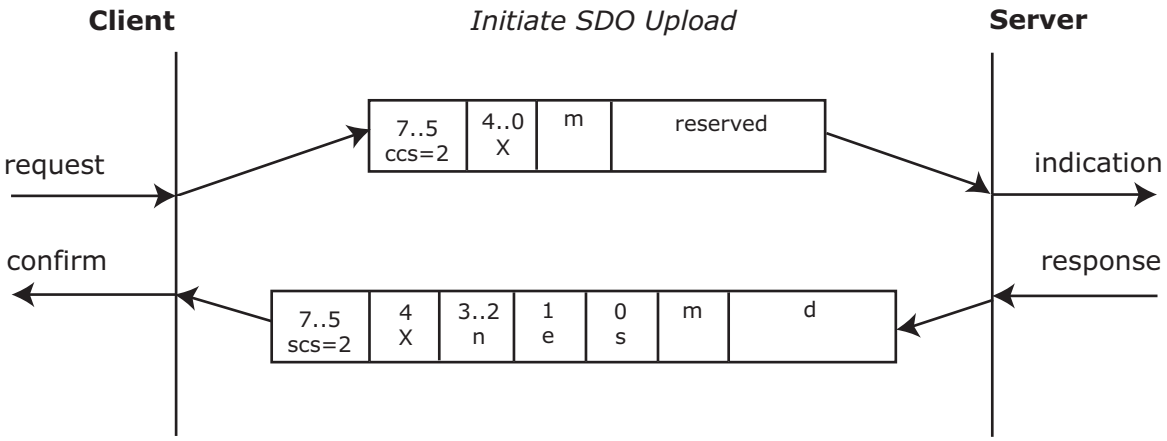


Figure 2.7: Initiate SDO Upload Protocol

- **ccs**: client command specifier
  2: initiate upload request
- **scs**: server command specifier
  2: initiate upload response
- **n**: only valid if e =1 and s=1, otherwise 0. If valid, it indicates the number of bytes in d that do not contain data. Bytes [8-n, 7] do not contain segment data.
- **e**: transfer type
  0: normal transfer
  1: expedeted transfer
- **s**: size indicator
  0: data set size is not indicated
  1: data set size is indicated
- **m**: multiplexor. It represents the index/sub-index of the data to be transfer by the SDO.
- **d**: data
  e=0, s=0: d is reserved for further use.
  e=0, s=1: d contains the number of bytes to be uploaded.
  Byte 4 contains the LSB and byte 7 contains the MSB.
  e=1, s=1: d contains the data of length 4-n to be uploaded, the encoding depends on the type of the data referenced by index and sub-index.
  e=1, s=0: d contains unspecified number of bytes to be uploaded.
- **X**: not used, always 0
- **reserved**: reserved for further use, always 0.

Abort SDO Transfer Protocol

This protocol is used to implement the Abort SDO Transfer Service.

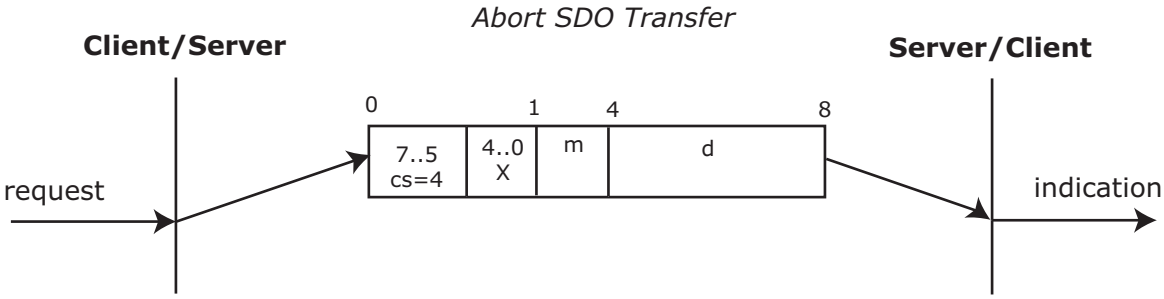


Figure 2.8: Abort SDO Transer Protocol

- **cs**: command specifier
  4: abort transfer request
- **X**: not used, always 0
- **m**: multiplexor. It represents index and sub-index of the SDO.
- **d**: contains a 4 byte abort code about the reason for the abort.

The abort code is encoded as UNSIGNED32 value.

Abort Code	Description
0503 0000h	Toggle bit not alternated.
0504 0000h	SDO protocol timed out.
0504 0001h	Client/Server command specifier not valid or unknown.
0504 0002h	Invalid block size (block mode only).
0504 0003h	Invalid sequence number (block mode only).
0504 0004h	CRC error (block mode only).
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object,.
0601 0001h	Attempt to read a write only object.
0601 0002h	Attempt to write a read only object.
0602 0000h	Object does not exist in the Object Dictionary.
0604 0041h	Object cannot be mapped to the PDO.
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility reason.
0604 0047h	General internal incompatibility in the device.
0606 0000h	Access failed due to an hardware error.
0607 0010h	Data type does not match, length of service parameter does not match.
0607 0012h	Data type does not match, length of service parameter too high.
0607 0013h	Data type does not match, length of service parameter too low.
0609 0011h	Sub-index does not exist.
0609 0030h	Value range of parameter exceeded (only for write access).

0609 0031h	Value of parameter written too high.
0609 0032h	Value of parameter written too low.
0609 0036h	Maximum value is less than minimum value.
0800 0000h	General error.
0800 0020h	Data cannot be tranferred or stored to the application.
0800 0021h	Data cannot be tranferred or stored to the application because of local control.
0800 0022h	Data cannot be tranferred or stored to the application because of presence device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error).

Table 2.8: SDO Abort Codes

2.4 – Synchronisation Object (SYNC)

The *Synchronisation Object* is broadcasted periodically by the SYNC producer. It provides the basic network clock. The time period between the SYNCs is specified by the standard parameter **communication cycle period** (Object 1006h).
In order to guarantee timely access to the CAN bus the SYNC is given a very high priority Identifier.

2.4.1 - SYNC Services

The SYNC transmission follows the *Producer/Consumer Push Model*. The service is unconfirmed.
The SYNC objects have the following *attributes*:

- user type: one of the values {consumer, producer}
- data type: none

2.4.2 - SYNC Protocol

One unconfirmed service is defined: **Write SYNC**.

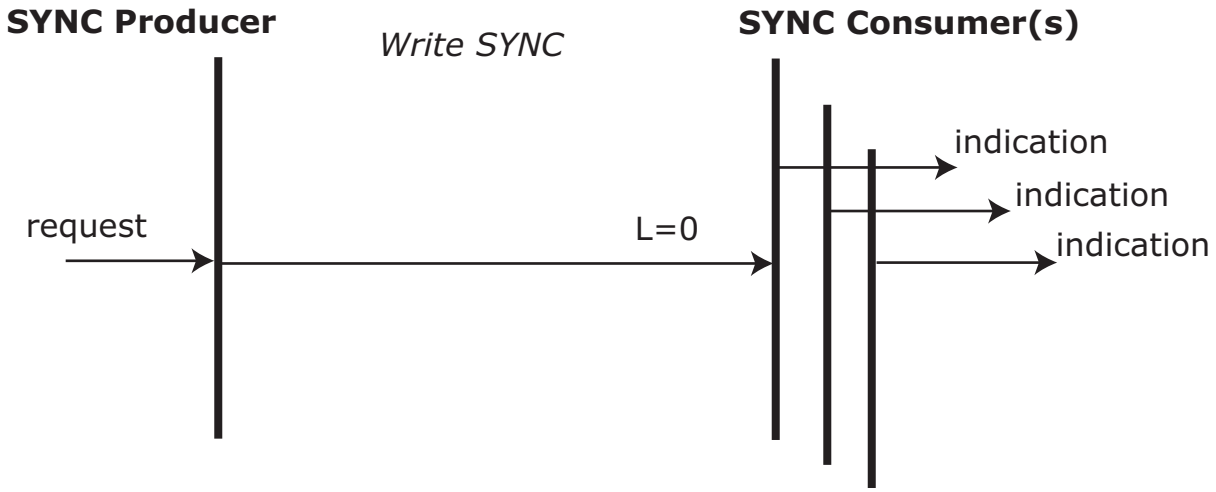


Figure 2.9: SYNC Protocol

The SYNC does not carry any data (L=0).

2.5 – Time Stamp Object (TIME)

By means of the Time Stamp Object a common time frame reference is provided to devices. It contains a value of the type TIME_OF_DAY.

2.5.1 - TIME Services

The Time Stamp Object transmission follows the *Producer/Consumer Push Model*. The service is unconfirmed.
The Time Stamp Objects have the following *attributes*:

- user type: one of the values {consumer, producer}
- data type: TIME_OF_DAY

2.5.2 - TIME Protocol

One unconfirmed service is defined: **Write TIME**.

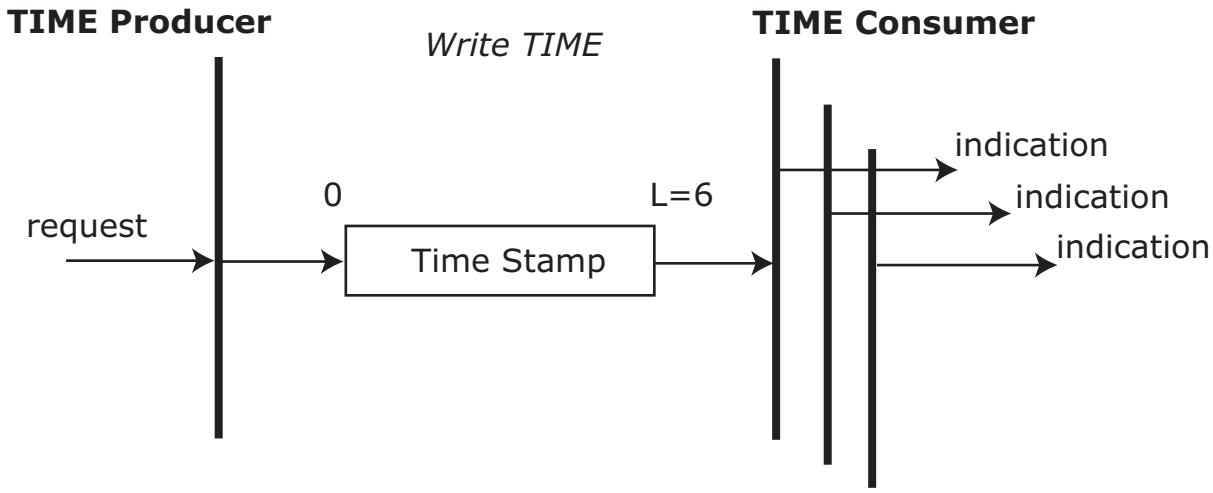


Figure 2.10: TIME Protocol

Time Stamp: 6 bytes of the Time Stamp Object.

2.6 – Emergency Object (EMCY)

2.6.1 - Emergency Object Usage

Emergency objects are triggered by the occurrence of a *device internal error situation* and are transmitted from an emergency producer on the device. An emergency object is transmitted only once per "error event" and it may be received by zero or more emergency consumers.

The emergency object is *optional*. If a device supports the emergency object, it has to support at least the two error codes: 00xx (Error Reset or No Error) and 10xx (Generic Error), while all other error codes are optional.
A device may be in one of two *emergency states* illustrated in Figure 2.11:

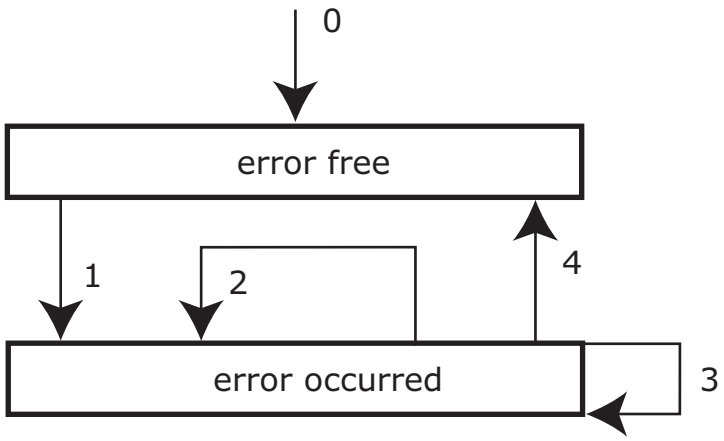


Figure 2.11: Emergency State Transition Diagram

0. After initialization the device enters the error free state if *no error is detected*. No error message is sent.

1. If the *device detects an internal error*, indicated in the first three bytes of the emergency message (error code and error register), it enters the error state. An emergency object with the appropriate error code and error register is transmitted.

2. *One, but not all error reasons are gone*. An emergency message containing error code 0000 (Error reset) may be transmitted together with the remaining errors in the error register and in the manufacturer specific error field.

3. *A new error occurs on the device*. The device remains in error state and transmits an emergency object with the appropriate error code. The new error code is filled in at the top of the array of the error codes (1003h).

4. *All errors are repaired*. The device enters the error free state and transmits an emergency object with the error code "reset error/no error".

2.6.2 - Emergency Object Data

The *Emergency Telegram* consists of 8 bytes with the data as shown in Table 2.9:

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
Emergency Error Code (see Table 2.10)		Error Register (Object 1001h) (see Table 2.11)		Alarm LO		Alarm HI	

Table 2.9: Emergency Data Object

Emergency Error Code (hex)	Meaning
0x5530	Eeprom alarm
0x2310	Over current
0x4200	Drive temperature
0x7300	Hall alarm
0x7305	Encoder alarm
0x2300	I2t drive
0x4201	Motor temperature
0x3000	Regenerative resistance
0x3200	Min/Max temperature
0x3001	Pre-alarm recovery
0x7303	Resolver temperature
0x8611	Following error
0x8000	Limit switch
0x4202	Printed Board temperature
0x2200	Overcurrent brake
0x7000	Mechanical brake
0x3002	In-rush bus
0x3003	Auxiliary voltage
0x5520	Flash alarm
0x8130	CanBus alarm
0xFF00	Homing error
0xFF07	Nstop alarm
0xFF08	Pstop alarm

Table 2.10: Emergency Error Codes

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Manifacturer specific	Reserved	Device profile error	Communication error	Temperature error	Voltege error	Current error	Generic error

Table 2.11: Error Register

2.6.3 - Emergency Object Service

Emergency object transmission follows the *Producer/Consumer Push Model*.
The emergency objects have the following *attributes*:

- user type: notifying device: producer
  receiving devices: consumer
- data type: STRUCTURE OF
  UNSIGNED(16) emergency_error_code,
  UNSIGNED(8) error_register,
  ARRAY(5) of UNSIGNED(8) manufacturer_specific_error_field.
- inhibit time: Application specific

2.6.4 - Emergency Object Protocol

One unconfirmed service is defined: **Write EMCY**.

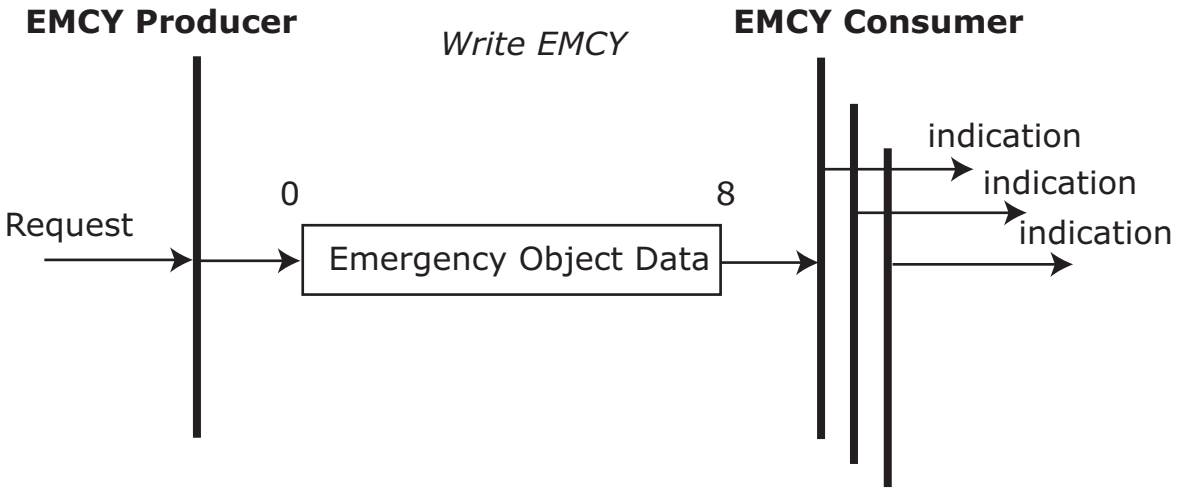


Figure 2.12: Emergency Object Protocol

It is not allowed to request an Emergency Object by a remote transmission request (RTR).

Chapter 3
Network Management (NMT)

3.1 - NMT State Machine

3.1.1 – Initialisation Procedure

In Figure 3.1 the general flow chart of the network initialisation process, controlled by a NMT Master, is shown:

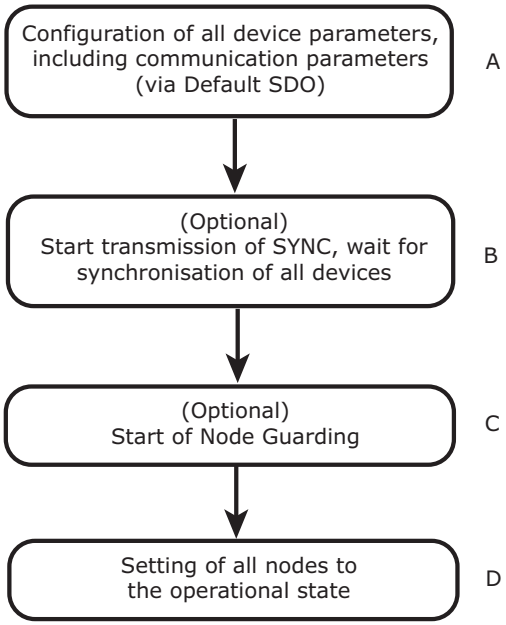


Figure 3.1: Flow Chart of the Network Initialisation Process

In step A the devices are in the node state PRE-OPERATIONAL which is entered automatically after power-on. In this state the devices are accessible via their Default-SDO using identifiers that have been assigned according to the Predefined Connection Set. In this step the configuration of device parameters takes place on all nodes which support parameter configuration.
This is done from a Configuration Application or Tool which resides on the node that is the client for the default SDOs. For devices that support these features the selection and/or configuration of PDOs, the mapping of application objects (PDO mapping), the configuration of additional SDOs and optionally the setting of COB-Ids may be performed via the Default-SDO objects.
In many cases a configuration is not even necessary as default values are defined for all application and communication parameters.
If the application requires the synchronisation of all or some nodes in the network, the appropriate mechanism can be initiated in the optional Step B. It can be used to ensure that all nodes are synchronised by the SYNC object before entering the node state OPERATIONAL in step D. The first transmission of SYNC object starts within 1 sync cycle after entering the PRE-OPERATIONAL state. In Step C Node Guarding can be activated (if supported) using the guarding parameters configured in step A.
With step D all nodes are enabled to communicate via their PDO Objects.

3.1.2 – Overview

In Figure 3.2 the state diagram of a device is shown. Devices enter the PRE-OPERATIONAL state directly after finishing the device initialisation. During this state device it is possible, using SDOs, parameter setting and ID-allocation. Then the nodes can be switched directly into the OPERATIONAL state.
The NMT state machine determines the behaviour of Communication function unit. The coupling of the application state machine to the NMT state machine is device dependent and falls into the scope of device profiles.

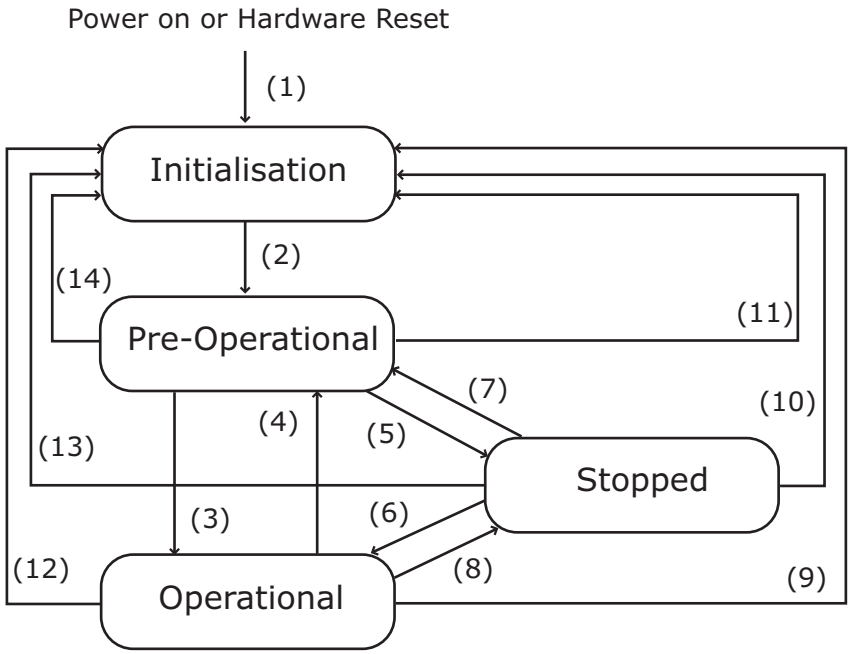


Figure 3.2: State Diagram of a Device

(1)	At Power on the initialisation state is entered autonomously
(2)	Initialisation finished - enter PRE-OPERATIONAL automatically
(3), (6)	Enter OPERATIONAL state indication
(4), (7)	Enter PRE-OPERATIONAL state indication
(5), (8)	Stop Remote Node indication
(9), (10), (11)	Reset Node indication
(12), (13), (14)	Reset Communication indication

Table 3.1: Trigger for State Transition

3.1.3 – States

Initialisation
The "initialisation" state is divided into three sub-states (see Figure 3.3) in order to enable a complete or partial reset of a node.

- Initialising: This is the first sub-state the device enters after power-on or hardware reset. After finishing the basic node initialisation the device enters autonomously into the state "Reset Application".
- Reset Application: In this state the parameters of the manufacturer specific profile area and of the standardised device profile area are set to their power-on values. After setting of the power-on values the state "Reset Communication" is entered autonomously.
- Reset Communication: In this state the parameters of the communication profile area are set to their power-on values. After this the state Initialisation is finished and the device executes the write boot-up object service and enters in the state PRE-OPERATIONAL.

Power-on values are the last stored parameters. If storing is not supported or has not been executed, the power-on values are the default values according to the communication and device profile specifications.

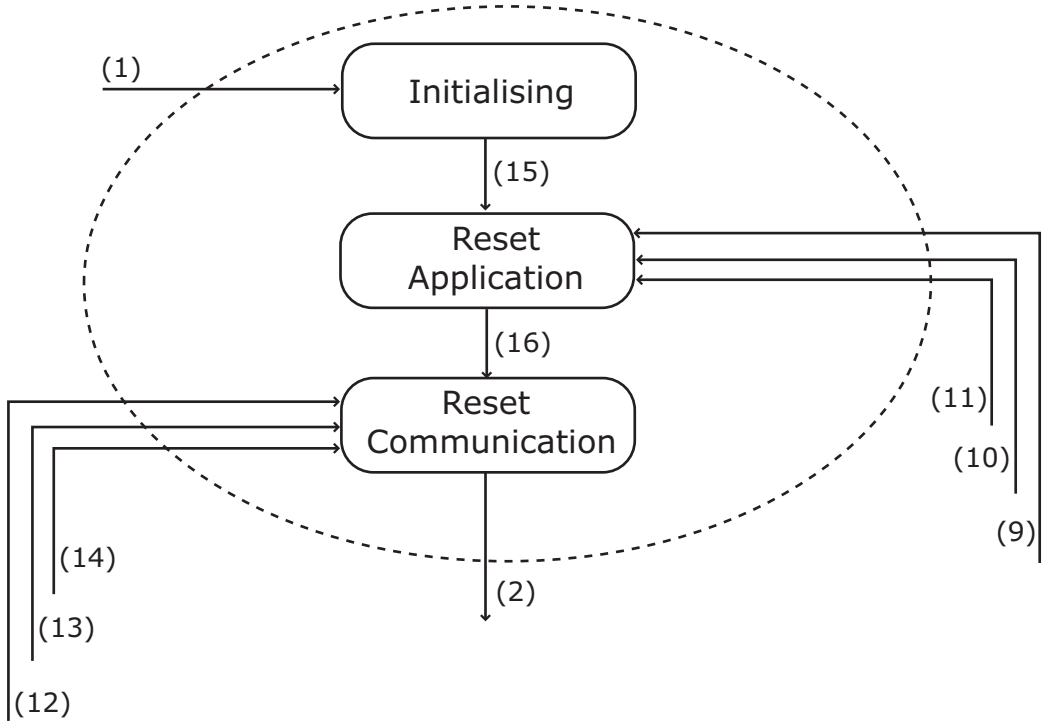


Figure 3.3: Structure of the Initialisation State

(1)	At Power on the initialisation state is entered autonomously
(2)	Initialisation finished - enter PRE-OPERATIONAL automatically
(9), (10), (11)	Reset Node indication
(12), (13), (14)	Reset Communication indication
(15)	Initialisation finished - Reset Application state is entered automatically
(16)	Reset Application is finished - Reset Application state is entered automatically

Table 3.2: Structure of the Initialisation State

Pre-Operational
In the PRE-OPERATIONAL state, communication via SDOs is possible. PDOs do not exist, so PDO communication is not allowed. Configuration of PDOs, device parameters and also the allocation of application objects (PDO mapping) may be performed by a configuration application. The node may be switched into the operational state directly by sending a Start Remote Node request.

Operational
In the OPERATIONAL state all communication objects are active. Transitioning to OPERATIONAL creates all PDOs; the constructor uses the parameters as described in the Object Dictionary. Object Dictionary Access via SDO is possible. Implementation aspects or the application state machine however may require to limit the access to certain objects while the state is operational.

Stopped
A device, switched into the STOPPED state, is forced to stop the communication all together (except node guarding, if active). Furthermore, this state can be used to achieve certain application behaviour.

Table 3.3 shows the relation between communication states and communication objects. Services on the listed communication may only be executed if the device involved in the communication are in the appropriate communication states.

		STATES			
		INITIALISING	PRE-OPERATIONAL	OPERATIONAL	STOPPED
Communication Objects	PDO			X	
	SDO		X	X	
	Synchronisation Object		X	X	
	Time Stamp Object		X	X	
	Emergency Object		X	X	
	Boot-Up Object	X			
	NMT Objects		X	X	X

Table 3.3: States and Communication Objects

State transition are caused by:
- Reception of an NMT object used for module control services;
- Hardware Reset;
- Module Control Services locally initiated by application events.

3.2 – Network Management Objects

The Network Management (NMT) is node oriented and follows a master-slave structure. NMT objects are used for executing NMT services. Through NMT services, nodes are initialised, started, monitored, resetted or stopped. NMT requires that one device in the network fulfils the function of the NMT Master; all other nodes are regarded as NMT slaves. A NMT slave is uniquely identified in the network by its Node-ID, a value in the range of [1,…,127].

3.2.1 – Module Control Services

Through Module Control Services, the NMT master controls the state of the NMT slaves. The state attribute is one of the following values:

- INITIALISING
- PRE-OPERATIONAL
- OPERATIONAL
- STOPPED

The Module Control Services can be performed with a certain node or with all nodes simultaneously. The NMT master control its own NMT state machine via local services. The Module Control Services are initialised locally also by NMT slaves.

Start Remote Node

Through this service the NMT Master sets the state of the selected NMT Slaves to OPERATIONAL.

Parameter	*Indication - Request*
Argument	**Mandatory**
Node-ID	selection
All	selection

Table 3.4: Start Remote Node

The service is unconfirmed and mandatory. After completion of the service, the state of the selected remote nodes will be OPERATIONAL. Broadcasting is indicated by a Node-ID of zero.

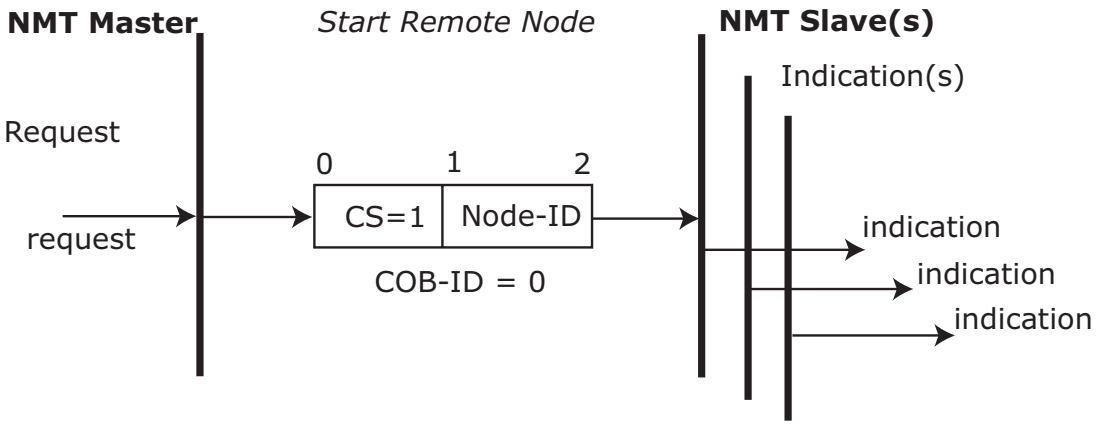


Figure 3.4: Start Remote Node Protocol

Stop Remote Node

Through this service the NMT Master sets the state of the selected NMT Slaves to STOPPED.

Parameter	*Indication - Request*
Argument	**Mandatory**
Node-ID	selection
All	selection

Table 3.5: Start Remote Node

The service is unconfirmed and mandatory. After completion of the service, the state of the selected remote nodes will be STOPPED. Broadcasting is indicated by a Node-ID of zero.

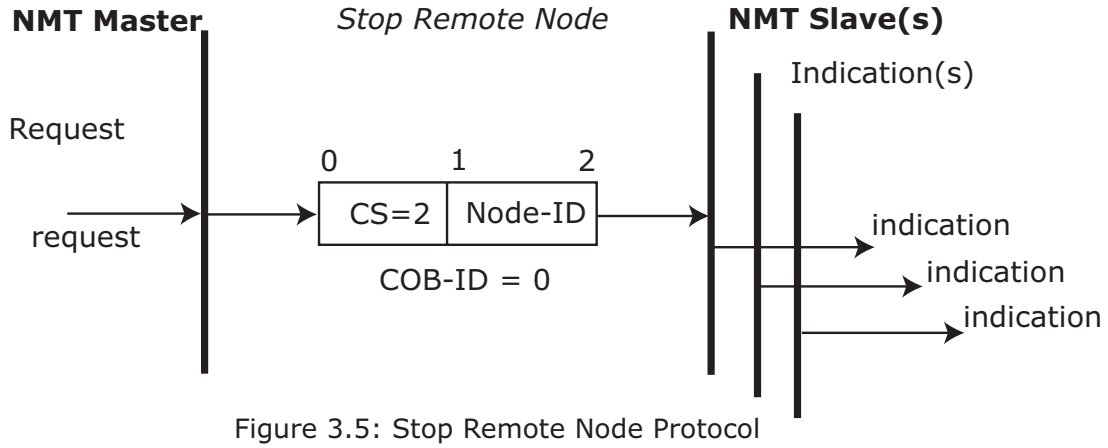


Figure 3.5: Stop Remote Node Protocol

Enter Pre-Operational

Through this service the NMT Master sets the state of the selected NMT Slaves to PRE-OPERATIONAL.

Parameter	*Indication - Request*
Argument	**Mandatory**
Node-ID	selection
All	selection

Table 3.6: Enter Pre-Operational

The service is unconfirmed and mandatory. After completion of the service, the state of the selected remote nodes will be PRE-OPERATIONAL. Broadcasting is indicated by a Node-ID of zero.

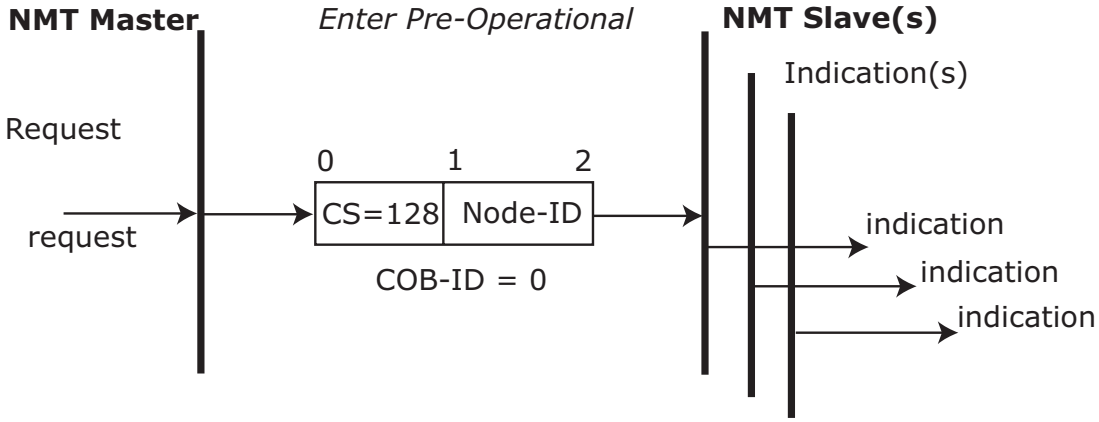


Figure 3.6: Enter Pre-Operational Protocol

Reset Node

Through this service the NMT Master sets the state of the selected NMT Slaves from any state to the "RESET APPLICATION" sub-state.

Parameter	*Indication - Request*
Argument	**Mandatory**
Node-ID	selection
All	selection

Table 3.7: Reset Node

The service is unconfirmed and mandatory. After completion of the service, the state of the selected remote nodes will be RESET APPLICATION. Broadcasting is indicated by a Node-ID of zero.

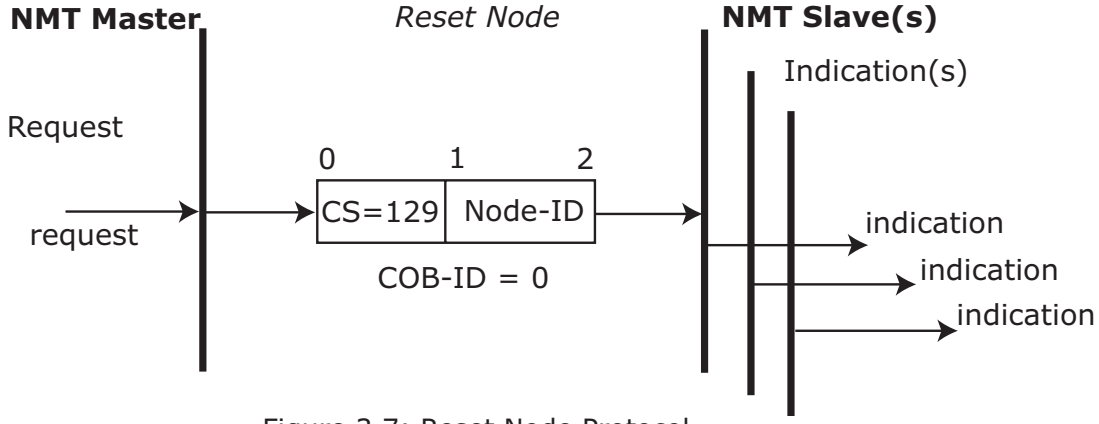


Figure 3.7: Reset Node Protocol

Reset Communication

Through this service the NMT Master sets the state of the selected NMT Slaves from any state to the "RESET COMMUNICATION" sub-state.

Parameter	*Indication - Request*
Argument	**Mandatory**
Node-ID	selection
All	selection

Table 3.8: Reset Communication

The service is unconfirmed and mandatory. After completion of the service, the state of the selected remote nodes will be RESET COMMUNICATION. Broadcasting is indicated by a Node-ID of zero.

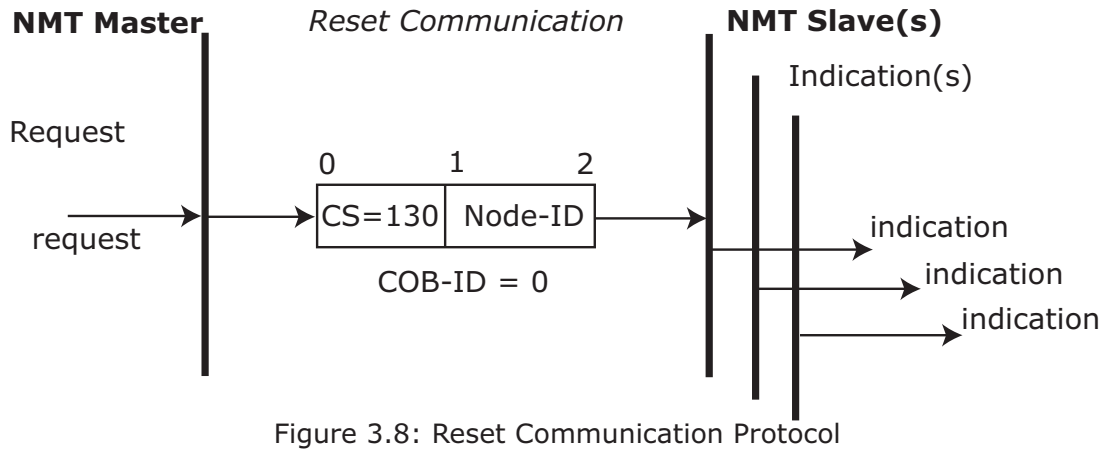


Figure 3.8: Reset Communication Protocol

3.3 – Error Control Protocols

3.3.1 – Node Guarding Protocol

This protocol is used to detect remote errors in the network. Each NMT Slave uses one remote COB for the Node Guarding Protocol. This protocol implements the provider initiated Error Control services.

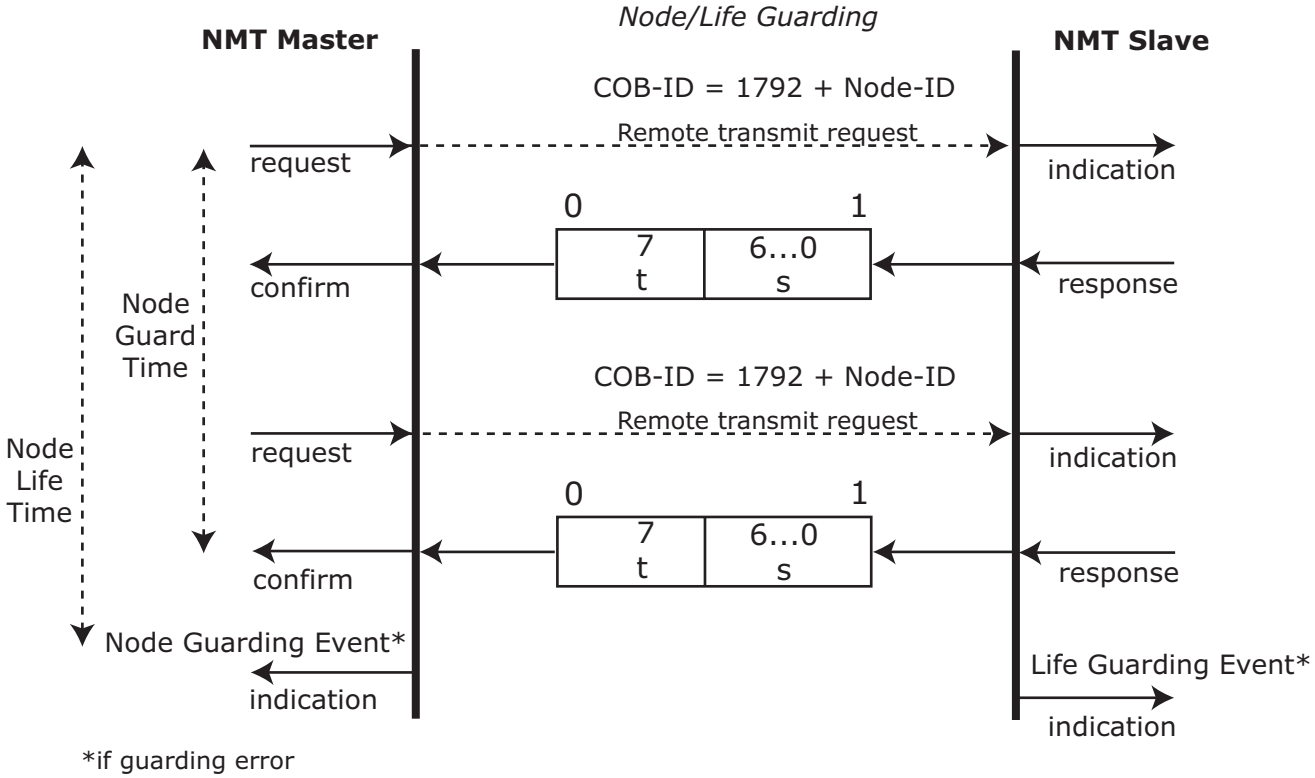


Figure 3.9: Node Guarding Protocol

s: the state of the NMT Slave
- 4: STOPPED
- 5: OPERATIONAL
- 127: PRE-OPERATIONAL

t: toggle bit. The value of this bit must alternate between two consecutive responses from the NMT Slave. If a response is received with the same value of the toggle-bit as in the preceding response then the new response is handled as if it was not received.

The NMT Master polls each NMT Slave at regular time intervals. This time-interval is called the *guard-time* and may be different for each NMT Slave. The response of the NMT Slave contains the state of the NMT Salve.
The *node life time* is given by the guard time multiplied by the life time factor. If the NMT Slave has not been polled during its life time, a *remote node error* is indicated through the "Life Guarding Event" service.
A remote node error is indicated through the "Node Guarding Event" service if:
- the remote transmit request is not confirmed within the node life time;
- the reported NMT slave state does not match the expected state.

3.3.2 – Bootup Protocol

This protocol is used to signal that a NMT Slave has entered the node state PRE-OPERATIONAL after the state INITIALISING. The protocol uses the same identifier as the error control protocol.

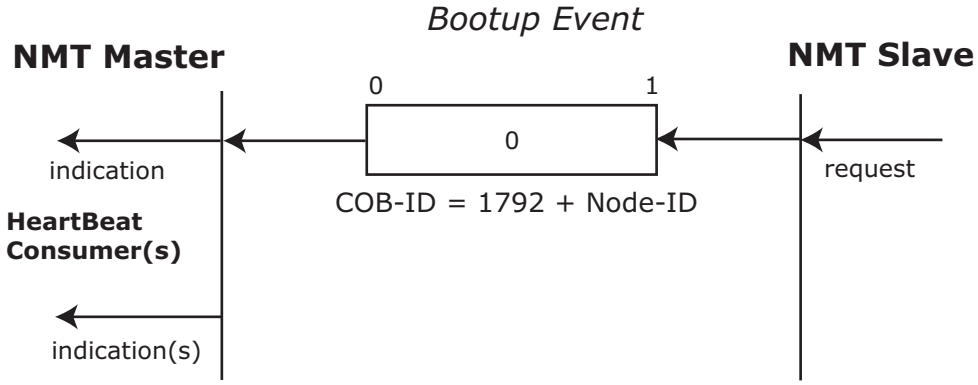


Figure 3.10: Bootup Protocol

One data byte is transmitted with value 0.

Chapter 4
Object Dictionary

4.1 - General Structure of the Object Dictionary

The format of the Object Dictionary entries used by all Axor drives is shown below:

Index (hex)	Object (Symbolic Name)	Name	Type	Attribute	M/O

Table 4.1: Format of Object Dictionary Headings

The complete Object Dictionary format consists of the six columns shown above.
The *Index* column denotes the object position within the Object Dictionary. This acts as a kind of address to reference the desired data field. The sub-index is not specified here. The sub-index is used to reference data fields within a complex object such as an array or record.
The *Object* column contains the Object Name and is used to denote what kind of object is at that particular index within the Object Dictionary. The following definitions are used:

Object Name	Comments	Object Code
NULL	A dictionary entry with no data fields.	0
DOMAIN	Large variable amount of data.	2
DEFTYPE	Denotes a type definition such as a Boolean, UNSIGNED, float and so on.	5
VAR	A single value such as an UNSIGNED8, Boolean, float and so on.	7
ARRAY	A multiple data field object where each data field is a simple variable of the same data type; e.g. array of UNSIGNED16 etc. Sub-index 0 is of UNSIGNED8 and therefore not part of the ARRAY data.	8
RECORD	A multiple data field object where the data fields may be any combination of simple variables. Sub index 0 is made of UNSIGNED8 and therefore it is not part of the RECORD data.	9

Table 4.2: Object Dictionary Object Definition

The *Name* column provides a simple textual description of the function of that particular object.

The *Type* column gives information as to the type of the object. These includes the following pre-defined types: BOOLEAN, floating point number, UNSIGNED Integer, Signed Integer, visible/octet string, time-of-day, time difference and DOMAIN. It also includes the pre-defined complex data type PDOMapping and may also include others which are either manufacturer or device specific. It is not possible to define records of records, arrays of records or records with arrays as fields of that record. In the case where an object is an array or a record the sub-index is used to reference one data field within the object.

The *Attribute* column defines the access rights for a particular object. The view point is from the bus into the device. It can be of the following:

Attribute	Description
rw	read and write access
wo	write only access
ro	read only access
Const	read only access, value is costant

Table 4.3: Access Attribute for Data Objects

Optional objects listed in the Object Dictionary with the Attribute rw may be implemented as read only. Exceptions are defined in the detailed object specification.

The *M/O* column defines whether the object is **M**andatory or **O**ptional. A *mandatory* object must be implemented on a device. An *optional* object needs not to be implemented on a device. The support of certain objects or features however may require the implementation of related data objects. In this case, the relations are described in the detailed object specification.

4.2 - Dictionary Components

Using the index, the Object Dictionary can be divided in the following way:

- Index 01h – 1Fh contain the *standard data type*.
- Index 20h – 23h contain predefined *complex data types*.
- The range of indices from 24h to 3Fh is not defined yet, but *reserved* for future standard data structures.
- The range 40h – 5Fh is free for manufacturers to define own *complex data types*.
- The indices from 60h to 7F contain device *profile specific standard data types*.
- From 80h to 9Fh are defined device *profile specific complex data types*.
- The range A0h – 25Fh is *reserved* for the data type definitions for Multiple Device Modules similar to entries 60h – 9Fh.
- The entries 360h – FFFh are *reserved* for possible future enhancement.
- The range **1000h – 1FFFh** contains the communication specific Object Dictionary entries. These parameters are called *communication entries*, their specification is common to all devices types, regardless of the device profile they use.
- The range 2000h – 5FFFh is free for *manufacturer specific profile definition*.
- The entries 6000h – 9FFFh contain the *standardised device profile parameters*.
- The range A000h – FFFFh is *reserved* for future use.

4.3 - Communication Entries (1000h - 1FFFh)

Objects described below are all communication specific Object Dictionary entries used by Axor drives.

Object 1000h: Device Type

It is composed of a LSW that describes the *device profile* that is used and a MSW which gives additional information about *optional functionality* of the device. In this case the device profile is 402 (0192h) and the additional information indicates the drive is a servo-drive (0002h).

OBJECT DESCRIPTION

INDEX	1000h
Name	Device Type
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory

ENTRY DESCRIPTION

Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0002 0192h

Object 1001h: Error Register

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

OBJECT DESCRIPTION

INDEX	1001h
Name	Error Register
Object Code	VAR
Data Type	UNSIGNED8
Category	Mandatory

ENTRY DESCRIPTION

Access	ro
PDO Mapping	Possible
Value Range	UNSIGNED8
Default Value	No

The structure of the error register is shown below:

Bit	M/O	Meaning
0	M	Generic error
1	O	Current
2	O	Voltage
3	O	Temperature
4	O	Communication Error
5	O	Device Profile Specific
6	O	Reserved (always 0)
7	O	Manufacturer Specific

Table 4.4: Structure of the Error Register

AXOR drives use all bits (excluding bit 6) to signal the presence of errors.

Object 1006h: Communication Cycle Length

OBJECT DESCRIPTION

INDEX	1006h
Name	Communication Cycle Length
Object Code	VAR
Data Type	Unsigned32
Category	Conditional: Mandatory for Sync Producer

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	Unsigned32
Default Value	0

Object 1008h: Manufacturer Device Name
Contains the *manufacturer device name*. The value is 524F 5841h and represents the ASCII code of the word AXOR.

OBJECT DESCRIPTION

INDEX	1008h
Name	Manufacturer Device Name
Object Code	VAR
Data Type	Visible String
Category	Optional

ENTRY DESCRIPTION

Access	Const
PDO Mapping	No
Value Range	No
Default Value	524F 5841h

Object 1009h: Manufacturer Hardware Version
Contains the *manufacturer hardware version* description.

OBJECT DESCRIPTION

INDEX	1009h
Name	Manufacturer Hardware Version
Object Code	VAR
Data Type	Visible String
Category	Optional

ENTRY DESCRIPTION

Access	Const
PDO Mapping	No
Value Range	No
Default Value	No

Object 100Ah: Manufacturer Software Version
Contains the *manufacturer software version* description.

OBJECT DESCRIPTION

INDEX	100Ah
Name	Manufacturer Software Version
Object Code	VAR
Data Type	Visible String
Category	Optional

ENTRY DESCRIPTION

Access	Const
PDO Mapping	No
Value Range	No
Default Value	No

Object 100Ch: Guard Time
The objects at index 100Ch and 100Dh include the *guard time* in milliseconds and the *life time factor*. The life time factor multiplied with the guard time gives the *life time* for the Life Guarding Protocol.

OBJECT DESCRIPTION

INDEX	100Ch
Name	Guard Time
Object Code	VAR
Data Type	UNSIGNED16
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED16
Default Value	1000

Object 100Dh: Life Time Factor
The *life time factor* multiplied with the guard time gives the life time for the Life Guarding Protocol. It is 0 if not used.

OBJECT DESCRIPTION

INDEX	100Dh
Name	Life Time Factor
Object Code	VAR
Data Type	UNSIGNED16
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	3

Object 1017h: Producer Hearbeat Time

OBJECT DESCRIPTION

INDEX	1017h
Name	Producer Heartbeat Time
Object Code	VAR
Data Type	UNSIGNED16
Category	Conditional. Mandatory, if guarding is not supported.

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED16
Default Value	0

Objects 1400h - 15FFh: Receive PDO Communication Parameter
Contain the *communication parameters* for the PDOs the device is able to receive.
The sub-index 0h contains the *number of valid entries* within the communication record. Its value is at least 2.
At sub-index 1h resides the *COB-ID* of the PDO. The structure and the description of PDO COB-ID entry are illustrated in Table 4.5 and Table 4.6:

	UNSIGNED32 MSB				LSB
bits	31	30	29	28-11	10-0
11-bit-ID	0/1	0/1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-bit Identifier
29-bit-ID	0/1	0/1	1	29-bit Identifier	

Table 4.5: Structure of PDO COB-ID entry

Bit Number	Value	Meaning
31 (MSB)	0	PDO exists/is valid
	1	PDO does not exist/is not valid
30	0	RTR allowed on this PDO
	1	no RTR 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 4.6: Description of 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 but not used, and therefore set to "not valid". 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.
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/receptions character of the PDO (see Table 4.7). 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 0030h) is generated.

Transmission Type	**PDO Transmission**				
	cyclic	acyclic	synchronous	asynchronous	RTR only
0		X	X		
1-240	X		X		
241-251	- reserved -				
252			X		X
253				X	X
254				X	
255				X	

Table 4.7: Description of transmission type

Synchronous means that the transmission of the PDO shall be related to the SYNC object.
Asynchronous means that the transmission of the PDO is not related to the SYNC object.
Cyclic means that the transmission of the PDO happens every n SYNC objects, with n between 1 and 240.
Acyclic means that the transmission of the PDO is caused by an event and it is synchronized with the SYNC object.
RTR only means that the PDO is only transmitted on remote transmission request (RTR only); e.g. at transmission type 252, the data is update (but not sent) immediately after reception of the SYNC object, while at transmission type 253 the data is updated at the reception of the remote transmission request. These value are only possible for TPDOs.
Receive PDOs are always triggered by the following SYNC upon reception of data independent of the transmission type 0-240.
For TPDOs transmission type 254 means the application event is manufacturer specific, transmission type 255 means the application event is defined in the device profile.

Sub-index 3h contains the *inhibit time*. This time is the 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.

Sub-index 4h is reserved. It does not have to be implemented.

In mode 254/255 additionally an event time can be used for TPDO. If an event timer exists for a TPDO the elapsed timer is considered to be an event. The *event timer* elapses as multiple of 1ms of the entry in sub-index 5h of the TPDO. This event will cause the transmission of the TPDO in addition to otherwise defined events. The occurrence of the events set the timer.
Independent of the transmission type the RPDO event timer is used recognise the expiration of the RPDO.

Objects 1400h : Receive PDO1 Communication Parameter

OBJECT DESCRIPTION

INDEX	1400h
Name	receive PDO1 communication parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	largest sub-index supported
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	COB-ID used by PDO
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	200h + Node-ID

Sub-Index	2h
Description	trasmission type
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	1

Objects 1401h : Receive PDO2 Communication Parameter

OBJECT DESCRIPTION

INDEX	1401h
Name	receive PDO2 communication parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	largest sub-index supported
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	COB-ID used by PDO
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	300h + Node-ID

Sub-Index	2h
Description	trasmission type
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	1

Objects 1402h : Receive PDO3 Communication Parameter

OBJECT DESCRIPTION

INDEX	1402h
Name	receive PDO3 communication parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	largest sub-index supported
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	COB-ID used by PDO
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	400h + Node-ID

Sub-Index	2h
Description	trasmission type
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	1

Objects 1403h : Receive PDO4 Communication Parameter

OBJECT DESCRIPTION

INDEX	1403h
Name	receive PDO4 communication parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	largest sub-index supported
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	COB-ID used by PDO
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	500h + Node-ID

Sub-Index	2h
Description	trasmission type
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	1

Objects 1600h - 17FFh: Receive PDO Mapping Parameter
Contain the *mapping* for the PDOs the device is able to receive.
The sub-index 0h contains the *number of valid entries* within the mapping record. This number 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..40h). This parameter can be used to verify the overall mapping length. It is mandatory.
The structure of the entries from sub-index 1h-40h is as follows:

MSB		LSB
index (16 bit)	sub-index (8 bit)	object length (8 bit)

Table 4.8: Structure of PDO Mapping Entry

If the change of the PDO mapping cannot be execute, the device responds with an Abort SDO Transfer Service.
Sub-index 0h 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 0h must be set to 0. Then the object can be remapped. When a new object is mapped by writing a subindex 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 0h 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 0h 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 the subindex 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 TPDO.
A device that supports dynamic mapping of PDOs must supports this during the state PRE-OPERATIONAL state. If dynamic mapping during the state OPERATIONAL is supported, the SDO client is responsible for data consistency.

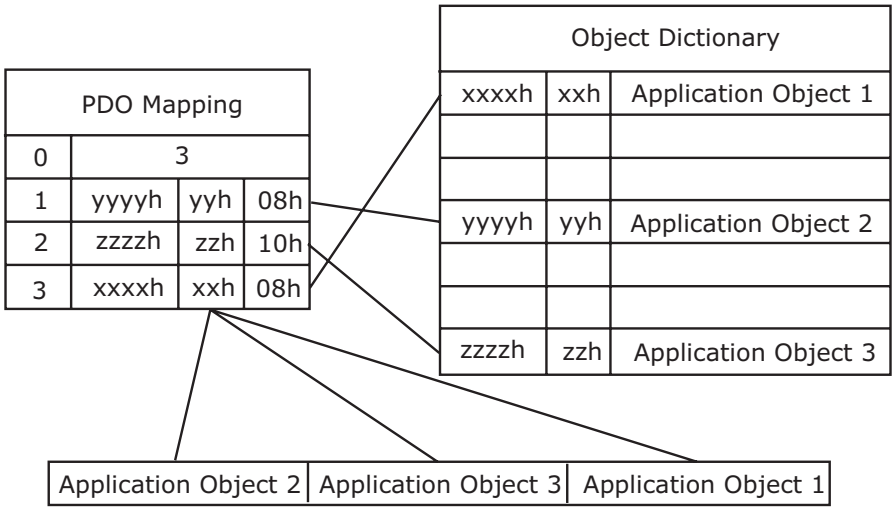


Figure 4.1: Principle of PDO mapping

PDO no.	Mapping Object index	Mapping Object name	Comment
1	6040h	Controlword	Controls the state machine.
2	6040h 6060h	Controlword Modes of operation	Controls the state machine and modes of operation.
3	6040h 607Ah	Controlword Target position	Controls the state machine and the target position.
4	6040h 60FFh	Controlword Target velocity	Controls the state machine and the target velocity.

Table 4.9: Predefined RPDO mapping

Object 1600h: Receive PDO1 Mapping

OBJECT DESCRIPTION

INDEX	1600h
Name	receive PDO1 mapping
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of mapped application objects in PDO
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0: deactivated 1-4: activated
Default Value	1

Sub-Index	1h
Description	RPDO1 mapping for the 1st application object to be mapped
Entry Category	Conditional, it depends on number and size of object be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60400010

Object 1601h: Receive PDO2 Mapping

OBJECT DESCRIPTION

INDEX	1601h
Name	receive PDO2 mapping
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of mapped application objects in PDO
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0: deactivated 1-4: activated
Default Value	2

Sub-Index	1h
Description	RPDO2 mapping for the 1st application object to be mapped
Entry Category	Conditional, it depends on number and size of object be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60400010

Sub-Index	2h
Description	RPDO2 mapping for the 2nd application object to be mapped
Entry Category	Conditional, it depends on number and size of object be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60600010

Object 1602h: Receive PDO3 Mapping

OBJECT DESCRIPTION

INDEX	1602h
Name	receive PDO3 mapping
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of mapped application objects in PDO
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0: deactivated 1-4: activated
Default Value	2

Sub-Index	1h
Description	RPDO3 mapping for the 1st application object to be mapped
Entry Category	Conditional, it depends on number and size of object be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60400010

Sub-Index	2h
Description	RPDO3 mapping for the 2nd application object to be mapped
Entry Category	Conditional, it depends on number and size of object be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	607A0020

Object 1603h: Receive PDO4 Mapping

OBJECT DESCRIPTION

INDEX	1603h
Name	receive PDO4 mapping
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of mapped application objects in PDO
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0: deactivated 1-4: activated
Default Value	2

Sub-Index	1h
Description	RPDO4 mapping for the 1st application object to be mapped
Entry Category	Conditional, it depends on number and size of object be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60400010

Sub-Index	2h
Description	RPDO4 mapping for the 2nd application object to be mapped
Entry Category	Conditional, it depends on number and size of object be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60FF0020

Object 1800h: Transmit PDO1 Communication Parameter
Contains the *communication parameters* for the PDO1 the device is able to transmit.

OBJECT DESCRIPTION

INDEX	1800h
Name	transmit PDO1 parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	largest sub-index supported
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	COB-ID used by PDO
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	180h + Node-ID

Sub-Index	2h
Description	transmission type
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	1

Object 1801h: Transmit PDO2 Communication Parameter
Contains the *communication parameters* for the PDO1 the device is able to transmit.

OBJECT DESCRIPTION

INDEX	1801h
Name	transmit PDO2 parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	largest sub-index supported
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	COB-ID used by PDO
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	280h + Node-ID

Sub-Index	2h
Description	transmission type
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	1

Object 1802h: Transmit PDO3 Communication Parameter
Contains the *communication parameters* for the PDO1 the device is able to transmit.

OBJECT DESCRIPTION

INDEX	1802h
Name	transmit PDO3 parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	largest sub-index supported
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	COB-ID used by PDO
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	380h + Node-ID

Sub-Index	2h
Description	transmission type
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	1

Object 1803h: Transmit PDO4 Communication Parameter
Contains the *communication parameters* for the PDO1 the device is able to transmit.

OBJECT DESCRIPTION

INDEX	1803h
Name	transmit PDO4 parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	largest sub-index supported
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	COB-ID used by PDO
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	480h + Node-ID

Sub-Index	2h
Description	transmission type
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED8
Default Value	1

Objects 1A00h - 1BFFh: Transmit PDO Mapping Parameter
Contain the *mapping* for the PDOs the device is able to transmit.

PDO no.	Mapping Object index	Mapping Object name	Comment
1	6041h	Statusword	Shows status
2	6064h 606Ch	Position actual value Velocity actual value	Show current position and velocity
3	6041h 6064h	Statusword Position actual value	Show status and current position
4	6041h 606Ch	Statusword Velocity actual value	Show status and current velocity

Table 4.10: Predefined TPDO mapping

Object 1A00h: Transmit PDO1 Mapping

OBJECT DESCRIPTION

INDEX	1A00h
Name	transmit PDO1 parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of mapped application objects in PDO
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0: deactivated 1 - 4: activated
Defaul Range	1

Sub-Index	1h
Description	TPDO1 mapping for the 1st application object to be mapped
Entry Category	Conditional; it depends on number and size of objects to be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60410010

Object 1A01h: Transmit PDO2 Mapping

OBJECT DESCRIPTION

INDEX	1A01h
Name	transmit PDO2 parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of mapped application objects in PDO
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0: deactivated 1 - 4: activated
Defaul Range	1

Sub-Index	1h
Description	TPDO2 mapping for the 1st application object to be mapped
Entry Category	Conditional; it depends on number and size of objects to be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60640020

Sub-Index	2h
Description	TPDO2 mapping for the 2nd application object to be mapped
Entry Category	Conditional; it depends on number and size of objects to be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	606C0020

Object 1A02h: Transmit PDO3 Mapping

OBJECT DESCRIPTION

INDEX	1A02h
Name	transmit PDO3 parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of mapped application objects in PDO
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0: deactivated 1 - 4: activated
Defaul Range	2

Sub-Index	1h
Description	TPDO3 mapping for the 1st application object to be mapped
Entry Category	Conditional; it depends on number and size of objects to be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60410010

Sub-Index	2h
Description	TPDO3 mapping for the 2nd application object to be mapped
Entry Category	Conditional; it depends on number and size of objects to be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60640020

Object 1A03h: Transmit PDO4 Mapping

OBJECT DESCRIPTION

INDEX	1A03h
Name	transmit PDO4 parameter
Object Code	RECORD
Category	Conditional; Mandatory for each supported PDO

ENTRY DESCRIPTION

Sub-Index	0h
Description	number of mapped application objects in PDO
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0: deactivated 1 - 4: activated
Defaul Range	2

Sub-Index	1h
Description	TPDO4 mapping for the 1st application object to be mapped
Entry Category	Conditional; it depends on number and size of objects to be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	60410010

Sub-Index	2h
Description	TPDO4 mapping for the 2nd application object to be mapped
Entry Category	Conditional; it depends on number and size of objects to be mapped
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	606C0020

Chapter 5
Device Control

5.1 – General information

The device control function block controls all functions of the drive and the power section.
It is divided into:

1. Device control of the state machine.
2. Operation mode function.

The state of the drive can be controlled by the *controlword*.
The state of the drive can be shown in the *statusword*.

5.2 – State Machine

The state machine describes the device status and the possible control sequence of the drive.
The state machine is controlled externally by the *controlword* and external signals; it is also controlled by internal signals like faults and modes of operation.
The current state can be read using the *statusword*.

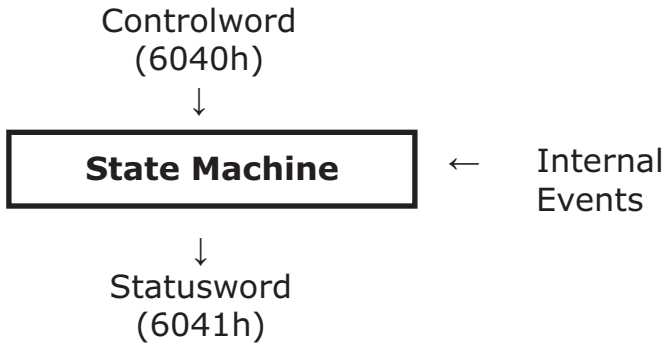


Figure 5.1: State Machine in system context

5.2.1 Drive states

The following states of the device are possible:

• NOT READY TO SWITCH ON:
Low level power (e.g. ± 15V, 5V) has been applied to the drive.
The drive is being initialised or is running self test.
A brake, if present, has to be applied in this state.
The drive function is disabled.

• SWITCH ON DISABLED:
Drive initialisation is complete.
The drive parameters have been set up.
Drive parameters may be changed.
High voltage may not be applied to the drive.
The drive function is disabled.

• READY TO SWITCH ON:
High voltage may be applied to the drive.
The drive parameters may be changed.
The drive function is disabled.

• SWITCHED ON:
High voltage has been applied to the drive.
The power amplifier is ready.
The drive parameters may be changed.
The drive function is disabled.

• OPERATION ENABLE:
No faults has been detected.
The drive function is enabled and power is applied to the motor.
The drive parameters may be changed.

• QUICK STOP ACTIVE:
The drive parameters may be changed.
The *quick stop function* is being executed.
The drive function is enabled and power is applied to the motor.

• FAULT REACTION ACTIVE:
The drive parameters may be changed.
A fault has occurred in the drive.
The quick stop function is being executed.
The drive function is enabled and power is applied to the motor.

• FAULT:
The drive parameters may be changed.
A fault has occurred in the drive.
High voltage switch –on/-off depends on the application.
The drive function is disabled.

5.2.2 State transitions

State transitions are caused by *internal events* in the drive or by commands from the host via the *controlword*.
The Figure 5.2 illustrates all possible state transitions.

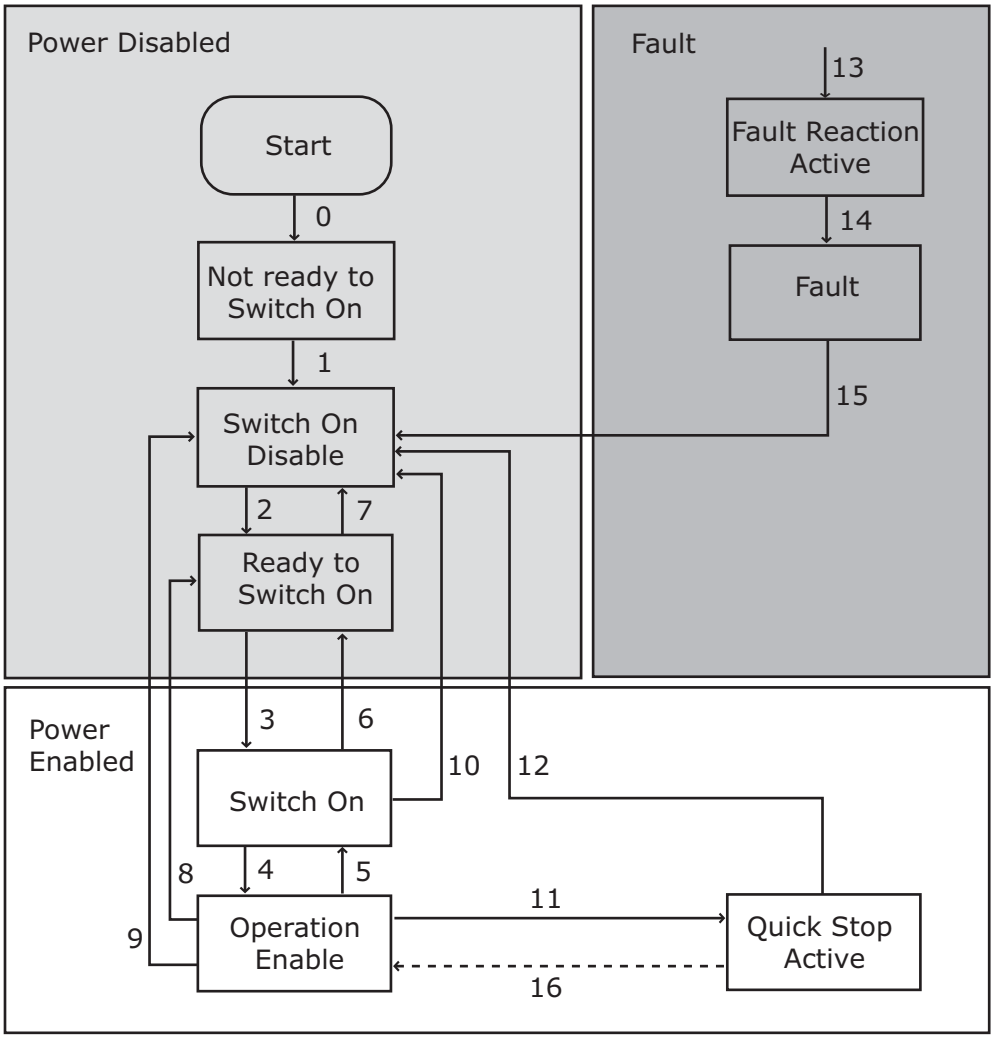


Figure 5.2: State transitions of the State Machine

- State Transition 0: START ⇨ NOT READY TO SWITCH ON
  Event: Reset
  Action: The drive self-tests and/or self-initialized.

- State Transition 1: NOT READY TO SWITCH ON ⇨ SWITCH ON DISABLED
  Event: The drive has self-tested and/or initialized successfully.
  Action: Activate communication.

- State Transition 2: SWITCH ON DISABLED ⇨ READY TO SWITCH ON
  Event: "Shutdown" command received from host.
  Action: None.

- State Transition 3: READY TO SWITCH ON ⇨ SWITCH ON
  Event: "Switch On" command received from host.
  Action: The power section is switched on if it is not already switched on.

- State Transition 4: SWITCH ON ⇨ OPERATION ENABLE
  Event: "Enable Operation" command received from host.
  Action: The drive function is enabled.

- State Transition 5: OPERATION ENABLE ⇨ SWITCH ON
  Event: "Disable Operation" command received from host.
  Action: The drive operation will be disabled.

- State Transition 6: SWITCH ON ⇨ READY TO SWITCH ON
  Event: "Shutdown" command received from host.
  Action: The power section is switched off.

- State Transition 7: READY TO SWITCH ON ⇨ SWITCH ON DISABLED
  Event: "Quick Stop" and "Disabled Voltage" commands received from host.
  Action: None.

- State Transition 8: OPERATION ENABLE ⇨ READY TO SWITCH ON
  Event: "Shutdown" command received from host.
  Action: The power section is switched off immediately and the motor is free to rotate if unbraked.

- State Transition 9: OPERATION ENABLE ⇨ SWITCH ON DISABLED
  Event: "Disabled Voltage" command received from host.
  Action: The power section is switched off immediately and the motor is free to rotate if unbraked.

- State Transition 10: SWITCH ON ⇨ SWITCH ON DISABLED
  Event: "Disabled Voltage" or "Quick Stop" command received from host.
  Action: The power section is switched off immediately and the motor is free to rotate if unbraked.

- State Transition 11: OPERATION ENABLE ⇨ QUICK STOP ACTIVE
  Event: "Quick Stop" command received from host.
  Action: The quick stop function is executed.

- State Transition 12: QUICK STOP ACTIVE ⇨ SWITCH ON DISABLED
  Event: "Quick Stop" is completed or "Disabled Voltage" command received from host.
  Action: The power section is switched off.

- State Transition 13: All states ⇨ FAULT REACTION ACTIVE
  Event: A fault has occurred in the drive.
  Action: Execute appropriate fault reaction.

- State Transition 14: FAULT REACTION ACTIVE ⇨ FAULT
  Event: The fault reaction is completed.
  Action: The drive function is disabled. The power section may be switched off.

- State Transition 15: FAULT ⇨ SWITCH ON DISABLED
  Event: "Fault Reset" command received from host.
  Action: A reset of the fault condition is carried out if no fault exists currently on the drive. After leaving the state *Fault* the bit *Fault Reset* of the controlword has to be cleared by the host.

- State Transition 16: QUICK STOP ACTIVE ⇨ OPERATION ENABLE
  Event: "Enable Operation" command received from host.
  Action: The drive function is disabled.

Notes:
A single state represents a special internal or external behaviour. The state of the drive also determines which commands are accepted. For example, it is only possible to start a point-to-point movement when the drive is in the state OPERATION ENABLE.

If a command is received which causes a change of state, this command must be processed completely and the new state attained before the next command can be processed.

"Fault occurred" implies that a fault in the drive has occurred. In this case there is a transition to the state FAULT REACTION ACTIVE. In this state the device will execute a special fault reaction. After the execution of this fault reaction the device will switch to the state FAULT. This state can only be left by the command "Fault Reset", but only if the fault is no more present.

5.2.3 Object Description - Standardised Profile Area

Object 6007h: Abort connection option code

OBJECT DESCRIPTION

INDEX	6007h
Name	Abort connection option code
Object Code	VAR
Data Type	INTEGER16
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	INTEGER16
Default Value	No

Object 6040h: Controlword

The *controlword* consists of bits for:

- the controlling of the state;
- the controlling of operation modes;
- manufactor specific options.

OBJECT DESCRIPTION

INDEX	6040h
Name	Controlword
Object Code	VAR
Data Type	UNSIGNED16
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED16
Default Value	No

DATA DESCRIPTION
The bits of the controlword are defined as follows:

15 11	10 9	8	7	6 4	3	2	1	0
Manufacturer Specific	Reserved	Halt	Fault Reset	Operation Mode Specific	Operation Enable	Quick Stop	Enable Voltage	Switch On
O	**O**	**O**	**M**	**O**	**M**	**M**	**M**	**M**

MSB LSB

where:
M = Mandatory
O = Optional

BITS 0, 1, 2, 3 AND 7:
Device control commands are triggered by the following bit patterns in the *controlword*:

COMMAND	**Bits of the *controlword***				
	Fault Reset	**Enable Operation**	**Quick Stop**	**Enable voltage**	**Switch On**
Shutdown	0	X	1	1	0
Switch on	0	0	1	1	1
Disable voltage	0	X	X	0	X
Quick Stop	0	X	0	1	X
Disable Operation	0	0	1	1	1
Enable Operation	0	1	1	1	1
Fault Reset	0 → 1	X	X	X	X

Note: Bits marked X are irrelevant.

BITS 4, 5, 6 AND 8:
These bits are operation mode specific.

BITS 9 AND 10:
These bits are reserved for further use. They must be set to zero.

BITS 11, 12, 13, 14 AND 15:
These bits are manufacturer specific. At the moment they are irrelevant for AXOR drives.

Object 6041h: Statusword
The *statusword* indicates the actual state of the drive. No bits are latched. The statusword consists of bits for:

- the actual state of the drive,
- the operating state of the mode,
- manufacturer specific options.

OBJECT DESCRIPTION

INDEX	6041h
Name	Statusword
Object Code	VAR
Data Type	UNSIGNED16
Category	Mandatory

ENTRY DESCRIPTION

Access	ro
PDO Mapping	Possible
Value Range	UNSIGNED16
Default Value	No

DATA DESCRIPTION

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MSB LSB

BIT	DESCRIPTION	M/O
0	Not Ready To Switch On	M
1	Switched On	M
2	Operation Enabled	M
3	Fault	M
4	Voltage Enabled	M
5	Quick Stop	M
6	Switch On Disable	M
7	Warning	O
8	Manufacturer Specific	O
9	Remote	M
10	Target Reached	M
11	Internal Limit Active	M
12 - 13	Operation Mode Specific	O
14 - 15	Manufaturer Specific	O

BITS 0, 1, 2, 3, 5 AND 6:
The following bits indicate the status of the device:

Value (binary)	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1111	Fault

BIT 4: Voltage Enabled
High voltage is applied to the drive when this bit is set to 1.

BIT 5: Quick Stop
When reset (1 → 0), 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.

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. The status of the drive does not change. The bit is set and reset by the device.

BIT 8:
This bit may be used by a drive manufacturer. At the moment is not used by Axor drive.

BIT 9: Remote
Remote bit is not used by Axor drives.

BIT 10: Target Reached
If bit 10 is set by the drive, then a set point has been reached. The set point is dependent on the operating mode. The change of a target value by software alters this bit. 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.

BIT 12 AND 13:
These bits are operation mode specific. The description is situated in the mode of operation chapters.

BITS 14 AND 15:
These bits are manufacturer specific. They are not used by Axor drives.

Object 6064h: Position Actual Value
This object represents the actual value of the position measurement device in user defined units.

OBJECT DESCRIPTION

INDEX	6064h
Name	Position Actual Value
Object Code	VAR
Data Type	INTEGER32
Category	Mandatory

ENTRY DESCRIPTION

Access	ro
PDO Mapping	Possible
Value Range	INTEGER32
Default Value	No

Object 6073h: Max Current Value

OBJECT DESCRIPTION

INDEX	6073h
Name	Max Current Value
Object Code	VAR
Data Type	UNSIGNED16
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED16
Default Value	No

Object 6075h: Motor Rated Current

OBJECT DESCRIPTION

INDEX	6075h
Name	Motor Rated Current
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	No

Object 607Eh: Polarity

OBJECT DESCRIPTION

INDEX	607Eh
Name	Polarity
Object Code	VAR
Data Type	UNSIGNED8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED8
Default Value	No

Object 607Fh: Max Profile Velocity
The max profile velocity is the maximum allowed speed in either direction during a profiled move. It is given in the same units as profile velocity.

OBJECT DESCRIPTION

INDEX	607Fh
Name	Max Profile Velocity
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	No

Object 6083h: Profile Acceleration
The profile acceleration is given in user defined acceleration units.

OBJECT DESCRIPTION

INDEX	6083h
Name	Profile Acceleration
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	No

Object 6084h: Profile Deceleration
The profile deceleration is given in user defined acceleration units.

OBJECT DESCRIPTION

INDEX	6084h
Name	Profile Deceleration
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	No

Object 6085h: Quick Stop Deceleration
The Quick Stop Deceleration is the deceleration used to stop the motor if the "Quick Stop" command is given. The Quick Stop Deceleration is given in the same units as the Profile Acceleration.

OBJECT DESCRIPTION

INDEX	6085h
Name	Quick Stop Deceleration
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	No

Object 6089h: Position Notation Index

OBJECT DESCRIPTION

INDEX	6089h
Name	Position Notation Index
Object Code	VAR
Data Type	UNSIGNED8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED8
Default Value	No

Object 608Ah: Position Dimension Index

OBJECT DESCRIPTION

INDEX	608Ah
Name	Position Dimension Index
Object Code	VAR
Data Type	UNSIGNED8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED8
Default Value	No

Object 608Bh: Velocity Notation Index

OBJECT DESCRIPTION

INDEX	608Bh
Name	Velocity Notation Index
Object Code	VAR
Data Type	INTEGER8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	INTEGER8
Default Value	No

Object 608Ch: Velocity Dimension Index

OBJECT DESCRIPTION

INDEX	608Ch
Name	Velocity Dimension Index
Object Code	VAR
Data Type	UNSIGNED8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED8
Default Value	No

Object 608Dh: Acceleration Notation Index

OBJECT DESCRIPTION

INDEX	608Dh
Name	Acceleration Notation Index
Object Code	VAR
Data Type	INTEGER8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	INTEGER8
Default Value	No

Object 608Eh: Acceleration Dimension Index

OBJECT DESCRIPTION

INDEX	608Eh
Name	Acceleration Dimension Index
Object Code	VAR
Data Type	UNSIGNED8
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED8
Default Value	No

Object 6092h: Feed Constant

OBJECT DESCRIPTION

INDEX	6092h
Name	Feed Constant
Object Code	ARRAY
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Sub-Index	0h
Description	Number of Entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2
Default Value	2

Sub-Index	1h
Description	Feed
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	30000

Sub-Index	2h
Description	Shaft revolutions
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	1

Object 60F4h: Following error actual value

OBJECT DESCRIPTION

INDEX	60F4h
Name	Following error actual value
Object Code	VAR
Data Type	INTEGER32
Category	Optional

ENTRY DESCRIPTION

Access	ro
PDO Mapping	Possible
Value Range	INTEGER32
Default Value	No

Object 60F9h: Velocity Control Parameter Set

OBJECT DESCRIPTION

INDEX	60F9h
Name	Velocity Control Parameter Set
Object Code	RECORD
Category	Optional

ENTRY DESCRIPTION

Sub-Index	0h
Description	Number of Entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	Kp
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED16

Sub-Index	2h
Description	Ki
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED16

Sub-Index	3h
Description	Kp
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED16

Sub-Index	4h
Description	PID Filter
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED16

Sub-Index	5h
Description	Feedback Filter
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED16

Object 60FBh: Position Control Parameter Set

OBJECT DESCRIPTION

INDEX	60FBh
Name	Position Control Parameter Set
Object Code	RECORD
Category	Optional

ENTRY DESCRIPTION

Sub-Index	0h
Description	Number of Entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2

Sub-Index	1h
Description	Dynamic Kp
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED16

Sub-Index	2h
Description	Static Kp
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED16

Object 60FDh: Digital Inputs

OBJECT DESCRIPTION

INDEX	60FDh
Name	Digital Inputs
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	ro
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	No

5.2.4 Object Description - Manufacturer Specific Profile Area

Object 2000h: Alarm Actual Value

OBJECT DESCRIPTION

INDEX	2000h
Name	Alarm Actual Value
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0

Object 2002h: Max Following Error

OBJECT DESCRIPTION

INDEX	2002h
Name	Max Following Error
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	---

Object 55FFh: Current Actual Value

OBJECT DESCRIPTION

INDEX	55FFh
Name	Current Actual Value
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

5.3 – Modes of Operation Function

The device behaviour depends on the activated modes of operation.
It is possible to implement different device modes. Since is not possible to operate the modes in parallel, the user is able to activate the required function by selecting a mode of operation.

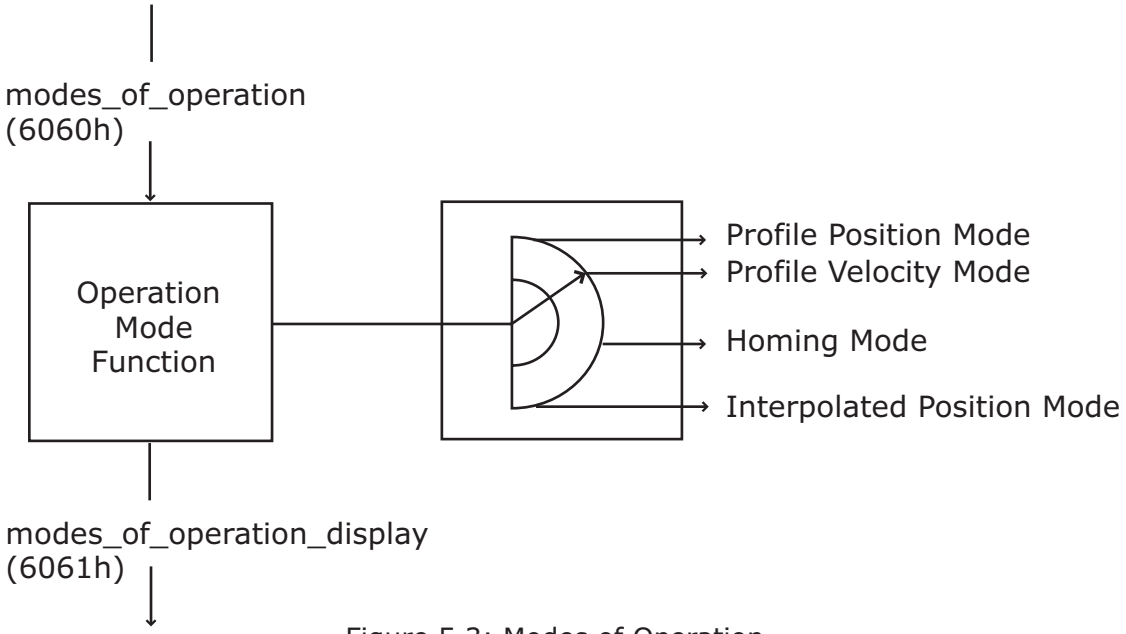


Figure 5.3: Modes of Operation

It is possible for the user to switch between the various modes of operation as long as this is supported by the device. It is possible to switch dynamically between different modes at any time.

The following **modes of operation** are supported:

- *Profile Position Mode*
- *Profile Velocity Mode*
- *Homing Mode*
- *Interpolated Position Mode*

Two objects are defined for management of the modes of operation:

- Modes of Operation (6060h)
- Modes of Operation Display (6061h)

Note: The *statusword* contains bits, whose meaning is dependent on the mode of operation. When changing mode of operation, these bits assume different meanings and need to be monitored.

Object 6060h: Modes of Operation
This object switches the actually chosen operation mode.

OBJECT DESCRIPTION

INDEX	6060h
Name	Modes of Operation
Object Code	VAR
Data Type	INTEGER8
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	INTEGER8
Default Value	1

DATA DESCRIPTION

Value	**Description**
1	Profile Position Mode
3	Profile Velocity Mode
6	Homing Mode
7	Interpolated Position Mode

Object 6061h: Modes of Operation Display
This object display shows the current mode of operation.

OBJECT DESCRIPTION

INDEX	6061h
Name	Modes of Operation Display
Object Code	VAR
Data Type	INTEGER8
Category	Mandatory

ENTRY DESCRIPTION

Access	ro
PDO Mapping	Possible
Value Range	INTEGER8
Default Value	No

5.4 – Profile Position Mode

The *target position* (object 607Ah) is applied to the *trajectory generator* that generates a *position demand value* for the position control loop.
The trajectory generator supports only linear ramps with reserved parameters for acceleration (6083h), deceleration (6084h) and speed limit (6081h).

5.4.1 Object Description

Below is given a list of objects typically used by Profile Position Mode.

Controlword of Profile Position Mode

15 9	8	7	6	5	4	3 0
(See 5.2)	Halt	(See 5.2)	abs/rel	Change set immedialtely	New set-point	(See 5.2)

MSB LSB

Name	Value	Description
New Set-Point	0	Does not assume *target position*
	1	Assume *target position*
Change set Immediately (not implemented)	0	Finish the actual positioning and then start the next positioning (1)
	1	Interrupt the actual positioning and start the next positioning (1)
Abs/rel	0	*Target position* is an absolute value
	1	*Target position* is a relative value
Halt	0	Execute positioning
	1	Stop axle with profile deceleration (if not supported with *profile deceleration*)

Statusword of Profile Position Mode

15 14	13	12	11	10	9 0
(See 5.2)	Followig Error	Set-Point acknowledge	(See 5.2)	Target reached	(See 5.2)

MSB LSB

Name	Value	Description
Target Reached	0	Halt = 0: Target position not reached
		Halt = 1: Axle decelerates
	1	Halt = 0: Target position reached
		Halt = 1: Velocity of axle is 0
Set-point acknowledge	0	Trajectory generator has not assume the positioning values (jet)
	1	Trajectory generator has assume the positioning values
Following Error	0	No following error
	1	Following error

Object 6067h: Position Window
The position window defines a symmetrical range of accepted positions relatively to the *target position*. If the actual value of the position encoder is within the *position window*, the target position is considered reached.

OBJECT DESCRIPTION

INDEX	6067h
Name	Position window
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	No

Object 6068h: Position Window Time
When the actual position is within the *position window* for the period defined by the position window time which is given in multiples of milliseconds, the corresponding bit 10 target reached in the statusword will be set to one.

OBJECT DESCRIPTION

INDEX	6068h
Name	Position Wimdow Time
Object Code	VAR
Data Type	UNSIGNED16
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED16
Default Value	No

Figure 5.4 shows the meaning of the "sub-function position reached". Symmetrically around the *target position* a window is defined for the accepted position range. If a drive is situated in the accepted position range over the time *position window time* the bit target reached (bit 10) in the *statusword* is set.

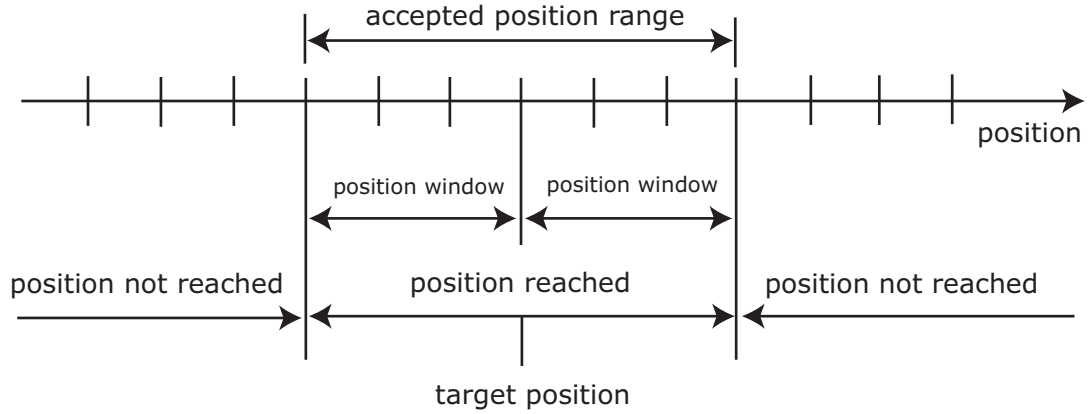


Figure 5.4: Position Reached

Object 607Ah: Target Position
The target position is the position that the Axor drive should move to when it is working in Profile Position Mode. The target position is given in user defined position units. It is converted to position increments using the *position factor*.

OBJECT DESCRIPTION

INDEX	607Ah
Name	Target Position
Object Code	VAR
Data Type	INTEGER32
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	INTEGER32
Default Value	No

Object 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 user defined speed units.

OBJECT DESCRIPTION

INDEX	6081h
Name	Profile Velocity
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	No

5.4.2 Functional description

In the Axor's drives the *target position* is applied in this way: after reaching the target position the drive unit signals this status to a host computer and then receives the 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 bit "change set immediatelly" is always set to 0.
Figure 5.5 illustrates the Set-point transmission from a host computer: after data is applied to the drive, a host signals that the data is valid by changing the bit *new setpoint* to 1 in the *controlword*. The drive responds with *set point acknowledge* set to 1 in the *statusword* after it recognised and buffered the new valid data. Now the host may release *new set-point* and afterwards the drive signals with *set point acknowledge* equal to 0 its ability to accept new data again.

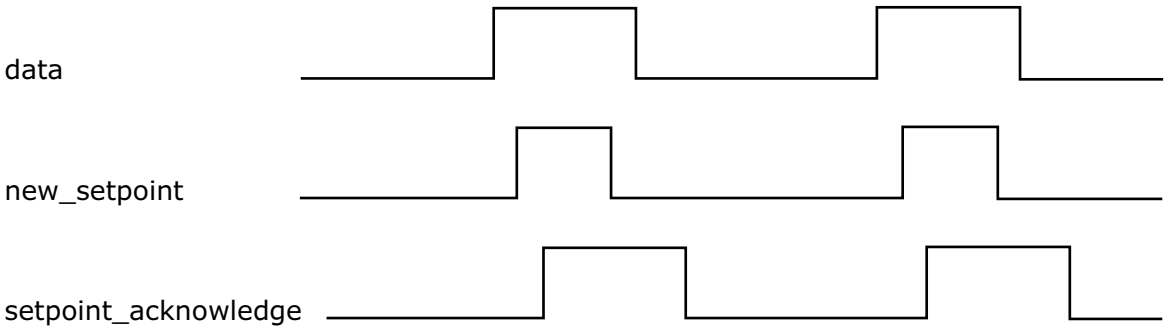


Figure 5.5: Set-point transmission from a host computer

The Figure 5.6 illustrates an example of this functioning: after reaching the target position x_1 at t_1 the velocity is reduced to zero. After signaling to the host that the set point is reached, the next target position x_2 is processed at t_2 and reached at t_3.

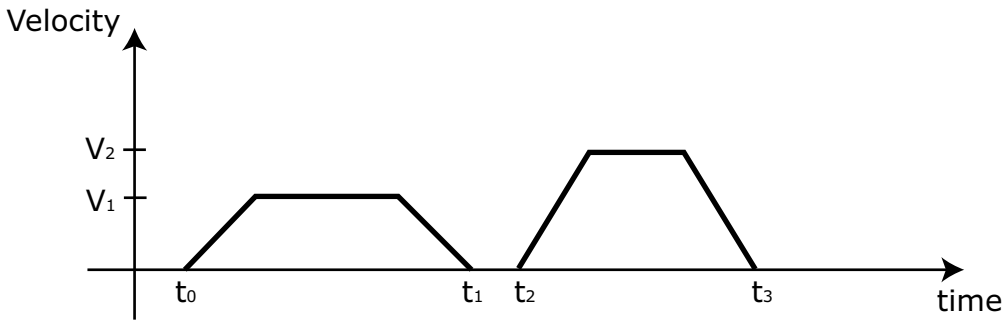


Figure 5.6: Single set-point

5.5 – Homing Mode

This paragraph describes the method by which a drive seeks the *home position*. 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.
The user can specify the *speeds*, *acceleration* and *method* of homing. There is a further object (*home offset*) which allows the user to displace zero in the user's coordinate system from the home position.
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.
There is no 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.

5.5.1 Object Description

Below is given a list of objects typically used by Homing Mode.

Controlword of Homing Mode

15 9	8	7	6 5	4	3 0
(See 5.2)	Halt	(See 5.2)	Reserved	Homing Operation Start	(See 5.2)

MSB LSB

Name	Value	Description
Homing Operation Start	0	Homing mode inactive
	0 → 1	Start Homing Mode
	1	Homing Mode Active
	1 → 0	Interrupt Homing Mode
Halt	0	Execute the instruction of bit 4
	1	Stop axle with homing acceleration

Statusword of Homing Mode

15 14	13	12	11	10	9 0
(See 5.2)	Homing Error	Homing Attained	(See 5.2)	Target Reached	(See 5.2)

MSB LSB

Homing Error	Homing Attained	Description
0	0	Homing operation not jet completed
0	1	Homing operation carried out successfully
1	0	Homing operation not successfully carried out
1	1	Prohibited condition

Object 607Ch: Homing Offset
The homing offset object determines the difference between the zero position for the application and the machine home position, it is measured in position units. 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*. All subsequent absolute moves shall be taken relative to this new zero position. This is illustrated in the following diagram:

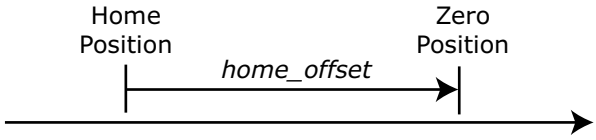


Figure 5.7: Homing Offset

OBJECT DESCRIPTION

INDEX	607Ch
Name	Home Offset
Object Code	VAR
Data Type	INTEGER32
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	INTEGER32
Default Value	0

Object 6098h: Homing Method
The homing method object determines the method that will be used during homing. See the "Functional Description" for further information.

OBJECT DESCRIPTION

INDEX	6098h
Name	Home Method
Object Code	VAR
Data Type	INTEGER8
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	INTEGER8
Default Value	3

The following table illustrates the ***homing methods*** implemented by Axor:

Homing Methods	Description
Homing Method 0	no homing
Homing Method 3	cw (clockwise) homing with NO (normally open) sensor + zero encoder
Homing Method 4	ccw (counterclockwise) homing with NC (normally close) sensor + zero encoder
Homing Method 5	ccw homing with NO sensor + zero encoder
Homing Method 6	cw homing with NC sensor + zero encoder
Homing Method 7	cw homing with NO sensor
Homing Method 8	ccw homing with NC sensor
Homing Method 9	ccw homing with NO sensor
Homing Method 10	cw homing with NC sensor
Homing Method 33	cw homing + zero encoder
Homing Method 34	ccw homing + zero encoder
Homing Method 35	immediate homing

See the "5.5.2 Functional Description" for further information.

Object 6099h: Homing Speeds
This entry in the object dictionary defines the speeds used during homing.

OBJECT DESCRIPTION

INDEX	6099h
Name	Home Speeds
Object Code	ARRAY
Data Type	UNSIGNED32
Category	Mandatory

ENTRY DESCRIPTION

Sub-Index	0
Description	Number of Entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2
Default Value	2

Sub-Index	1
Description	Speed during search for switch
Entry Category	Mandatory
Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	0

Sub-Index	2
Description	Speed during search for zero
Entry Category	Mandatory
Access	rw
PDO Mapping	Possible
Value Range	UNSIGNED32
Default Value	0

Object 609Ah: Homing Acceleration

OBJECT DESCRIPTION

INDEX	609Ah
Name	Homing Acceleration
Object Code	VAR
Data Type	UNSIGNED32
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	Unsigned32
Default Value	No

5.5.2 Functional Description

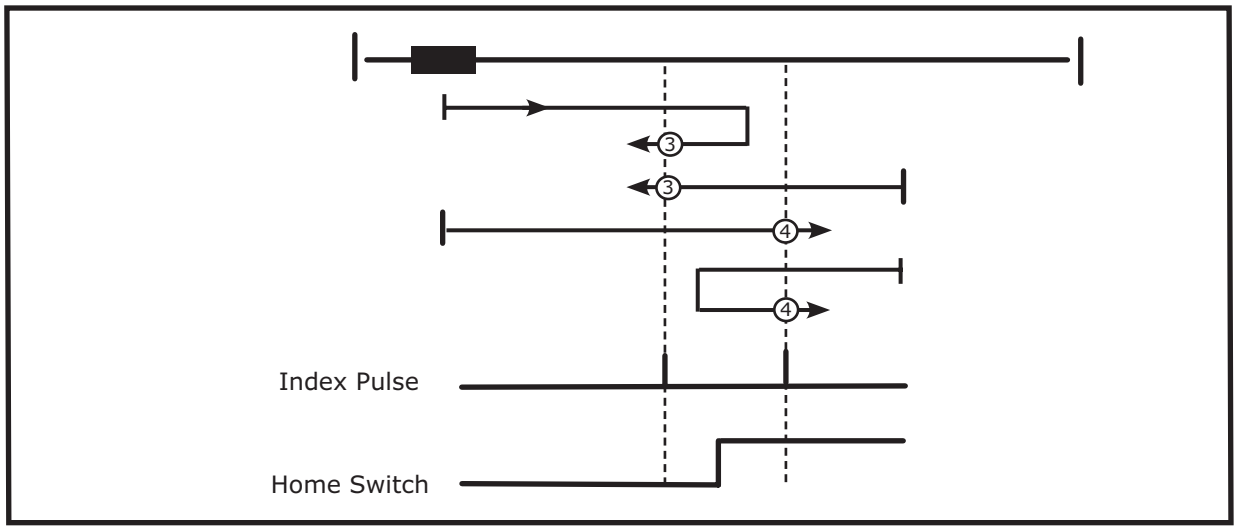
By choosing a method of homing by writing a value to *homing method* will clearly establish:

- the *homing signal*,
- the *direction of actuation* and where appropriate
- the *position of the index pulse*.

The homing methods implemented by Axor are the followings:

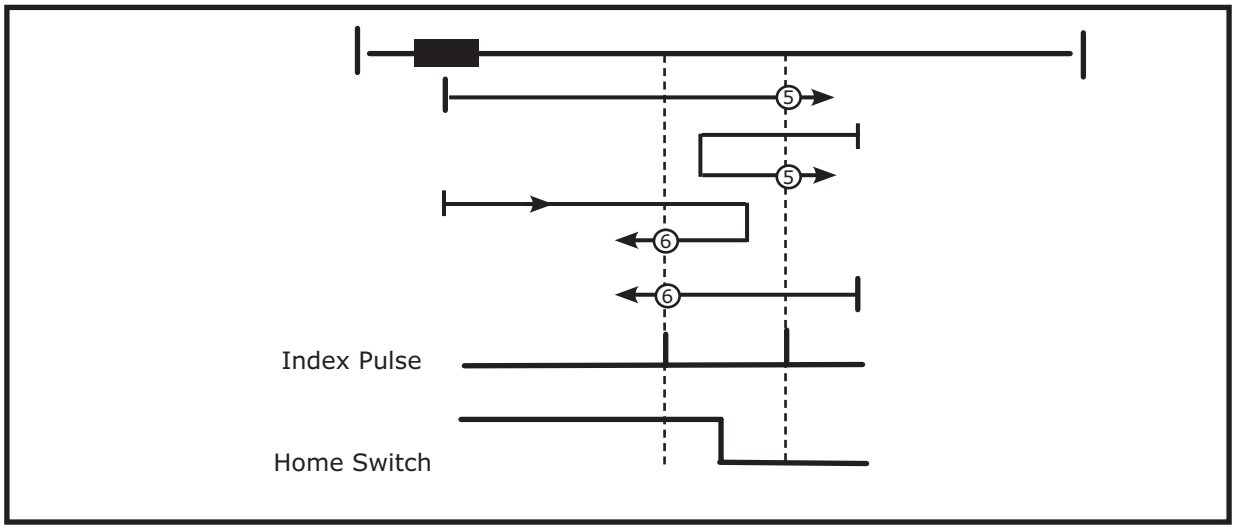
• Methods 3 and 4: Homing on the positive home switch and index pulse

The home position is at the index pulse to either to the left (method 3) or the right (method 4) 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.



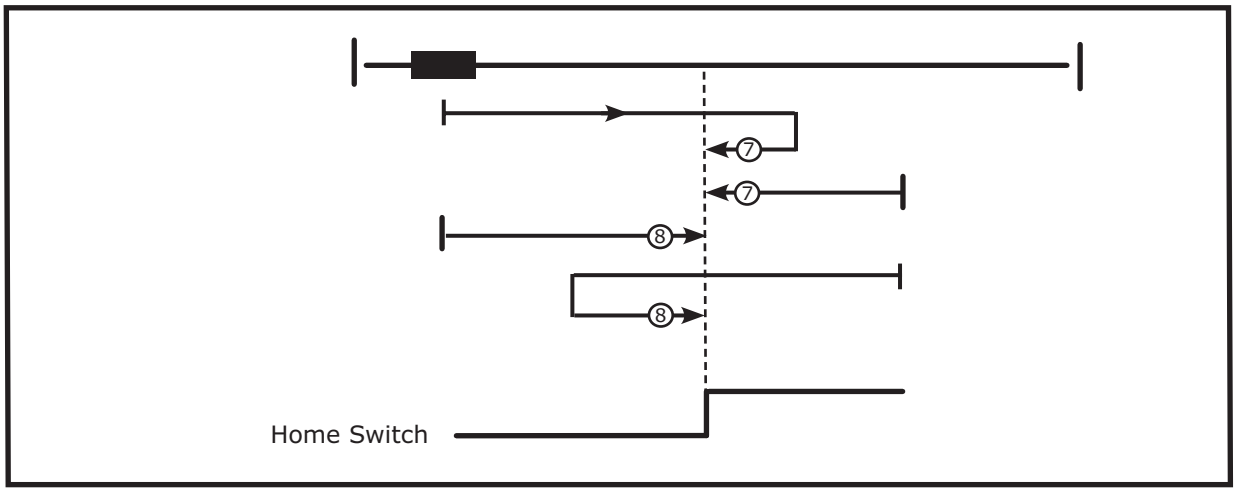
• Methods 5 and 6: Homing on the negative home switch and index pulse

The home position is at the index pulse to either to the left (method 5) or the right (method 6) 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.



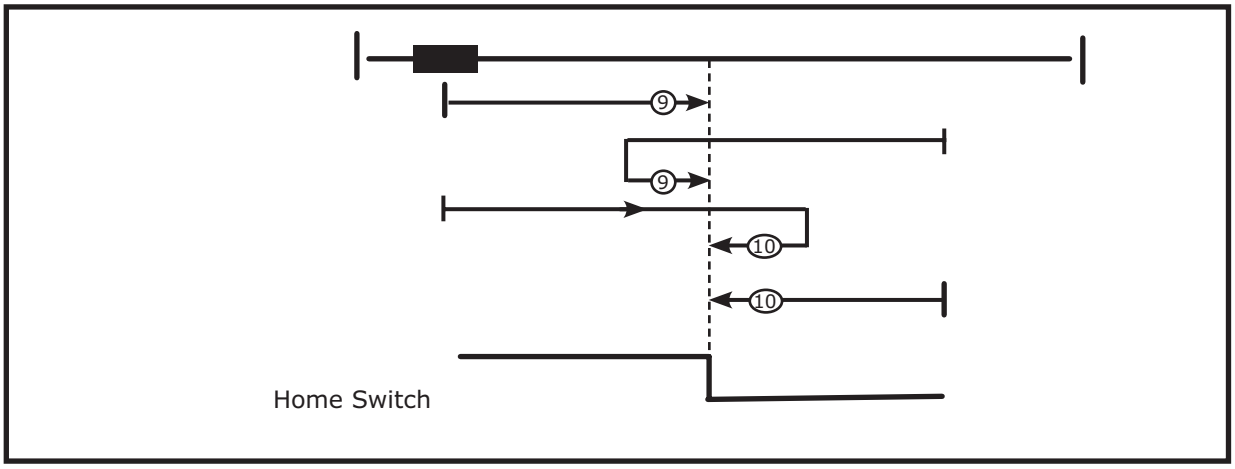
• Methods 7 and 8: Homing on the positive home switch

The home position is either to the left (method 7) or the right (method 8) 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.



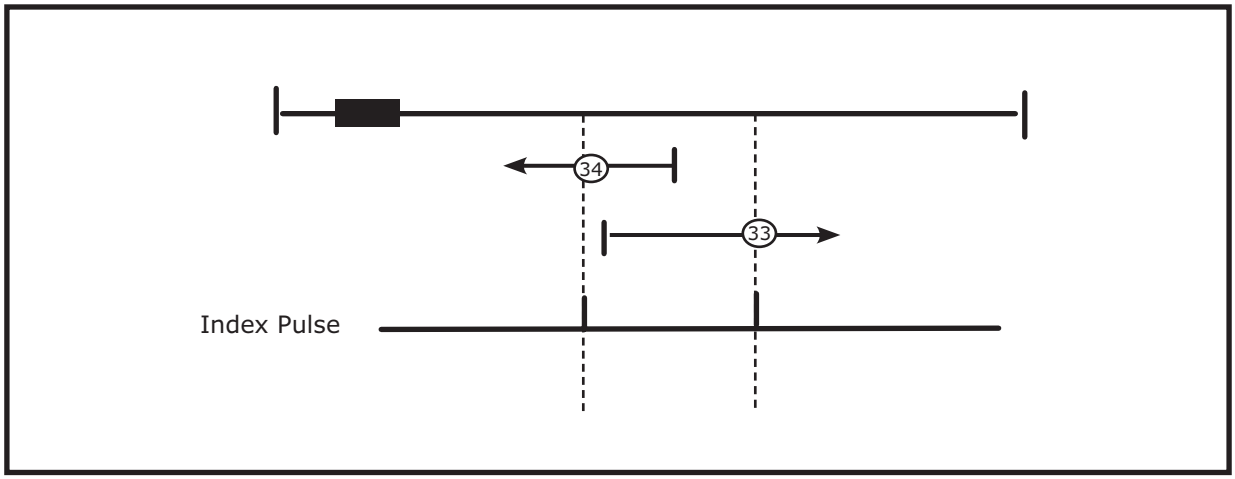
• Methods 9 and 10: Homing on the negative home switch

The home position is to either to the left (method 9) or the right (method 10) 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.



• Methods 33 and 34: Homing on the index pulse

The home position is at the index pulse found in the negative (34) or positive (33) direction respectively.



• Method 35: Homing on the current position

The current position is taken to be the home position.

5.6 – Profile Velocity Mode

The *target velocity* (60FFh) is applied to the *ramp generator.* This one produces a *velocity demand value* that is set as reference velocity for the speed loop. The ramp generator can be used only with trapezoidal ramps with different parameters for acceleration (6083h) and deceleration (6084h).

5.6.1 Object Description

Below is given a list of objects typically used by Profile Velocity Mode.

Controlword of Profile Velocity Mode

15 9	8	7	6 4	3 0
(See 5.2)	Halt	(See 5.2)	Reserved	(See 5.2)
MSB				LSB

Name	Value	Description
Halt	0	Execute the motion
	1	Stop axle

Statusword of Profile Velocity Mode

15 14	13	12	11	10	9 0
(See 5.2)	Max Slippage Error	Speed	(See 5.2)	Target Reached	(See 5.2)
MSB					LSB

Name	Value	Description
Target Reached	0	Halt = 0: *Target velocity* not (yet) reached
		Halt = 1: Axle decelerates
	1	Halt = 0: *Target velocity* reached
		Halt = 1: Axle has velocity 0
Speed	0	Speed is not equal to 0
	1	Speed is equal to 0
Max Slippage Error	0	Maximum slippage not reached
	1	Maximum slippage reached

Object 606Ch: Velocity Actual Value
The velocity actual value is used as input for the velocity controller.

OBJECT DESCRIPTION

INDEX	606Ch
Name	Velocity Actual Value
Object Code	VAR
Data Type	INTEGER32
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	INTEGER32
Default Value	No

Object 60FFh: Target Velocity
The target velocity is the input for the trajectory generator and the value is given in velocity units.
OBJECT DESCRIPTION

INDEX	60FFh
Name	Target Velocity
Object Code	VAR
Data Type	INTEGER32
Category	Mandatory

ENTRY DESCRIPTION

Access	rw
PDO Mapping	Possible
Value Range	INTEGER32
Default Value	No

5.7 – Interpolated Position Mode

The ***Interpolated Position Mode*** is used to control multiple coordinated axles or a single axle with the need for time-interpolation of setpoint data.
The Interpolated Position Mode uses sync object for a time coordination of the related drive units. The interpolation cycle time is defined by the object *interpolation_time_period*.

The Interpolated Position Mode allows a host controller to transmit a stream of interpolation data with either an implicit or explicit time reference to a drive unit. If the drive supports an input buffer, the interpolation data may be sent in bursts rather than continuously in real time.
The buffer size is the number of interpolation data records which may be sent to a drive to fill the input buffer and it is not the size in bytes. Devices without input buffer capabilities have to accept at least one interpolation data item.

The interpolation algorithm is defined in the *interpolation_submode_select*. Only linear interpolation method is develop. This requires only one interpolation data item to be buffered for the calculation of the next demand value. For each interpolation cycle, the drive will calculate a *position_demand_value* by interpolating positions over a period of time.

5.7.1 Object Description

Below is given a list of objects typically used by Interpolated Position Mode.

Object 60C0h: Interpolation Sub Mode Select

OBJECT DESCRIPTION

INDEX	60C0h
Name	Interpolation Sub Mode select
Object Code	VAR
Data Type	INTEGER16
Category	Optional

ENTRY DESCRIPTION

Access	rw
PDO Mapping	No
Value Range	INTEGER16
Default Value	0: Linear Interpolation

Object 60C1h: Interpolation Data Record

OBJECT DESCRIPTION

INDEX	60C1h
Name	Interpolation Data Record
Object Code	RECORD
Category	Optional

ENTRY DESCRIPTION

Sub-Index	0h
Description	Number of Entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1
Default Value	1

Sub-Index	1h
Description	The first parameter if ip function
Entry Category	Mandatory
Access	rw
PDO Mapping	Possible
Data Type	UNSIGNED32
Default Value	No

Object 60C2h: Interpolation Time Period

OBJECT DESCRIPTION

INDEX	60C2h
Name	Interpolation Time Period
Object Code	RECORD
Category	Optional

ENTRY DESCRIPTION

Sub-Index	0h
Description	Number of Entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2
Default Value	2

Sub-Index	1h
Description	Interpolation Time Units
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Data Type	UNSIGNED8
Default Value	0

Sub-Index	2h
Description	Interpolation Time Index
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Data Type	INTEGER8
Default Value	-3

Object 60C3h: Interpolation Sync Definition

OBJECT DESCRIPTION

INDEX	60C3h
Name	Interpolation Sync Definition
Object Code	ARRAY
Data Type	UNSIGNED8
Category	Optional

ENTRY DESCRIPTION

Sub-Index	0h
Description	Number of Entries
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2
Default Value	2

Sub-Index	1h
Description	Synchronize on group
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Data Type	UNSIGNED8
Default Value	0

Sub-Index	2h
Description	Ip sync every n event
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Data Type	UNSIGNED8
Default Value	1

Chapter 6
Appendix

6.1 Examples

In the following pages we illustrate some examples.
All examples need first to send a NMT by using the "Start Remote Node" request (see chapter 3). Doing this there is the translation from the "Pre-Operational" state to the "Operational".

6.1.1 Example 1: Positioning

Suppose we want to execute a "positioning". We need the following objects: Target Position (object 607Ah), Profile Velocity (object 6081h), Profile Acceleration (object 6083h), Profile Deceleration (object 6084h), Modes of Operation (object 6060h), Controlword (object 6040h).
The sequence is the following:

1- set the Target Position:

Object Name	Object Index	Value inserted (hex)
Target Position	607Ah.0h	0089 9680h

2- set the Profile Velocity:

Object Name	Object Index	Value inserted (hex)
Profile Velocity	6081h.0h	0002 0B8Bh

3- set the Profile Acceleration:

Object Name	Object Index	Value inserted (hex)
Profile Acceleration	6083h.0h	005B 0009h

4- set the Profile Deceleration:

Object Name	Object Index	Value inserted (hex)
Profile Deceleration	6084h.0h	001E 8084h

5- set the Profile Position Mode:

Object Name	Object Index	Value inserted (hex)
Modes of Operation	6060h.0h	01h

6- execute the transition from the state "Switch On Disable" to "Ready to Switch On", writing in the controlword the value 0006h:

Object Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	0006h

7- execute the transition from the state "Ready to Switch On" to "Switch On", writing in the controlword the value 0007h:

Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	0007h

8- execute the transition from the state "Switch On" to "Operation Enable", writing in the controlword the value 000Fh:

Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	000Fh

9- start the positioning writing in the controlword the value 001Fh:

Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	001Fh

6.1.2 Example 2: Homing Procedure

Suppose we want to execute a "homing procedure". We need the following objects: Modes of Operation (object 6060h), Homing Method (object 6058h), Controlword (object 6040h). The sequence is the following:

1- set the Homing mode of operation:

Object Name	Object Index	Value inserted (hex)
Modes of Operation	6060h.0h	06h

2- set the "Immediate" homing method (method 35):

Object Name	Object Index	Value inserted (hex)
Homing Method	6098h.0h	23h

3- execute the transition from the state "Switch On Disable" to "Ready to Switch On", writing in the controlword the value 0006h:

Object Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	0006h

4- execute the transition from the state "Ready to Switch On" to "Switch On", writing in the controlword the value 0007h:

Object Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	0007h

5- execute the transition from the state "Switch On" to "Operation Enable", writing in the controlword the value 000Fh:

Object Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	000Fh

6- start the homing writing in the controlword the value 001Fh:

Object Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	001Fh

6.1.3 Example 3: Speed Control

Suppose we want to "applay a speed reference". We need the following objects: Modes of Operation (object 6060h), Controlword (object 6040h), Target Velocity (object 60FFh).
The seguence is the following:

1- set the Profile Velocity Mode:

Object Name	Object Index	Value inserted (hex)
Modes of Operation	6060h.0h	03h

2- execute the transition from the state "Switch On Disable" to "Ready to Switch On", writing in the controlword the value 0006h:

Object Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	0006h

3- execute the transition from the state "Ready to Switch On" to "Switch On", writing in the controlword the value 0007h:

Object Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	0007h

4- execute the transition from the state "Switch On" to "Operation Enable", writing in the controlword the value 000Fh:

Object Name	Object Index	Value inserted (hex)
Controlword	6040h.0h	000Fh

Attention: this command causes the start of the moving.

5- set the Target Velocity:

Object Name	Object Index	Value inserted (hex)
Target Positon	60FFh.0h	0000 1000h

6.2 Objects' list

6061h	Modes of Operation Display	95
6064h	Position Actual Value	80
6067h	Position Window	97
6068h	Position Window Time	98
606Ch	Velocity Actual Value	109
6073h	Max Current Value	80
6075h	Motor Rated Current	81
607Ah	Target Position	99
607Ch	Home Offset	103
607E	Polarity	81
607Fh	Max Profile Velocity	82
6081h	Profile Velocity	100
6083h	Profile Acceleration	82
6084h	Profile Deceleration	83
6085h	Quick Stop Deceleration	83
6089h	Position Notation Index	84
608Ah	Position Dimension Index	84
608Bh	Velocity Notation Index	85
608Ch	Velocity Dimension Index	85
608Dh	Acceleration Notation Index	86
608Eh	Acceleration Dimension Index	86
6092h	Feed Constant	87
6098h	Homing Method	103
6099h	Homing Speed	104
609Ah	Homing Acceleration	105
60C0h	Interpolation Sub Mode Select	111
60C1h	Interpolation Data Record	111
60C2h	Interpolation Time Period	112
60C3h	Interpolation Sync Definition	113
60F4h	Following error actual value	88
60F9h	Velocity Control Parameter Set	88
60FBh	Position Control Parameter Set	90
60FDh	Digital Inputs	91
60FFh	Target Velocity	109

Since AXOR INDUSTRIES MOTORS SERVO DRIVES *1988*

AXOR INDUSTRIES®

viale **Stazione, 5**
36054 Montebello Vic.
Vicenza - Italy

phone **(+39) 0444 440441**
fax **(+39) 0444 440418**
info@axorindustries.com

www.axorindustries.com

CERTIFIED UNI EN ISO 9001 CERTIFIED N.9135 CERTIFIED

CSQ CERTIFICAZIONE DEI SISTEMI DI QUALITA' DELLE AZIENDE
UNI EN ISO 9001 CERTIFIED N° 9135