NiLAB
Neue innovative Linearantriebe

CANopen – Object Dictionary

Implementation guide

Date: 19/05/2022

Revision: 1.00

Date	Note
19/05/22	First version

NiLAB GmbH
Hans-Sachs-Straße 16
9020 Klagenfurt am Wörthersee (AT)
Tel: +43 720 513 258 (VoIP)
http://www.nilab.at

Index

1 Introduction

This document must be considered as integral port of the official documents relating to CANopen standard:

- CiA CANopen – Device Profile Drives and Motion Control – DSP 402 v1.1, which will be referred to DSP 402;
- CiA CANopen - Application Layer and Communication Profile – DS 301 v4.01, which will be referred to DSP 301.

1.1 Main features

NMT	Slave
Error checking	Node guarding, heartbeat
Node ID:	Parameter saved in the EEPROM of the drive
Bitrate:	Parameter saved in the EEPROM of the drive
Number of PDO	Four in reception (RPDO) – four in transmission (TPDO)
PDO modalities	Synchronous (cyclic and acyclic), asynchronous
PDO linking	Yes
PDO mapping	Variable, up to eight entities for PDO
Number of SDO:	One server – Zero client
Emergency messages	Yes
CANopen version	DS 301 v4.01
Device Profile	DSP 402 v1.1
Modalities	Profile position mode
	Velocity mode
	Homing mode
	Interpolated position mode

1.2 Abbreviations and Terms

CAN	Controller Area Network
CiA	CAN in Automation
COB	Communication Object is the transport unit. The data must be transmitted over the CAN network in a COB
COB-ID	COB Identifier: identifies uniquely a COB over the net. The identifier determines the priority of the COP in MAX sub-level
MAC	Medium Access Control: it is one of the sub-levels of the CAN Data Link Layer and its function is to arbitrate the access to the bus
PDO	Process Data Object
SDO	Service Data Object
IDO	Internal Data Object
EDS	Electronic Data Sheet
NMT	Network Management
OD	Object Dictionary
PDS	Power Drive System
RPDO	Receive (incoming) PDO
TPDO	Transmit (outgoing) PDO
ro	Read-Only
rw	Read-Write
co	Constant, read-only

NB:

- the numbers in hexadecimal base are prefixed with "0x" and are indicated by the subscript "h" ;
- the numbers without prefix are in decimal base.

2 CANopen Object Dictionary

The object dictionary is essentially a grouping of objects accessible via the network in an ordered predefined fashion.
Each object within the object dictionary is addressed using:

- a 16 bit **index**;
- an 8 bit **Sub index.**

2.1 0x1000 to 0x1018 : Communication object

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
0x1000$_h$	0	Device type	UINT32	ro	N	0x00020192	--
0x1001$_h$	0	Error register	UINT8	ro	N	0	--
0x1002$_h$	0	Manufacturer Status Register	UINT32	ro	N		
0x1003$_h$ - Pre-defined error field							
0x1003$_h$	0	Number of Errors	UINT8	ro	N	0 to 4	--
0x1003$_h$	1	Standard error field	UINT32	ro	N	0	--
--							
0x1005$_h$	0	COB-ID SYNC message	UINT32	rw	N	0x00000080	--
0x1006$_h$	0	Communication Cycle Period	UINT32	rw	N	0x00000000	--
0x1007$_h$	0	Synchronous Window Length	UINT32	rw	N	0x00000000	--
0x1008$_h$	0	Manufacturer device name	vSTRING	co	N		--
0x1009$_h$	0	Manufacturer hardware version	vSTRING	co	N	“Last version”	--
0x100A$_h$	0	Manufacturer software version	vSTRING	co	N	“Last version”	--
0x100C$_h$	0	Guard time	UINT16	rw	N	0	--
0x100D$_h$	0	Life time factor	UINT8	rw	N	0	--
0x1014$_h$	0	COB ID Emergency	UINT32	rw	N	NODE_ID+0x00000080	--
0x1015$_h$	0	Inhibit Time EMCY	UINT16	rw	N	0	--
0x1016$_h$ - Consumer heartbeat time							
0x1016$_h$	0	Highest sub-index supported	UINT8	ro	N	8	--
0x1016$_h$	1	Consumer heartbeat time	UINT32	rw	N	0x00000000	--
--							
0x1017$_h$	0	Producer heartbeat time	UINT16	rw	N	0	--
0x1018$_h$ - Identity Record							
0x1018$_h$	0	Number of entries	UINT8	ro	N	0x04	--
0x1018$_h$	1	Vendor ID	UINT32	ro	N	0x00001A21	--
0x1018$_h$	2	Product Code	UINT32	ro	N	4	--
0x1018$_h$	3	Revision Number	UINT32	ro	N	0x00000001	--
0x1018$_h$	4	Serial Number	UINT32	ro	N	0	--

2.2 0x1200 to 0x127F : SDO Server Parameter

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
0x1200$_h$ - Server SDO #1							
0x1200$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1200$_h$	1	COB_ID Client->Server (rx)	UINT32	ro	N	NODE_ID+0x600	--
0x1200$_h$	2	COB_ID Server ->Client (tx)	UINT32	ro	N	NODE_ID+0x580	--
0x1200$_h$ - Server SDO #2							
0x1201$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1201$_h$	1	COB_IDExt Client->Server (rx)	UINT32	ro	N	NODE_ID+0x00000600	--
0x1201$_h$	2	COB_IDExt Server ->Client (tx)	UINT32	ro	N	NODE_ID+0x00000580	--

2.3 0x1400 to 0x15FF: RPDO Communication Parameter

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
0x1400$_h$ - RPDO Communication Parameter #1							
0x1400$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1400$_h$	1	COB-ID used by PDO	UINT32	rw	N	NODE_ID+0x00000200	--
0x1400$_h$	2	Transmission Type	UINT8	rw	N	0xFF	--
0x1400$_h$	3	Inhibit Time	UINT16	rw	N	--	--
0x1400$_h$	5	Event Timer	UINT16	rw	N	--	--
0x1401$_h$ - RPDO Communication Parameter #2							
0x1401$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1401$_h$	1	COB-ID used by PDO	UINT32	rw	N	NODE_ID+0x00000300	--
0x1401$_h$	2	Transmission Type	UINT8	rw	N	0xFF	--
0x1401$_h$	3	Inhibit Time	UINT16	rw	N	--	--
0x1401$_h$	5	Event Timer	UINT16	rw	N	--	--
0x1402$_h$ - RPDO Communication Parameter #3							
0x1402$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1402$_h$	1	COB-ID used by PDO	UINT32	rw	N	NODE_ID+0x00000400	--
0x1402$_h$	2	Transmission Type	UINT8	rw	N	0xFF	--
0x1402$_h$	3	Inhibit Time	UINT16	rw	N	--	--
0x1402$_h$	5	Event Timer	UINT16	rw	N	--	--
0x1403$_h$ - RPDO Communication Parameter #4							
0x1403$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1403$_h$	1	COB-ID used by PDO	UINT32	rw	N	NODE_ID+0x00000500	--
0x1403$_h$	2	Transmission Type	UINT8	rw	N	0xFF	--
0x1403$_h$	3	Inhibit Time	UINT16	rw	N	--	--
0x1403$_h$	5	Event Timer	UINT16	rw	N	--	--

2.4 0x1600 to 0x17FF: RPDO Mapping Parameter

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
0x1600$_h$ - RPDO 1 Mapping Parameter #1							
0x1600$_h$	0	Number of mapped objects	UINT8	rw	N	0	--
0x1600$_h$	1	#1 Mapped object	UINT32	rw	N	--	--
0x1600$_h$	2	#2 Mapped object	UINT32	rw	N	--	--
0x1600$_h$	3	#3 Mapped object	UINT32	rw	N	--	--
0x1600$_h$	4	#4 Mapped object	UINT32	rw	N	--	--
0x1601$_h$ - RPDO Mapping Parameter #2							
0x1601$_h$	0	Number of mapped objects	UINT8	rw	N	0	--
0x1601$_h$	1	#1 Mapped object	UINT32	rw	N	--	--
0x1601$_h$	2	#2 Mapped object	UINT32	rw	N	--	--
0x1601$_h$	3	#3 Mapped object	UINT32	rw	N	--	--
0x1601$_h$	4	#4 Mapped object	UINT32	rw	N	--	--
0x1602$_h$ - RPDO Mapping Parameter #3							
0x1602$_h$	0	Number of mapped objects	UINT8	rw	N	0	--
0x1602$_h$	1	#1 Mapped object	UINT32	rw	N	--	--
0x1602$_h$	2	#2 Mapped object	UINT32	rw	N	--	--
0x1602$_h$	3	#3 Mapped object	UINT32	rw	N	--	--
0x1602$_h$	4	#4 Mapped object	UINT32	rw	N	--	--
0x1603$_h$ - RPDO Mapping Parameter #4							
0x1603$_h$	0	Number of mapped objects	UINT8	rw	N	0	--
0x1603$_h$	1	#1 Mapped object	UINT32	rw	N	--	--
0x1603$_h$	2	#2 Mapped object	UINT32	rw	N	--	--
0x1603$_h$	3	#3 Mapped object	UINT32	rw	N	--	--
0x1603$_h$	4	#4 Mapped object	UINT32	rw	N	--	--

2.5 0x1800 to 0x19FF: TPDO Communication parameter

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
0x1800$_h$ - TPDO Communication Parameter #1							
0x1800$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1800$_h$	1	COB-ID used by PDO	UINT32	rw	N	NODE_ID+0x00000180	--
0x1800$_h$	2	Transmission Type	UINT8	rw	N	0xFF	--
0x1800$_h$	3	Inhibit Time	UINT16	rw	N	--	--
0x1800$_h$	5	Event Timer	UINT16	rw	N	--	--
0x1801$_h$ - TPDO Communication Parameter #2							
0x1801$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1801$_h$	1	COB-ID used by PDO	UINT32	rw	N	NODE_ID+0x00000280	--
0x1801$_h$	2	Transmission Type	UINT8	rw	N	0xFF	--
0x1801$_h$	3	Inhibit Time	UINT16	rw	N	--	--
0x1801$_h$	5	Event Timer	UINT16	rw	N	--	--
0x1802$_h$ - TPDO Communication Parameter #3							
0x1802$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1802$_h$	1	COB-ID used by PDO	UINT32	rw	N	NODE_ID+0x00000380	--
0x1802$_h$	2	Transmission Type	UINT8	rw	N	0xFF	--
0x1802$_h$	3	Inhibit Time	UINT16	rw	N	--	--
0x1802$_h$	5	Event Timer	UINT16	rw	N	--	--
0x1803$_h$ - TPDO Communication Parameter #4							
0x1803$_h$	0	Number of entries	UINT8	ro	N	0x02	--
0x1803$_h$	1	COB-ID used by PDO	UINT32	rw	N	NODE_ID+0x00000480	--
0x1803$_h$	2	Transmission Type	UINT8	rw	N	0xFF	--
0x1803$_h$	3	Inhibit Time	UINT16	rw	N	--	--
0x1803$_h$	5	Event Timer	UINT16	rw	N	--	--

2.6 0x1A00 to 0x1BFF: TDPO Mapping Parameter

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
0x1A00$_h$ - TPDO Communication Parameter #1							
0x1A00$_h$	0	Number of entries	UINT8	rw	N	0	--
0x1A00$_h$	1	#1 Mapped object	UINT32	rw	N	--	--
0x1A00$_h$	2	#2 Mapped object	UINT32	rw	N	--	--
0x1A00$_h$	3	#3 Mapped object	UINT32	rw	N	--	--
0x1A00$_h$	4	#4 Mapped object	UINT32	rw	N	--	--
0x1A01$_h$ - TPDO Communication Parameter #2							
0x1A01$_h$	0	Number of entries	UINT8	rw	N	0	--
0x1A01$_h$	1	#1 Mapped object	UINT32	rw	N	--	--
0x1A01$_h$	2	#2 Mapped object	UINT32	rw	N	--	--
0x1A01$_h$	3	#3 Mapped object	UINT32	rw	N	--	--
0x1A01$_h$	4	#4 Mapped object	UINT32	rw	N	--	--
0x1A02$_h$ - TPDO Communication Parameter #3							
0x1A02$_h$	0	Number of entries	UINT8	rw	N	0	--
0x1A02$_h$	1	#1 Mapped object	UINT32	rw	N	--	--
0x1A02$_h$	2	#2 Mapped object	UINT32	rw	N	--	--
0x1A02$_h$	3	#3 Mapped object	UINT32	rw	N	--	--
0x1A02$_h$	4	#4 Mapped object	UINT32	rw	N	--	--
0x1A03$_h$ - TPDO Communication Parameter #4							
0x1A03$_h$	0	Number of entries	UINT8	rw	N	0	--
0x1A03$_h$	1	#1 Mapped object	UINT32	rw	N	--	--
0x1A03$_h$	2	#2 Mapped object	UINT32	rw	N	--	--
0x1A03$_h$	3	#3 Mapped object	UINT32	rw	N	--	--
0x1A03$_h$	4	#4 Mapped object	UINT32	rw	N	--	--

2.7 Manufacturer-Specific Profile Area

2.8 0x2000 to 0x3240: Parameters defined by the manufacturer

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
$0x2000_h$ → $0x2400_h$	0	Parameters	UINT16	rw	Y	--	--
$0x3000_h$ → $0x3040_h$	0	Input Variables	UINT16	rw	Y	--	--
$0x3200_h$ → $0x3240_h$	0	Output Variables	UINT16	rw	Y	--	--

2.9 0x3FFC to 0x3FFF: Modbus over CANopen Gateway (MOC)

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
*$0x3FFC_h$	0	MOC_Server->Packet	--	rw	N	--	--
*$0x3FFD_h$	0	MOC_Server->Packet	--	rw	N	--	--
*$0x3FFE_h$	0	MOC_Server->Packet	--	rw	N	--	--
*$0x3FFF_h$	0	MOC_Client->Packet	--	rw	N	--	--

2.10 0x5000 to 0x60FF: Standard Profile area

Index	SubIndex	Name	Type	Access	Mappable?	Default value	Units
0x5000$_h$	0	Out Position latch count	UINT16	rw	Y	0	--
0x5010$_h$	0	Slave control Word	UINT16	rw	Y	0	
0x5011$_h$	0	Slave Target Speed rpm	UINT16	rw	Y	0	
0x5020$_h$	0	Slave Status Word	UINT16	rw	Y	0	
0x5021$_h$	0	Slave Warning Word	UINT16	rw	Y	0	
0x5030$_h$	0	Master Warning Word	UINT16	rw	Y	0	
0x5050$_h$	0	Comando Asse X	UINT16	rw	Y	0	
0x5051$_h$	0	Comando Asse Y	UINT16	rw	Y	0	
0x5052$_h$	0	Dummy	UINT16	rw	Y	0	
From 0x5100$_h$ To 0x513F$_h$ – App Share							
0x5100$_h$ → 0x513F$_h$	--	Reserved for APP Share	UINT16	rw	Y	0	
--							
0x5200$_h$	0	Delta deceleration	UINT16	rw	Y	0	
0x5201$_h$	0	Delta acceleration	UINT16	rw	Y	0	
0x6040$_h$	0	Control word	UINT16	rw	Y	0	--
0x6041$_h$	0	Status word	UINT16	ro	Y	0	--
0x605D$_h$	0	Halt option code	INT16	rw	N	0x01	--
0x6060$_h$	0	Modes of operation	INT8	rw	Y	3	
0x6061$_h$	0	Modes of operation display	INT8	ro	Y	3	
0x6064$_h$	0	Position actual value	INT32	ro	Y	0	
0x6065$_h$	0	Following error window	UINT32	rw	Y	0xFFFFFFFF	[disable]
0x6066$_h$	0	Following error time out	UINT16	rw	Y	0	ms
0x6067$_h$	0	Position Window	UINT32	rw	Y	0xFFFFFFFF	[disable]
0x6068$_h$	0	Position Window Time Out	UINT16	rw	Y	0	ms
0x606C$_h$	0	Velocity actual value	INT32	ro	Y	0	
0x6073$_h$	0	Max Current	UINT16	rw	Y	1000	
0x607A$_h$	0	Target position	INT32	rw	Y	0	
0x607C$_h$	0	Home Offset	INT32	rw	Y	0	
0x607D$_h$ - Software Position Limit							
0x607D$_h$	0	Number of mapped objects	UINT8	ro	N	0x02	
0x607D$_h$	1	Min Position Limit	INT32	rw	Y	0x7FFFFFFF	
0x607D$_h$	2	Max Position Limit	INT32	rw	Y	0x80000000	
--							
0x607E$_h$	0	Polarity	UINT8	rw	Y	0	
0x607F$_h$	0	Max Profile Velocity	UINT32	rw	Y	0xFFFFFFFF	
0x6081$_h$	0	Profile velocity	UINT32	rw	Y	0	
0x6083$_h$	0	Profile acceleration	UINT32	rw	Y	0	
0x6084$_h$	0	Profile deceleration	UINT32	rw	Y	0	
0x6085$_h$	0	Quick stop deceleration	UINT32	rw	Y	0	

0x6086$_h$	0	Motion profile type	INT16	ro	Y	0	
0x6089$_h$	0	Position notation index	INT8	rw	Y	0xFA	
0x608A$_h$	0	Position dimension index	UINT8	rw	Y	0x01	
0x608B$_h$	0	Velocity notation index	INT8	rw	Y	0xFA	
0x608C$_h$	0	Velocity dimension index	UINT8	rw	Y	0xA6	
0x608D$_h$	0	Acceleration notation index	INT8	rw	Y	0xFA	
0x608E$_h$	0	Acceleration dimension index	UINT8	rw	Y	0xA6	
0x6092h - Feed coant							
0x6092$_h$	0	NrOfObjects	UINT8	ro	N	2	
0x6092$_h$	1	Feed	UINT32	rw	Y	65536	
0x6092$_h$	2	Shaft revolutions	UINT32	rw	Y	1	
--							
0x6098$_h$	0	Homing method	INT8	rw	Y	35	
0x6099h - Homing speeds							
0x6099$_h$	0	Number of mapped objects	UINT8	ro	N	2	
0x6099$_h$	1	Speed during search for switch	UINT32	rw	Y	0	
0x6099$_h$	2	Speed during search for zero (not used)	UINT32	rw	Y	0	
--							
0x609A$_h$	0	Homing acceleration	UINT32	rw	Y	0	
0x60C1h - Interpolation data record							
0x60C1$_h$	0	Number of mapped objects	UINT8	ro	N	0x01	
0x60C1$_h$	1	Position setpoint	INT32	rw	Y	0x00	
0x60C2h - Interpolation time period							
0x60C2$_h$	0	Number of mapped objects	UINT8	ro	N	0x02	
0x60C2$_h$	1	Interpolation time units	UINT8	rw	N	2	
0x60C2$_h$	2	Interpolation time index	INT8	ro	Y	-2	
0x60C3$_h$ - Interpolation Sync definition							
0x60C3$_h$	0	Number of mapped objects	UINT8	ro	N	0x02	
0x60C3$_h$	1	Interpolation sync units	UINT8	rw	N	0	
0x60C3$_h$	2	Interpolation Sync index	INT8	rw	N	1	
--							
0x60C5$_h$	0	Max acceleration	UINT32	rw	Y	0xFFFFFFFF	
0x60C6$_h$	0	Max deceleration	UINT32	rw	Y	0xFFFFFFFF	
0x60F4$_h$	0	Following error actual value	INT32	ro	Y		
0x60FD$_h$	0	Digital inputs	UINT32	ro	Y		
0x60FE$_h$ - Digital outputs							
0x60FE$_h$	0	Number of objects	UINT8	ro	N	2	
0x60FE$_h$	1	Physical outputs	UINT32	rw	Y		
0x60FE$_h$	2	Bit mask	UINT32	rw	Y		
0x60FF$_h$	0	Target velocity	INT32	rw	Y		
0x6300$_h$ - Cam state register							
0x6300$_h$	0	Number of objects	UINT8	ro	N	0x01	
0x6300$_h$	1	Channel 1	UINT8	ro	Y		
0x6301$_h$ - Cam enable							
0x6301$_h$	0	Number of objects	UINT8	ro	N	1	

0x6301$_h$	1	Channel 1	UINT8	rw	Y		
0x6302$_h$ - Cam polarity							
0x6302$_h$	0	Number of objects	UINT8	ro	N	1	
0x6302$_h$	1	Channel 1	UINT8	rw	Y		
0x6310$_h$ - Cam 1 low limit							
0x6310$_h$	0	Number of objects	UINT8	ro	N	1	
0x6310$_h$	1	Channel 1	INT32	rw	N		
0x6311$_h$ - Cam 2 low limit							
0x6311$_h$	0	Number of objects	UINT8	ro	N	1	
0x6311$_h$	1	Channel 1	INT32	rw	N		
0x6312$_h$ - Cam 3 low limit							
0x6312$_h$	0	Number of objects	UINT8	ro	N	1	
0x6312$_h$	1	Channel 1	INT32	rw	N		
0x6313$_h$ - Cam 4 low limit							
0x6313$_h$	0	Number of objects	UINT8	ro	N	1	
0x6313$_h$	1	Channel 1	INT32	rw	N		
0x6314$_h$ - Cam 5 low limit							
0x6314$_h$	0	Number of objects	UINT8	ro	N	1	
0x6314$_h$	1	Channel 1	INT32	rw	N		
0x6315$_h$ - Cam 6 low limit							
0x6315$_h$	0	Number of objects	UINT8	ro	N	1	
0x6315$_h$	1	Channel 1	INT32	rw	N		
0x6316$_h$ - Cam 7 low limit							
0x6316$_h$	0	Number of objects	UINT8	ro	N	1	
0x6316$_h$	1	Channel 1	INT32	rw	N		
0x6317$_h$ - Cam 8 low limit							
0x6317$_h$	0	Number of objects	UINT8	ro	N	1	
0x6317$_h$	1	Channel 1	INT32	rw	N		
0x6320$_h$ - Cam 1 high limit							
0x6320$_h$	0	Number of objects	UINT8	ro	N	1	
0x6320$_h$	1	Channel 1	INT32	rw	N		
0x6321$_h$ - Cam 2 high limit							
0x6321$_h$	0	Number of objects	UINT8	ro	N	1	
0x6321$_h$	1	Channel 1	INT32	rw	N		
0x6322$_h$ - Cam 3 high limit							
0x6322$_h$	0	Number of objects	UINT8	ro	N	1	
0x6322$_h$	1	Channel 1	INT32	rw	N		
0x6323$_h$ - Cam 4 high limit							
0x6323$_h$	0	Number of objects	UINT8	ro	N	1	
0x6323$_h$	1	Channel 1	INT32	rw	N		
0x6324$_h$ - Cam 5 high limit							
0x6324$_h$	0	Number of objects	UINT8	ro	N	1	
0x6324$_h$	1	Channel 1	INT32	rw	N		
0x6325$_h$ - Cam 6 high limit							
0x6325$_h$	0	Number of objects	UINT8	ro	N	1	
0x6325$_h$	1	Channel 1	INT32	rw	N		

0x6326$_h$ - Cam 7 high limit							
0x6326$_h$	0	Number of objects	UINT8	ro	N	1	
0x6326$_h$	1	Channel 1	INT32	rw	N		
0x6327$_h$ - Cam 8 high limit							
0x6327$_h$	0	Number of objects	UINT8	ro	N	1	
0x6327$_h$	1	Channel 1	INT32	rw	N		

3 CANopen device

3.1 Introduction

The CANopen protocol specifies the services offered by the CAN Application Layer (CAL), corresponding to the seventh level of the ISO/OSI reference model.

The CAL offers many services that can be grouped by:

- **CAN-based Message Specification** (**CMS**): it is a language used to describe the COBs that are used in the device. It also includes rules for data encoding;
- **Network Management** (**NMT**): it allows to manage the initialization, configuration and error management of the network nodes;
- **Distributor** (**DBT**): manages the allocation of the identifiers of the COBs that are used by the CMS services;
- **Layer Management** (**LMT**): allows you to configure the parameters of the network.

3.2 Profiles

The profile defines a subset of services defined by the communication protocol, restricting their range. In particular, the CANopen specifications are composed of profiles based on the CAN reference model.

CANopen profiles are mainly two:

- **CANopen Communication Profile** (**DS 301**): every device is equipped with this profile that realizes the interface between CAN and CAL;
- **CANopen Device Profile**: it defines how the functionalities of the device are reachable from the CAN bus and what Communication Profile it is necessary to use. It is a specific profile for the drive that makes its behaviour towards the CAN network unique, describing the basic mechanisms for communication between devices. This document describes the particular Device Profile *"Drives and Motion Control"* of the Drive in compliance with the DSP 402 standard.

The Device Profile is further divided in two sections:

- device control which specifies a finite-state machine which regulates the operations of the device;
- modes of operation, decided by the manufacturer, which are the different modes implemented and that can be used.

3.3 Object and object dictionary

CANopen uses an object-oriented approach: each device is represented by a set of objects, which everyone has a specific functionality. Depending on the particularity which it represents, the object can be composed from:

- a single variable;
- an array if the components are all of the same type;
- a record if the components are heterogeneous.

Each object is associated with an unsigned 16-bit **index** to select the object itself and with an unsigned 8-bit **sub-index** that allows you to select every single element of complex objects. The sub-index is always 0 for simple objects and starts at 1 for complex objects.

The most important part of the Device Profile is the description of the **Object Dictionary**. It is essentially the list of the objects of the devices accessible from the network in a predefined mode.

The dictionary objects are divided into categories which are associated with a specific range of index values:

- **0x0000**: not used;
- **0x0001** → **0x001F**: *Static Data Types:* these objects contain type definitions for standard data (boolean, integer, floating point, string, …). They cannot be read or written;
- **0x0020** → **0x003F**: Complex Data Types: these objects define structure composed by standard types that are common to all devices;
- **0x0040** → **0x005F**: Manufacturer Specific Complex Data Types: these objects also define structures that are composed of standard types defined by the device;
- **0x0060** → **0x0FFF**: actually reserved;
- **0x1000** → **0x1FFF**: Communication Profile Area: it contains the parameters for the communication. These objects are common to all the devices.
- **0x2000** → **0x5FFF**: Manufacturer Specific Profile Area
- **0x6000** → **0x9FFF**: Standardized Device Profile Area: it contains all the objects common to a certain class of devices that can be read and written over the network and describe their parameters and functions;
- **0xA000** → **0xFFFF**: actually reserved;

3.4 Access to the Drive

Access to the drive via CAN bus is done using mainly two types of objects, which differ in their communication characteristics:

- **Service Data Object** (**SDO**): they carry large amounts of data, have low priority on the network and are asynchronous;
- **Process Data Object** (**PDO**): they are used to improve real-time transfers of small amounts of high priority data. They are synchronous and asynchronous type. SDO must be used to map the fields and communication parameters of each PDO in the Object Dictionary.

Also important is the **Synchronization Object** (**Sync**), which is broadcasted to all nodes of the network from a Master device and which is used to implement synchronous events on the network.

4 Modes of Operation

The operational mode is selected with object **0x6060 Modes Of Operation** according to the following list:

- 1 → Profile position mode
- 3 → Velocity mode
- 6 → Homing mode
- 7 → Interpolated position mode

4.1 Profile position mode

The position mode is an operating mode where the position to which the axle must brought is sent the drive through the network. It must reach a position level following a certain speed profile. The movement can be controlled by the network. For the positioner it is possible to choose the velocity profile, which can be trapezoidal, S-shaped or sine type. This follows the Device Profile DSP 402 as far as the trapezoidal profile is concerned. The S-shaped or sine type profiles are within the manufacturer-specific ones.

Object with greatest interest:

- 0x6065 → Following Error Window
- 0x6066 → Following Error Time Out
- 0x607A → Target Position
- 0x6081 → Profile Velocity
- 0x6083 → Profile Acceleration
- 0x6084 → Profile Deceleration
- 0x6086 → Motion Profile Type
- 0x6092 → Feed Constant

4.1.1 Trapezoidal speed profile

The positioner creates a trapezoidal speed profile. Motion is characterized by limited acceleration and speed. The parameters that define the acceleration can be specified for both the acceleration phase and the deceleration phase.

Two different modes can be selected for managing the set-point:

- **single set-point**: in this case the movement in progress must be ended before the start of the next movement;
- **immediately update set-point**: instead in this case the movement starts immediately even if the previous one is still in progress.

It is also possible to vary both the maximum speed and the accelerations during positioning by modifying the values on the relevant objects.

The sequence used to sent the data is compliant with the standard and is described as following:

1. the set-point is sent using **Target Position** (*obj 0x607A*);
2. the first and fourth bit of **Control Word** (*obj 0x6040*) is set to 1. This indicates the *New Set-Point*;
3. the drive will respond by setting to 1 the twelfth bit (*Set-Point Acknowledge*) of the **Status Word** (*obj 0x6041*) to indicate that the set-point was accepted. The movement will be performed at the same time;
4. It is then necessary to bring the *New Set-Point* to zero. As response to this, as soon as the drive is able to acquire other set-points, it will reset the *Set-Point Acknowledge.*

This entails a different behaviour between the two modalities:

- in “immediately update set-point” the *Set-Point Acknowledge* will be set to zero following after sending of each set-point;
- in “single set-point” only following the start of the processing of the last set-point sent.

4.2 Velocity position mode

The axle reaches the velocity specified as target following the acceleration and deceleration imposed.

Object with greatest interest:

- 0x6065 → Velocity Window
- 0x6066 → Velocity Window Time
- 0x607A → Velocity Threshold
- 0x6081 → Velocity Threshold Time
- 0x6083 → Profile Acceleration
- 0x6084 → Profile Deceleration
- 0x6086 → Motion Profile Type
- 0x6092 → Feed Constant
- 0x60FF → Target Velocity

4.3 Homing mode

The mode searches the zero position of the axle. For search the zero position of the axle different modalities are implemented:

- **3 & 4**: homing on the positive home switch and index impulse;
- **5 & 6**: homing on the negative home switch and index impulse;
- **19 & 20 & 21 & 22**: homing without an index pulse;
- **33 & 34**: homing on the index pulse;
- **35**: homing on the current position.

It is possible to use two limit switch at the end or a home switch in the middle. Some methods, after finding the switch, search for the encoder index pulse (marker) to obtain a greater position.

The user can specifies both the velocity and the acceleration of the homing, remembering that the method uses a higher speed to search for the switch and a lower speed to search for the index pulse.

It is also possible to define a position offset: in this case, after the end of the standard procedure, the position is not reset but a values is assigned to it (- Offset).

The following convention are assumed:

- the extreme **left** of the axle determines the **lower** level, so the **left** limit switch is the **negative** one;
- the extreme **right** of the axle determines the **higher** level, so the **right** limit switch is the **positive** one.

For the choice of the method it is necessary to define:

- the homing signal used (limit switch or home switch);
- the appropriate actuation direction;
- the use or not of the index pulse.

<u>Object with greatest interest:</u>

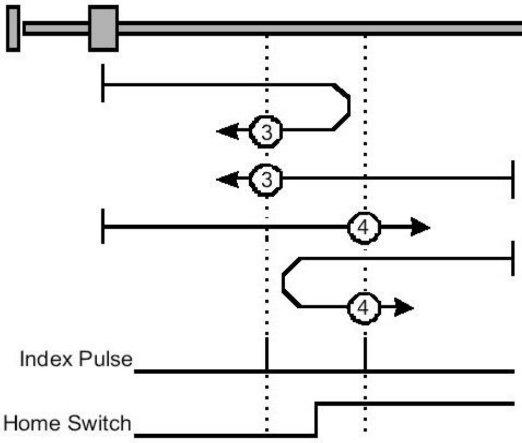
- 0x607C → Home Offset
- 0x6092 → Feed Constant
- 0x6098 → Homing Method
- 0x6099 → Homing Speed
  - Sub Index 1 → fast speed
  - Sub Index 2 → slow speed
- 0x609A → Homing Acceleration

4.3.1 Methods 3 & 4: Homing on the positive home switch

In this methods the direction of the movement depends on the state of the home switch at the start of the homing procedure. The zero position coincides with the first pulse index immediately on the right or left of the point where the home switch changes state.

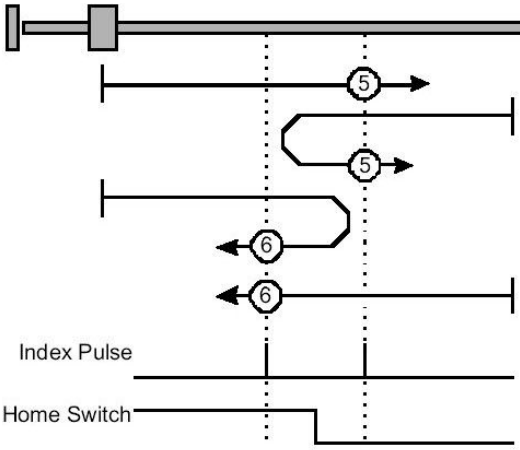
In method 3, if the switch is initially in the low state the axle moves to the right and when the switch changes state the motion is reversed and the first index pulse is searched. If the switch is initially in the high state the axle moves to the left and when the state of the switch changes the first index pulse is searched without reversing the motion.

The method 4 is the opposite of the method 3: if the switch is initially in the low state the motion is to the right and the first index pulse is search after the changes of the state without reversing the motion. If the switch is initially in the high state, the motion is to the left and then the state of the switch changes the motion is reversed and the first index pulse is searched.



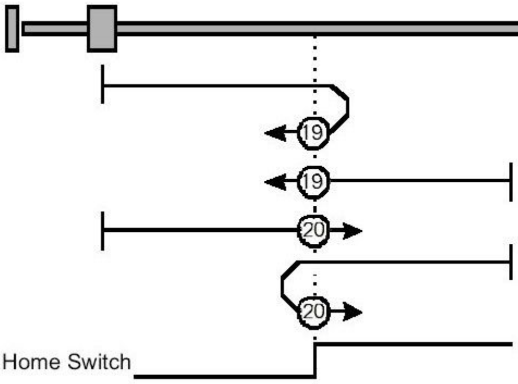
4.3.2 Methods 5 & 6: Homing on the negative home switch

These two methods are similar to the methods 3 and 4, but the operation are made in specular mode. The following image shows the differences.



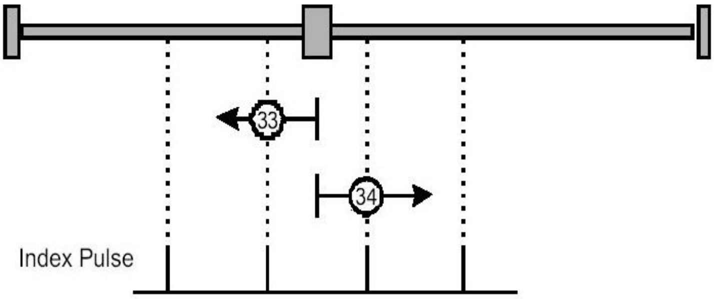
4.3.3 Methods 19 → 22: Homing without an index impulse

These methods are similar to the methods 3 → 6, except that the home position is not dependent on the index pulse. It is dependent only on the relevant home or limit switch transitions. This implies that the reset of the zero position occurs in correspondence of the change of state of the home switch.



4.3.4 Methods 33 & 34: Homing on the index impulse

The search of the zero position starts with moving the axle from the current position to the left (for method 33) and to the right (for method 34). The reset of the position takes place when the first marker is acquired.

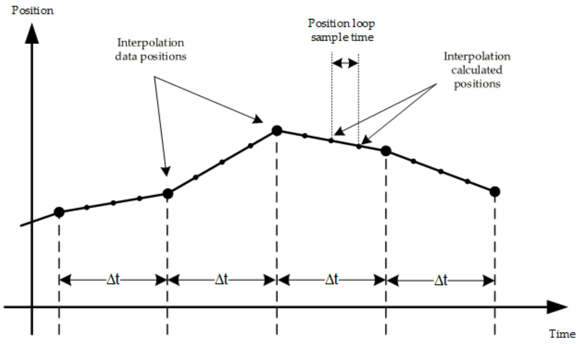


4.3.5 Method 5: Homing on the current position

The current position is taken to be home position.

4.4 Interpolated position mode

The axle follow the position set-point that is cyclically supplied to the slave using a master device. This method is used to control multiple coordinated axles or a single axle with the need for time-interpolated of set-point data. The interpolated mode uses a temporal synchronization mechanism based on the *Sync* object. The interpolation time must be specified in the *Interpolation Time Period* object. The interpolation implemented on our drive is linear.


Position
Interpolation data positions
Position loop sample time
Interpolation calculated positions
Δt
Δt
Δt
Δt
Time

From the previous image:

- **interpolated data position**: these data stands for the position set-point and are sent from the slave device on every *Sync* message;
- **position loop sample time**: these data represents the closing time of the space loop inside the drive, which in our case is 1 ms;
- **Δt**: this is the sync time sent from the master device.

The drive can divide the time between the set-points in different parts and it can updates and check the trajectory at double frequency with respect to the sending of the set-point itself. This method became indispensable if we want to manage multiple nodes in the same network. In fact, in CANopen only three devices can be updated in 1 ms, so if you want to update more than three devices you must increase the interpolation time. Subsequently, the set-points will be linearly interpolated and the set-point will be updated internally in the drive at each sampling of the space loop.

Object with greatest interest:

- 0x6092 → Feed Constant
- 0x60C1 → Interpolation Data Record
- 0x2022 → IP Position Set-Point 16 Bit
- 0x60C2 → Interpolation Time Period

5 CANopen objects

5.1 Device profile and Manufacturer specific

All the objects of the tables represented in the first pages of this document are better described in document *DSP 402.* This section contains only the manufacturer's specifications and the description of the "*manufacturer specific*" objects (0x2000 → 0x5FFF).

5.1.1 Object description of Device Profile

0x6040 *Control Word*: as DSP 402 specification. No Manufacturer Specific bit used.

0x60*41 Status Word*: as DSP 402 specification. No Manufacturer Specific bit used.

0x6060 *Modes of Operation*: already described.

0x6092 Feed constant: defines how many user defined units for one revolution of the crankshaft. After the user has fixed the units (u), also the units of velocity (u/s) and acceleration (u/s/ms) are fixed.
The sub-index 1 (*Feed*) is therefore necessary to specify the increase in the position in user units for each revolution of the crankshaft.
The sub-index 2 (*shaft revolutions*) is fixed to 1.

5.2 Notes about Guard, SDO and PDO

5.2.1 Guard services

The HeartBeat mechanism is active if the object 0x1017 (*ProducerHeartBeatTime*) is different from zero. The value of the object 0x1017 are the transmission times of the Heartbeat in ms.
In this configuration, if a Remote Request is sent to the drive with the COB-ID of the Node Guarding (*701 + NodeID*), the slave responds by sending the status but without changing the toggle bit.

The Node Guarding service is active if the object 0x100C (*GuardTime*) is different from zero. The drive responds to each Remote Request with COB-ID of the Node Guarding (*701 + NodeID*), changing the toggle-bit each time. The response is sent even if the DLC field in the request is set to zero.

Furthermore, if the object 0x100D (*LifeTimeFactor*) has been set to nonzero, the drive monitors the times with which the master sends the Remote Request and, if the maximum time (*LifeTimeFactor * GuardTime*) expires, a flag is raised to enable the sending an EMCY. When the master starts sending Remote Request again, the drive set to zero the flag and starts responding again. The EMCY is not reset.

5.2.2 SDO service

The slave accept every type of data: whoever sends the data must check that they are correct. If during a write (*SDO Download*) an error occurs, the object is not written and a response SDO is sent containing the error code that occurred (*Abort code*).

The mapping of a PDO must respect the parameters of length, mappability, existence of the object to be mapped. In the same PDO it is possible to map objects for a maximum length of 8 byte; the granularity is of 1 byte. To delete a mapping is necessary to change the *Number of Entries* of the PDO.

5.2.3 PDO service

The communication parameters that define the operation of the **TPDOs** are contained in the objects 0x1800, 0x1801, 0x1802 and 0x1803.

If *Type* is*:*

- *0* → the TPDO is synchronous and acyclic. It is sent on the first *Sync* after the occurrence of a specific event;
- *between 1 and 240* → the TPDO is synchronous and cyclic. The value (**n**) indicates how often the TPDO must be sent. So, in this case, the TPDO is synchronous and cyclic;
- *254 or 255 and EvenTimer is different from zero* → the TPDO is asynchronous and it is sent when:
    - the internal timer exceeds the time indicated in *EventTimer;*
    - an event occured on a condition specified by the user.

  In both cases, a time greater than *InhibitTime* must have passed since the last sending.

The communication parameters that define the operation of the **RPDOs** are contained in the objects 0x1400, 0x1401, 0x1402 and 0x1403.

If *Type* is:

- *between 0 and 240* → RPDO are synchronous and they are implemented at the *Sync* immediately after their reception;
- *254 or 255* → RPDO are asynchronous and they are implemented just received by the slave.

6 Drive configuration

6.1 Input, Output and Parameters

The drive parameters start from index 0x2000 to 0x2400. Each of the parameters occupies and index, so it is accessible without sub-index (sub-index at 0). The definition of the various are available on the interface *ByInterface.* Instead of Parameters, there are the Input (0x3000 to 0x3040) and Output (0x3200 to 0x3240) variables.

Unlike Input and Output variables, Parameters can be saved with an apposite command into the EEPROM of the drive if the user wants to restore the value after the turn off of the drive. Only the parameters from 0x2000 to 0x2400 can be saved in flash. To save the parameters, set to 1 the eight bit of IN_CMD_0 (0x2000).

6.2 CAN configuration

To configure optimally the drive to use CANopen you must configure three parameters using *ByInterface*:

1. **Node ID**: P0001, address 0x001. From 1 to 246
2. **OPMODE**: P0010, address 0x001A. Set to 8
3. **Bit-rate**: P0018, address 0x0012. From 0 to 7:
   - 0 → 1 Mbit
   - 1 → 800 Kbit (not supported)
   - 2 → 500 Kbit
   - 3 → 250 Kbit
   - 4 → 125 Kbit
   - 5 → 50 Kbit
   - 6 → 20 Kbit
   - 7 → 8 Kbit

After setting Node ID and Bit-rate the values must be saved with the procedure described previously and the drive must be restarted because these parameters are read only when the drive is turned on.

7 Alarm codes

When an alarm occurs it is reported by the third bit of the *Status Word*. The alarms of the drives can be read, after an emergency, on the registers:

- Output[4] mapped on address 0x3204;
- Output[5] mapped on address 0x3205.

This registers can also be read using a SDO. For reset the alarms set to 1 the seventh bit of the *Control Word.*

Code	Coding	Description
		0x3204 – Output Variable #4
F00	Flash failure	Anomaly in the EEPROM of the parameters
F01	Loaded default parameters	Error while reading user parameters
F02	Loaded protected parameters	Error while reading calibration parameters
F03	F03	*Reserved*
F04	I^2t Drive	Maximum value of I^2t reached
F05	Over voltage	Maximum DC-Link voltage exceeded
F06	Under voltage	Minimum DC-Link voltage reached or drive enable while DC-Link not yet active
F07	Power fail	Over current or fault on power stage of the converter
F08	Heatsink temperature	Maximum temperature of heatsink reached
F09	Motor temperature	Maximum motor temperature reached
F10	Regenerative resistance overload	Breaking power reached
F11	Internal temperature	Maximum MCU temperature reached
F12	Over current regenerative circuit	Over current on the breaking circuit reached
F13	Over voltage 24V	Reached the upper limit of the auxiliary voltage (24V)
F14	Under voltage 24V	Reached the lower limit of the auxiliary voltage (24V)
F15	F15	--
		0x3205 - Output Variable #5
F16	Brake	--
F17	Fan	--
F18	Phase Loss	--
F19	Feedback	Feedback anomaly. Check the wiring or safety devices
F20	Feedback Initialization	Error while feedback initialization. Check the wiring
F21	Current Over Range	--
F22	Sin Cos Over Range	Error while calculating Sin Cos offset
F23	Over Speed	--
F24	Generic Fault	--
F25	F25	--
F26	Fieldbus	Alarm during the configuration of the CAN interface
F27	F27	--
F28	F28	--
F29	External Fault	--
F30	STO	--
F31	F31	--

8 Examples

8.1 Some examples of data

Example of the *structure* of a data:

- 23 → Type or request (SDO Download Initiate Request)
- 00 18 → Index (0x1800)
- 01 → Sub-index (0x01)
- 81 01 00 C0 → Data (0xC0001801)

Data	Comment
23 00 18 01 81 01 00 C0	Set COBID of TPDO1 to extended 0x181; then disable
2F 01 18 02 FE 00 00 00	Set *Transmission Type* of TPDO2 to 0xFE (*Event driven*)
2F 60 60 00 03 00 00 00	Set Modes of Operation to Velocity mode
23 83 60 00 FF FF FF FF	Set profile of acceleration
2B 01 18 03 00 00 00 00	Set *Inhibit Time* of TPD02 to 0x0000
2B 00 14 05 C8 00 00 00	Set *Event Timer* of RPDO1 to 0x00C8
23 01 1A 01 20 00 1C 32	Map object with index 0x321C, sub-index 0x00 and dimension 0x20 (32bit) in TPDO2
2F 01 1A 00 02 00 00 00	Set Number Of Objects = 2 to TPDO2
23 00 16 01 10 00 40 60	Map object with index 0x6040, sub-index 0x00 and dimension 0x20 (16 bit) in RPDO1. This is the *Control Word*
2B 20 23 00 34 12 00 00	Request to write to Parameter 800 the value 0x1234
40 18 10 04 00 00 00 00	Request to read of Serial Number in Identity Record

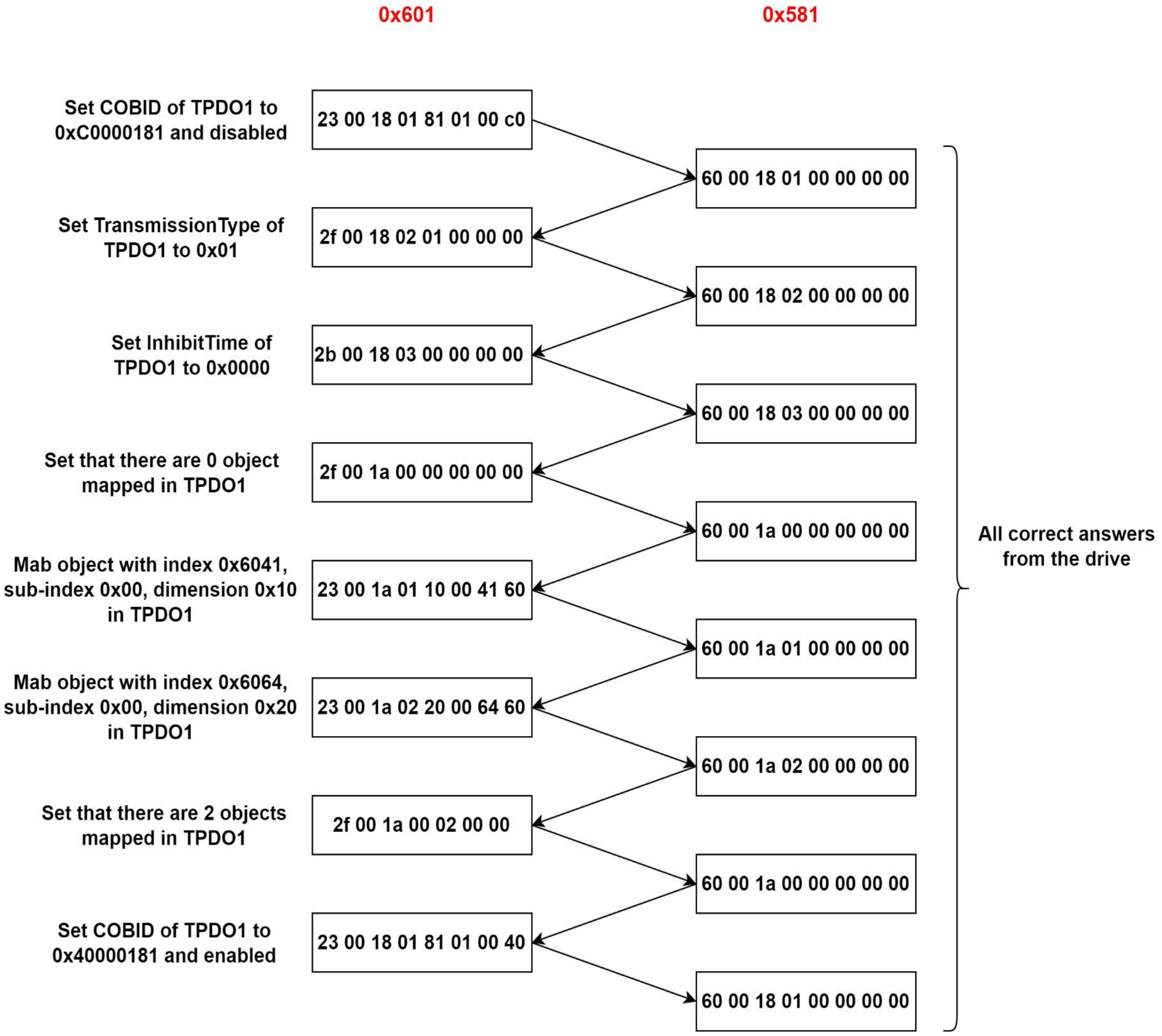
Example of *response*:

- 60/80 → 60 = correct write, 80 = error on write, ...
- 00 18 → Index (0x1800)
- 01 → Sub-index (0x01)
- 00 00 00 00 → In case of 0x80 return error code as CANopen specification

8.2 TPDO Mapping example

This example shows how to map a TPDO:

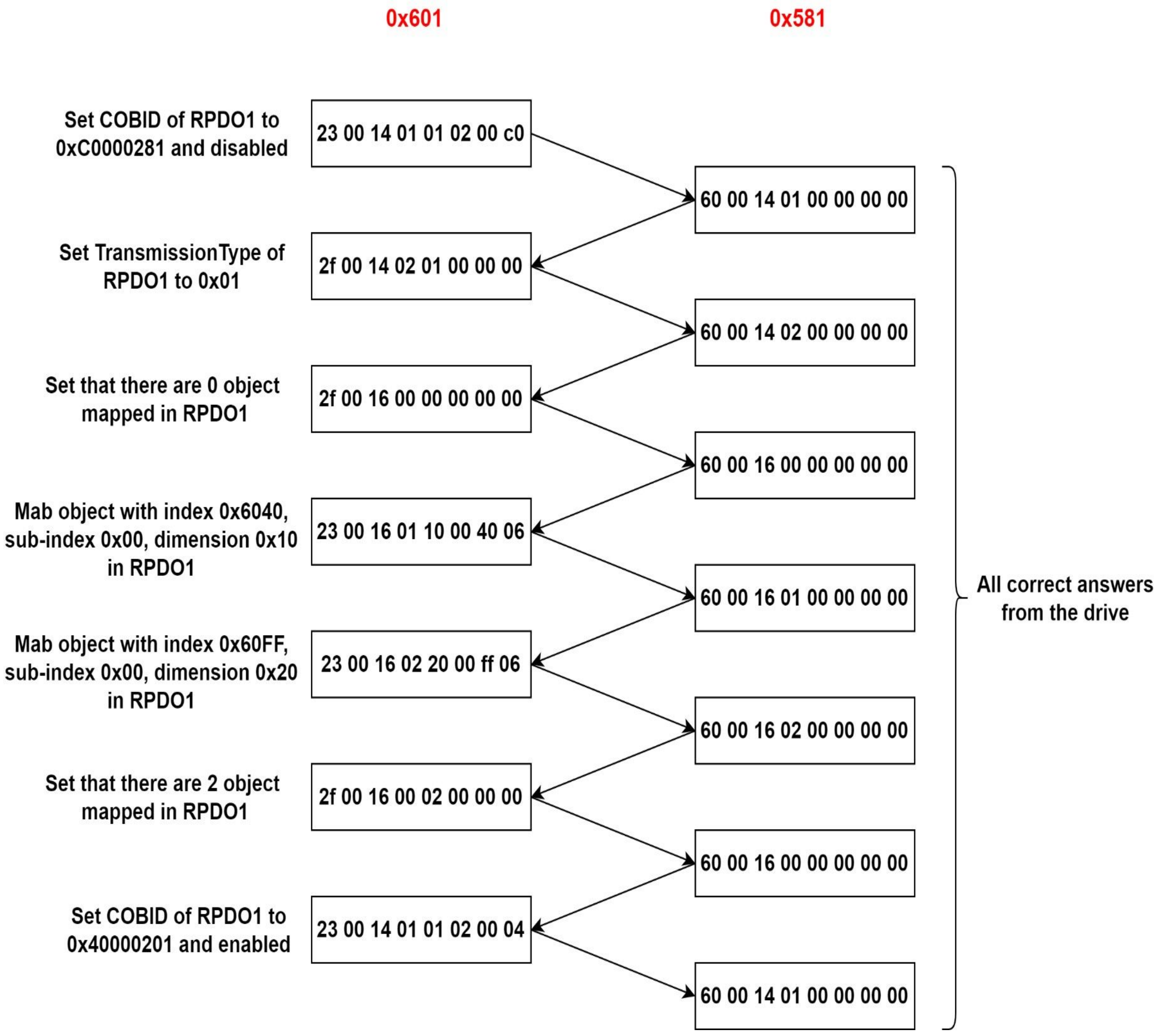
- The *COBID* of the TPDO1 is mapped but TPDO1 stays *disabled;*
- *Transmission Type* = 0x01 indicates the sync;
- *Inhibit Time* is set to 0;
- Set that there are zero objects mapped on TPDO1;
- Two objects with different address (*0x6041* and *0x6064)* and size (*0x10* and *0x20*) are mapped in TPDO1;
- Set that there are *two objects* mapped on TPDO1;
- At the end the TPDO1 is *enabled;*


0x601
0x581
Set COBID of TPDO1 to 0xC0000181 and disabled
23 00 18 01 81 01 00 c0
60 00 18 01 00 00 00 00
Set TransmissionType of TPDO1 to 0x01
2f 00 18 02 01 00 00 00
60 00 18 02 00 00 00 00
Set InhibitTime of TPDO1 to 0x0000
2b 00 18 03 00 00 00 00
60 00 18 03 00 00 00 00
Set that there are 0 object mapped in TPDO1
2f 00 1a 00 00 00 00 00
60 00 1a 00 00 00 00 00
Mab object with index 0x6041, sub-index 0x00, dimension 0x10 in TPDO1
23 00 1a 01 10 00 41 60
60 00 1a 01 00 00 00 00
Mab object with index 0x6064, sub-index 0x00, dimension 0x20 in TPDO1
23 00 1a 02 20 00 64 60
60 00 1a 02 00 00 00 00
Set that there are 2 objects mapped in TPDO1
2f 00 1a 00 02 00 00
60 00 1a 00 00 00 00 00
Set COBID of TPDO1 to 0x40000181 and enabled
23 00 18 01 81 01 00 40
60 00 18 01 00 00 00 00
All correct answers from the drive

8.3 RPDO Mapping example

This example shows how to map a RPDO:

- The *COBID* of the RPDO1 is mapped but RPDO1 stays *disabled;*
- *Transmission Type* = 0x01 indicates the sync;
- Set that there are zero objects mapped on RPDO1;
- Two objects with different address (*0x6040* and *0x60FF)* and size (*0x10* and *0x20*) are mapped in RPDO1;
- Now there are *two objects* mapped on RPDO1;
- At the end the RPDO1 is *enabled;*


0x601
0x581
Set COBID of RPDO1 to 0xC0000281 and disabled
Set TransmissionType of RPDO1 to 0x01
Set that there are 0 object mapped in RPDO1
Mab object with index 0x6040, sub-index 0x00, dimension 0x10 in RPDO1
Mab object with index 0x60FF, sub-index 0x00, dimension 0x20 in RPDO1
Set that there are 2 object mapped in RPDO1
Set COBID of RPDO1 to 0x40000201 and enabled
All correct answers from the drive

```
23 00 14 01 01 02 00 c0
60 00 14 01 00 00 00 00
2f 00 14 02 01 00 00 00
60 00 14 02 00 00 00 00
2f 00 16 00 00 00 00 00
60 00 16 00 00 00 00 00
23 00 16 01 10 00 40 06
60 00 16 01 00 00 00 00
23 00 16 02 20 00 ff 06
60 00 16 02 00 00 00 00
2f 00 16 00 02 00 00 00
60 00 16 00 00 00 00 00
23 00 14 01 01 02 00 04
60 00 14 01 00 00 00 00
```