Inspiring Motion
Since 1988

Gold DS-301 Implementation Guide

January 2017 (Ver. 1.016) **www.elmomc.com**

Elmo
Motion Control

Notice

Revision History

Version	Date	Changes
Ver. 1.000	Nov 2012	Initial Release
Ver. 1.001	Aug 2013	Vladyslav changes: 0x6040 in table: RxMap,TxMap 6072, 6073 UINT Object 0x44 (PV data type) changed Obj 0x80 “Interpolation” instead of “Interpolated” Tab of EMCY error codes was copied from ECAT manual
Ver. 1.002	29th Sep 2013	Vladyslav: Object 0x2046 added in table and description Object 0x2087 added in table and description Object 0x2060 added in table and description

Version	Date	Changes
		Object 0x2061 added in table and description Object 0x2062 added in table and description
Ver. 1.003	1st Oct 2013	Descriptions of ECAT only objects are added. See chapter 17 All data types as SIGNED8, SIGNED16, SIGNED24, SIGNED32 replaced with INTEGER8, INTEGER16, INTEGER24, INTEGER32 accordingly Table in chapter 2.8 Object dictionary –data types changed 0x10F1 name changed Chap 2.8, table changed
Ver. 1.004	October 2013	Updated Template, and corrections 0x2020 sub indexes 2 and 5 in ms, max = 1000000000ms
Ver. 1.005	March 2014	Representation of Error Codes in Tab 4.1 was changed. In accordance with this changed : 0x1003, 0x1006,0x1400, 0x1800 Added new error to tab 4.1 Added Description of 1003 Edited Objects: 0x2F20, 0x2F21. Object 0x2020.1 restricted to 50000 Corrections to the Object dictionary
Ver. 1.006	April 2014	Table 3.2 (Object Dict)0x2005 type INT32 (was UINT32) (Object Dict)Obj 2090 UINT32 (was STRING) (Object Dict) 20B0 sub 0 RO (was RW) (Object Dict) 2206 UINT16 (was INT16) Object Dict) 2E00 write RxMap instead of ECAT…CAN… 0x60B0,0x60B1,0x60B2,0x60C1,0x60C2,0x60FF CAN : Rx Map , no TxMap 0x1804 deleted Object 0x2030, example fixed Initiate SDO Upload Protocol fixed, chapter 4.3 Change in Table 16.7 Motor Faults and EMCY according to Command Reference
Ver. 1.007	Mar 2015	Chapter 6.2 , example corrected New tables 6.1, 16.6, 16.7, 16.8
Ver 1.008	Apr 2015	Changes in Tables 16.7, 16.8 Changes in objects 0x2020,0x20B0,0x2F45,0x2081

Version	Date	Changes
		Object dictionary table changed Changes to section 16.34. Object 0x2E10: Home on Touch Probe
Ver 1.009	Apr 2015	Object table changed Changed objects; 0x1000, 0x1018, 0x1600-0x1603, 0x1A00-0x1A03, 0x2060, 0x2202, 0x2F70, 0x1017, 0x2085 Object 0x2E00 is added
Ver 1.010	Aug 2015	Changes to: Object 0x2201: Digital input low byte Object 0x20FD: Digital Inputs Changes to Chapter 6.2 Elmo Error Codes
Ver 1.011	Sep 2015	Changes to Table 6.1 Elmo Emergency Codes Changes to 0x2F70: CAN Encoder Range
Ver 1.012	Sep 2015	Change in Table 6-1 Emergency codes and table 16-7 Motor Fault and EMCY Changed Table 7-3 Supported NMT Services Table 3-2 Objects updated Addition of Chapter 4: Addressing Elmo Parameters via CANopen Objects together with Section 17.46. Objects 0x3000 to 0x32A3: Elmo parameters objects Object description updated: 0x2202, 0x22A1, 0x22A3, 0x2020
Ver 1.013	Dec 2015	Objects changed: 0x20FD, 0x2202, 0x1800, 0x1801, 0x1802, 0x1803, 1400 Objects 0x60E0,0x60E1 removed in object table (tab 3-2) Changes to object table (Tab 3-2) Changes to chapter 2.7
Ver 1.014	June 2016	Correction to Object 2205/2
Ver 1.015	Oct 2016	Correction to section 17.46 Object 0x3000 Correction to section 7.2 Correction to section 4.1. Elmo Parameters via SDO Interface (CoE \ CANopen) example Corrections to sections 17.25. Object 0x2203: Application Object and 17.42. Object 0x2F41: DS402 Configuration object

Version	Date	Changes
Ver 1.016	Jan 2017	Section 7.1 Emergency codes changed Object 0x1016 description changed, chapter 16.12

Chapter 1: Introduction

This manual explains how to implement EtherCAT and CANopen DS-301 communication with Elmo's Gold DSP-based digital servo drives. It provides a description of the Gold drives and the means of implementing communication based on the EtherCAT and CiA CANopen protocols.

Most of the Gold functionality is standard, according to CiA documents DS-301, version 4.01, DSP 402 (proprietary) and the CiA OS interpreter. In this document, emphasis is placed on manufacturer-specific behaviors, although it also repeats certain CiA standard material, to enhance understanding and to complete certain descriptions. The manual contains data which is relevant to the operation of the Elmo drive.

1.1. Relevant Elmo Documentation

This manual is part of the Elmo Gold documentation set, which also includes:

- Generic Gold hardware manuals providing full instructions for for all Panel Based and BLM Gold digital servo drives and individual Installation Guides.
- The EASII User Manual, which includes explanations of all the software tools that are a part of Elmo's Application Software environment.
- The Gold Command Reference Manual, which describes, in detail, each software command used to manipulate the motion controller.
  This is the main source of detailed explanations of all Gold commands mentioned in this manual.
- The Gold Language & User Program Manual, which describes the comprehensive software used with Gold digital servo drives

1.2. Terms and Abbreviations

The following **terms** and **abbreviations** are used in this manual:

Term / Abbreviation	Definition
CAL	CAN application layer.
CAN client or master	A host — typically a PC — or other control equipment that supervises the nodes of a network.
CAN server or CAN slave	A node in the CAN network that can give service under control of the CAN master.
CMS	CAN message specification.
COB	Communication object; a CAN message.
COB-ID	A binary bit-field that includes the ID of the server with which the master talks, and the type of COB.
DSP	Digital Signal Processor
EDS	Electronic data sheet; a standard form of all CAN objects supported by a device. The EDS is used by external CAN configurators.
ID	Identifier; the name by which a CAN device is addressed.
LSB	Least Significant Bit (or Byte)
LSS	Layer setting service: methods for configuring the ID and baud rate of a slave, using the standard DSP 305
MSB	Most Significant Bit (or Byte)
NMT	Network Management
Object	A CAN message with a meaningful functionality and/or data. Objects are referenced according to addresses in the object dictionary. DO: Data object.
OD	Object dictionary, which is the full set of objects supported by the node. It is the interface between the application and communication (see “Object” below)
PLC	Programmable controller. A PLC can serve as a CAN master for SimplIQ digital servo drives.
Receive	In this manual, “received” data is sent from the control equipment to the servo drive.

Term / Abbreviation	Definition
Transmit	In this manual, "transmitted" data is sent from the servo drive to the other equipment.
NONE	**Wherever NONE is defined as the Default Value for an Object or parameter, no default value is relevant.**

The following table (Table 1-1) lists the shortened terms used in this manual:

Prefix/Suffix	Definition
UU	User defined Units
Cnt/sec	counts per second
Sub	Sub Index
TxMap	Mappable to TPDO
RxMap	Mappable to RPDO

Table 1-1 Shortened Terms

1.3. Gold Line Communications

Gold Line digital servo drives can simultaneously communicate using both CAN and USB communication lines, which are always open for communication. The communication parameters are set using the PP command.

The following table compares the main features of both communication modes, as implemented with Elmo Gold Line digital servo drives:

Features	CANopen
Baud rate	50,000 - 1,000,000
Interpreter method	Binary or ASCII
Fast referencing	Yes, for PVT, PT and ECAM (To be edited and applied)
Network of servo drives	Yes
Multiple servo drive synchronization	Yes
Standardization	Compliant with CiA standard
Special equipment required	CAN communication interface (available as an add-on ISA or PCMCIA card for PCs) with appropriate software
Ease of use	Basic capabilities included in Elmo EAS program and the GMAS master.

CANopen communication achieves higher rates and is able to support the following advanced functions:

- High speed online reference generation, required for supporting complex motions
- Binary interpretation, which gives a simple method of communicating the drive commands and parameter via Elmo interpreter (refer to Gold Language & User Program Manual)
- Servo network applications

To benefit from CAN communication and the CiA DS-301 CANopen standard, the user must have a good understanding of the basic programming and timing issues of a CANopen network.

Chapter 2: CANopen Basics

This chapter describes the general CANopen communication features most relevant to Elmo Gold servo drives.

2.1. Physical Layer

CAN is a serial communication standard in which the transferred data is coded as electrical pulses on a two-wire communication line. The device that handles the CAN physical layer is called the CAN controller. The device that transmits data over the CAN lines is called the CAN transceiver. Gold digital servo drives use a CAN controller built into the drive DSP.

2.2. Standard vs. Extended Addressing

Each CAN message frame includes an arbitration field that defines the type of data sent and its address. CAN version 2.0A supports 11 arbitration bits for this purpose; the seven least significant define the address and the four most significant define the type of message sent. 16 message types are supported. CAN version 2.0B supports 29 arbitration bits, of which the seven least significant define the address and 21 bits define the message type. In CiA DS-301, the arbitration bits indicate the object and the node-ID, together comprising the COB-ID. The Elmo drive supports CAN 2.0A mode.

CAN communication is prioritized so that messages with higher priority are transmitted first. The arbitration field determines the message priority: The lower the number in the arbitration field, the higher the message priority. ID 0 gives the highest priority. The Gold drives support the CAN version 2.0A (11-bit) addressing method only, meaning that it ignores messages of 29 bits. A setup parameter (**PP[13]** - Node ID) selects the CAN object identification to be used.

2.3. Client - Server Relations

A CAN master (or client) is a controller that makes requests to nodes to respond to its commands. A CAN slave (or server) responds to the commands issued by the CAN master. The CAN protocol permits both single-master and multiple-master networks.

The Gold servo drives assume a single-master network arrangement, in which the servo drives are the slaves and the machine controller or PLC is the master. Every servo drive has a unique ID in the range [1…127]. The network master does not require an ID. As a slave, the servo drive never sends an unrequested message, other than emergencies. The drive responds only to messages addressed to its ID or to broadcast messages, which have an ID of 0. All messages sent by a servo drive are marked with its own ID.

⚠ **If two servo drives have been assigned the same ID, the CAN network may crash.**

2.4. RTR – Remote Transmission Request

The RTR is not supported by the Elmo drive.

2.5. Object Dictionary

An object dictionary (OD) is a naming structure that gives a unique identifier to each data item — or "object" —that is communicated over the CAN bus. An object is identified by an index and, if it is a complex object, also by a sub-index. A CANopen client can manipulate an object of a CANopen server by referring to its identifier, according to the access permission of the object. An object's access permission may be read-only, write-only, or read-write.

CiA DS-301 requires a set of mandatory objects for all CANopen devices. Other OD items are predefined by CiA DS-301 to have fixed identifiers, if supported. The OD also accommodates manufacturer-specific objects.

2.6. Communication Objects

The data-byte units transported through a CAN network are called communication objects (COBs). Gold servo drive uses the following COB types:

COB Type	Description
Service data object (SDO)	SDO messages provide a direct access to the Object Dictionary The server receives the SDO, which specifies in its message which object is to be dealt with. SDOs are confirmed messages requires a response from the server before the next message can be transmitted. SDOs are used to transfer sequence of segments of data set from the client to the server and vice versa. Depending on the size of data, SDO can transfer block of data of up to 127 segments in a single transfer. In case of small data size, up to 4 bytes, the SDO transfer (expedite SDO) can be performed in a single transfer.
Process data object (PDO)	PDO are short unconfirmed messages , used for real time data transfer m/o protocol overhead. The data type loaded on a PDO must be defined during a configuration process namely PDO mapping. The PDO mapping typically performed in PreOP state. After the mapping procedure, the PDO corresponds to a an object from the Object Dictionary. After the configuration, the payload of the PDO is defined and known to both the server and the client allowing a direct access to the object without going thru the OD. PDO transmission from the drive is called TPDO and if received by the drive it is called RPDO. PDOs are transmitted and received by the drive according to a predefined event. PDO setting (i.e. mapping, transmission type) is defined by SDO.
Emergency (EMCY)	Emergency messages are used by the servo drives to warn of

COB Type	Description
	an exception. The EMCY is the only COB type that a servo drive transmits without first being explicitly asked. EMCY objects are similar to servo drive “interrupts”: they eliminate the need to poll the servo drive continuously for the emergency status.
Network Management (NMT)	NMT objects follows the Master-Slave structure. The NMT master can control the CANopen state machine, initialize, monitor, reset and stop the communication of any CAN device in the network.
Synchronous signal (SYNC)	SYNC signals provide a basic network synchronization mechanism. The SYNC master produce a periodic SYNC signal allowing the devices to operate simultaneously. The SYNC has a high priority on the network arbitration which reduces the delay of the signal to minimum.
Layer Setting Service (LSS)	This service is used to assign IDs and baud rates to newly installed products.

The type of COB transmitted is indicated in the arbitration field of the message, and thereby determines its priority. The relation between bits 8 to 11 of the arbitration field (COB-ID) and the COB type is presented in the following table:

COB Type	Bits 7 - 10 of COB-ID (function code)	COB ID Range (function code + Node ID)
NMT	0000	0
SYNC	0001	128 (80h)
Time Stamp	0010	256 (100h)
Emergency	0001	129…255 (81h…ffh)
PDO1 - Transmit	0011	385…511 (181h…1ffh)
PDO1 - Receive	0100	513…639 (201h…27fh)
PDO2 - Transmit	0101	641…767 (281h…2ffh)
PDO2 - Receive	0110	769…895 (301h…37fh)
PDO3 - Transmit	0111	897…1023 (381h…3ffh)
PDO3 - Receive	1000	1025…1151 (401h…47fh)
PDO4 - Transmit	1001	1153…1279 (481h…4ffh)
PDO4 - Receive	1010	1281…1407 (501h…57fh)
SDO - Transmit	1011	1409…1535 (581h…5ffh)
SDO - Receive	1100	1537…1663 (601h…67fh)
Error control (node guarding)	1110	1793…1919 (701h…77fh)

Example:

The COB-ID of PDO1, when received by node number 2, is binary 01000000010, which is decimal 514, or 202 hexadecimal. The IDs of the servo drives are set in the range 1…127.

2.7. Object Dictionary - Data Types

The Elmo CAN controller supports the following data types:

Index (hex)	Object	Name
0002	DEFTYPE	INTEGER8 (INT8)
0003	DEFTYPE	INTEGER16 (INT16)
0004	DEFTYPE	INTEGER32 (INT32)
0005	DEFTYPE	UNSIGNED8 (UINT8)
0006	DEFTYPE	UNSIGNED16 (UINT16)
0007	DEFTYPE	UNSIGNED32 (UINT32)
0008	DEFTYPE	REAL32
0009	DEFTYPE	VISIBLE_STRING
0010	DEFTYPE	INTEGER24
0016	DEFTYPE	UNSIGNED24
0020	DEFSTRUCT	PDO_COMMUNICATION_PARAMETER
0021	DEFSTRUCT	PDO_MAPPING
0022	DEFSTRUCT	SDO_PARAMETER
0023	DEFSTRUCT	IDENTITY
0040	DEFTYPE	PVT_DATA_PARAMETERS
0041	DEFTYPE	PT_DATA_PARAMETERS
0042	DEFTYPE	BINARY_INTERPRETER_QUERY
0043	DEFTYPE	BINARY_INTERPRETER_COMMAND
0044	DEFTYPE	DS402_PV_DATA
0045	DEFTYPE	HOME_ON_BLOCK_LIMITS
0080	DEFTYPE	DS402_INTERPOLATED_TIME_PERIOD
0081	DEFTYPE	DS402_ INTERPOLATED_DATA_CONFIGURATION
0082	DEFTYPE	DS402_INTERPOLATION_DATA_RECORD

2.7.1. PDO Communication Parameter - Object 0x20

Index	Sub-index	Field in PDO Communication Parameter Record	Data Type
0x0020	0h	Number of supported entries in record	UNSIGNED8
	1h	COB-ID	UNSIGNED32
	2h	Transmission type	UNSIGNED8
	3h	Inhibit time	UNSIGNED16
	4h	Reserved	UNSIGNED8
	5h	Event timer	UNSIGNED16

2.7.2. PDO Mapping - Object 0x21

Index	Sub-index	Field in PDO Parameter Mapping Record	Data Type
0021h	0h	Number of mapped objects in PDO	UNSIGNED8
	1h	First object to be mapped	UNSIGNED32
	2h	Second object to be mapped	UNSIGNED32
…	…	…	…
	8h	8th object to be mapped	UNSIGNED32

2.8. Servo Drive Device-Specific Data Types

he following tables describes manufacturer specific object that are used by Elmo drive:

2.8.1. PVT Data Parameters Object 0x40

MSB		LSB
Time (UNSIGNED8)	Velocity (INTEGER24)	Position (INTEGER32)

2.8.2. PT Data Parameters - Object 0x41

MSB	LSB
Position 2 (INTEGER32)	Position 1 (INTEGER32)

2.8.3. Binary Interpreter Query - Object 0x42

MSB							LSB
7	6	5	4	3	2	1	0
				Attribute high	Attribute low	Letter low	Letter high

For more information about the binary interpreter query, refer to Chapter 13:Binary Interpreter Commands.

2.8.4. Binary Interpreter Command - Object 0x43

MSB							**LSB**
7	6	5	4	3	2	1	0
Data high	Data	Data	Data low	Attribute high	Attribute low	Letter low	Letter high

For more information about the binary interpreter query, refer to Chapter 13:Binary Interpreter Commands.

2.8.5. DSP 402 PV Data - Object 0x44

Index	Sub-index	Field in Interpolation Time Period Record	Data Type
0044h	0h	Number of entries	UNSIGNED8
	1h	Target Position	INTEGER32
	2h	Target Velocity	INTEGER32

2.8.6. Home On Block Limits - Object 0x45

Index	Sub-index	Field in Interpolation Time Period Record	Data Type
0x0045	0h	Number of entries	UNSIGNED8
	1h	Torque limit	UNSIGNED16
	2h	Time Limit	UNSIGNED32
	3h	Distance Limit	UNSIGNED32
	4h	Detection Velocity Limit	UNSIGNED32
	5h	Detection Velocity Time Limit	UNSIGNED32

2.8.7. DS402 Interpolated Time Period , Object 0x80

Index	Sub-index	Field in Interpolation Time Period Record	Data Type
0x0080	0h	Number of entries	UNSIGNED8
	1h	Interpolation time units	UNSIGNED8
	2h	Interpolation time index	INTEGER8

2.8.8. DSP 402 Interpolated Data Configuration - Object 0x81

Index	Sub-index	Field in Interpolation Time Period Record	Data Type
0x0081	0h	Number of entries	UNSIGNED8
	1h	Maximum buffer size	UNSIGNED32
	2h	Actual buffer size	UNSIGNED32
	3h	Buffer organization	UNSIGNED8
	4h	Buffer position	UNSIGNED16
	5h	Size of data record	UNSIGNED8
	6h	Buffer clear	UNSIGNED8

*Refer to the restrictions in the MAN-G-DS402 manual

2.8.9. DSP 402 Interpolated Data Record - Object 0x82

Index	Sub-index	Field in Interpolation Time Period Record	Data Type
0x0082	0h	Number of entries	UNSIGNED8
	1h	Target Position	INTEGER32
	2h	Target Velocity	INTEGER32

2.9. Representation of Numbers

CAN communication delivers numerical data stored in binary form. Integers are stored by their binary representation, while floating-point numbers are stored according to the IEEE representation. Gold digital servo drives support three types of data: short integers (two bytes), long integers (four bytes) and floating-point numbers (four bytes). These multiple-byte numbers are stored in the CAN messages according to CAN standards, using the “little endian” (Intel-type) convention. With this method, the number is inverted before storage: The most significant byte of the number receives the lowest address and the least significant byte receives the highest address. More information about the little endian method is provided in Chapter 19: Little and Big Endians.

Example:
The following is an 8-byte CAN message: **22 16 10 51 10 27 20 00**

Bytes 0 - 1	0x1622
Bytes 2 - 3	0x5110
Bytes 4 - 7	0x00202710

The CAN message data field will be as follows:

Byte	Contents
0	0x22
1	0x16
2	0x10
3	0x51
4	0x10
5	0x27
6	0x20
7	0x00

Chapter 3: The Object Dictionary

The object dictionary is essentially a grouping of objects that are accessible via receive and transmit SDOs. Part of the object can be mapped to transmit and receive PDOs (TPDO and RPDO, respectively) in a predefined manner.

The following layout (Table 3-1) is used with the objects in the object dictionary:

Index (Hex)	Object
0	Not used
0001 - 001F	Static data type
0020 - 003F	Complex data type
0040 - 005F	Manufacturer-specific data type
0060 - 0FFF	Reserved
1000 - 1FFF	Communication profile area
2000 - 2FFF	Manufacturer-specific profile area
6000 - 6FFF	Standardized device profile area
A000 - FFF	Reserved

Table 3-1 Object Dictionary Layout

The following table (Table 3-2) lists the objects supported by Gold digital servo drives. Each object is addressed by a 16-bit index. Some of the objects may include 8-bit sub-indices, which are described in the object description. The object **Name** is that given by CiA or Elmo according to object type. An **Attribute** can be RO (read only), WO (write only) or RW (read and write). The objects 0x0001 – 0x2FFF are described in remaining chapters of this manual. Objects 0x6000 – 0x6FFF are described in MAN-G-DS402 manual.

Refer to the Table 1-1 for a definition of the shortened terms used in this manual.

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
1000/0	Device type	UINT32	RO	No	CAN, ECAT. Return 0x192
1001/0	Error Register	UINT8	RO	No	CAN, ECAT.
1002/0	Manufacturer Status Register	UINT32	RO	TxMap	CAN, ECAT. Similar to **SR** command
1003/16	Pre-Defined Error Field	UINT32	RO	No	CAN, ECAT. Up to 16 last transmitted EMCY messages
1006/0	Communication Cycle Period	UINT32	RW	No	CAN only. Present for compatibility reasons.

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
1008/0	Manufacture Device Name	STRING	CONSTANT	No	CAN, ECAT. Drive given name
1009/0	Manufacture Hardware Version	STRING	CONSTANT	No	CAN, ECAT.HW identification number
100A/0	Manufacture Software Version	STRING	CONSTANT	No	CAN, ECAT. Similar to **VR** command
100B/0	CANopen Node ID	UINT8	RO	No	CAN only. Similar to **PP[13]** command
1010/1	Store Parameters	UINT32	Sub 0: RO, Sub 1: RW	No	CAN, ECAT. Similar to **SV** command
1011/1	Restore Default Parameters	UINT32	Sub 0: RO, Sub 1: RW	No	CAN, ECAT. Similar to **LD** command
1016/2	Consumer heartbeat time	UINT32	Sub 0 –RO, Sub 1,2: RW	No	CAN only
1017/0	Producer heartbeat time	UINT16	RW	No	CAN only
1018/4	Identity Object	UINT32	RO	No	CAN, ECAT. Used in LSS for drive identification
1023/3	**OS** command	RECORD	RW	No	CAN only. See OS Interpreter chapter in DS301 Manual
1024/0	**OS** Command mode	UINT8	WO	No	CAN only.
1029/1	Error Behavior object	UINT8	Sub 0: RO, Sub 1: RW	No	CAN only. Loss of heartbeat communication response.
10E0/2	Device ID Reload	INT16	Sub 0: RO, Sub 1,2: RW	No	ECAT only
10F1/2	SYNC error setting	UINt32	Sub 0,1: RO; Sub 2: RW	No	ECAT only
1400/2 - 1403/2	RPDO communication parameter	Data type 0x20	CAN: RW ECAT: RO	No	CAN only. Receive PDO mapping communication parameters of PDO1 to PDO4.
1600	RPDO mapping parameter	UINT32	CAN: RW ECAT: RO	No	CAN, ECAT. Receive PDO mapping parameters; CAN: Up to 8 entries. ECAT: Up to 3 entries.
1601, 1602	RPDO mapping parameters	UINT32	CAN: RW ECAT: RO	No	CAN, ECAT. Receive PDO mapping parameters; CAN: Up to 8 entries. ECAT: Up to 2 entries.

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
1603	RPDO mapping parameters	UINT32	CAN: RW ECAT: RO	No	CAN, ECAT. Receive PDO mapping parameters; CAN: Up to 8 entries. ECAT: Up to 4 entries.
1604/4	RPDO mapping parameters	UINT32	RO	No	ECAT only. Receive PDO mapping parameters;
1605/7	RPDO mapping parameters	UINT32	RO	No	ECAT only. Receive PDO mapping parameters;
1606/6	RPDO mapping parameters	UINT32	RO	No	ECAT only. Receive PDO mapping parameters;
1607/8, 1608/8	RPDO mapping parameters	UINT32	RW	No	ECAT only. Receive PDO mapping parameters;
160A/1	RPDO mapping parameters	UINT32	RO	No	ECAT only. Receive PDO mapping parameters;
160B/2	RPDO mapping parameters	UINT32	RO	No	ECAT only. Receive PDO mapping parameters;

160C/1-160F/1;
1611/1-1619/1;
161C/1, 161D/1
	RPDO mapping parameters	UINT32	RO	No	ECAT only. Receive PDO mapping parameters;

161A/1	RPDO_161A Mapping	UINT32	RO	No	CAN,ECAT. Receive PDO mapping parameters;
161E/2	RPDO mapping parameters	UINT32	RO	No	ECAT only. Receive PDO mapping parameters;
161F/1 - 1621/1	RPDO mapping parameters	UINT32	RO	No	ECAT only. Receive PDO mapping parameters;

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
1800/5 – 1803/5	TPDO communication parameter	UINT32	CAN: RW ECAT: RO	No	CAN only. Transmit PDO mapping communication parameters of PDO1 to PDO4. CAN: Sub-indexes 1-3, 5 only exist. ECAT: Up to 4 entries.
1A00	TPDO mapping parameter	UINT32	CAN: RW ECAT: RO	No	CAN, ECAT. Transmit PDO mapping parameters; CAN: Up to 8 entries. ECAT: Up to 3 entries
1A01	TPDO mapping parameter	UINT32	CAN: RW ECAT: RO	No	CAN, ECAT. Transmit PDO mapping parameters; CAN: Up to 8 entries. ECAT: Up to 4 entries
1A02	TPDO mapping parameter	UINT32	CAN: RW ECAT: RO	No	CAN, ECAT. Transmit PDO mapping parameters; CAN: Up to 8 entries. ECAT: Up to 5 entries
1A03	TPDO mapping parameter	UINT32	CAN: RW ECAT: RO	No	CAN, ECAT. Transmit PDO mapping parameters; CAN: Up to 8 entries. ECAT: Up to 4 entries
1A04/6	TPDO mapping parameter	UINT32	RO	No	ECAT only. Transmit PDO mapping parameters;
1A07/8-1A08/8	TPDO mapping parameter	UINT32	RW	No	ECAT only. Transmit PDO mapping parameters;
1A0A/1	TPDO mapping parameter	UINT32	RO	No	ECAT only. Transmit PDO mapping parameters;
1A0B/2	TPDO mapping parameter	UINT32	RO	No	ECAT only. Transmit PDO mapping parameters.
1A0C/1 - 1A24/1	TPDO mapping parameter	UINT32	RO	No	ECAT only. Transmit PDO mapping parameters.
1C00/4	SM Communication type	UINT8	RO	No	ECAT only
1C10/0	SM0 PDO assignment	UINT16	RW	No	ECAT only, NOT TO BE USED (CTT only)

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
1C11/0	SM1 PDO assignment	UINT16	RW	No	ECAT only, NOT TO BE USED (CTT only)
1C12/30	SM2 (Outputs) PDO assignment	UINT16	RW	No	ECAT only
1C13/35	SM3 (Inputs) PDO assignment	UINT16	RW	No	ECAT only
1C32/32	Sync Manager 2 output parameters	UINT32, UINT16	Sub 1, 7, 8, 10: RW; Rest Sub: RO	No	ECAT only ,ECAT Outputs
1C33/32	Sync Manager 3 input parameters	UINT32, UINT16	Sub 0, 2, 6, 9, 11, 14, 32: RO, Rest sub: RW	No	ECAT only ,ECAT Inputs
2005/0	Fast reference	INT32	RW	Yes	CAN, ECAT
2012/0	Set binary Interpreter object	UINT64	WO	RxMap	CAN only. Map to rPDO2
2013/0	Get binary Interpreter object	UINT64	RO	TxMap	CAN only. Map to tPDO2
2020/5	Home Block limit parameters	UINT32, UINT16	Sub 0: RO, Sub 1-5: RW	No	CAN, ECAT. Sub 4 OV[64], Sub 5 OV[65]
2030/16	Upload recording data	UINT64	RO	No	CAN only
2035/0	Upload data parameters	UINT32	RW	No	CAN only
2036/0	Upload data (UL)	UINT64	RO	No	CAN only
2041/0	Time stamp uSec resolution	UINT32	RO	TxMap	CAN, ECAT
2045/0	Block upload Inhibit time parameter	UINT16	RW	No	CAN only
2046/0	Distributed clock inhibit time	UINT16	RW	No	ECAT only. In mSec
2051	Download data (DL)	UINT64	WO	No	CAN only
2060/0	Parameters Checksum	UINT16	RO	No	CAN, ECAT
2061/0	FoE Download Parameters Error	UINT16	RO	No	ECAT only
2062/0	FoE Parameters Last String Send To Drive	STRING	RO	No	ECAT only
207B/2	Additional Position range limit	INT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT. Modulo range
2081/5	Extended error code	INT32	RO	No	CAN, ECAT. Reflects **EE[]**
2082/0	CAN controller status	UINT32	RO	TxMap	CAN only, OV[60]
2085/0	Extra Status register	INT16	RO	TxMap	CAN, ECAT, OV[61]
2086/0	STO Status Register	UINT32	RO	No	CAN, ECAT, OV[62]

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
2087/0	PAL Version	UINT16	RO	No	CAN, ECAT
2090/0	CAN DF implementation	UINT32	WO	No	CAN only
20A0/0	Additional Position in UU	INT32	RW	TxMap	CAN, ECAT
20B0/9	Socket additional function	UINT32	Sub 0: R Sub 1-9: RW	No	CAN, ECAT
20E0/0	ECAT alias object	UINT16	RW	No	ECAT only
20FC/2	Absolute Sensors Functions	UINT16	WO	No	CAN, ECAT
20FD/0	Digital input (0x60FD alias)	UINT32	RW	No	CAN, ECAT. Allows write function
2201/0	Low byte of DS402 Digital inputs	UINT8	RO	TxMap	CAN only
2202/3	Extended input	UINT32	**ECAT:** Sub 0,1: RO, Sub 2,3: RW **CAN:** Sub 0: RO, Sub 1-3: RW	**ECAT:** Sub-index1 TxMap; **CAN:** Sub-index 1 TxMap	CAN, ECAT
2203/0	Application Object	UINT32	RO	TxMap	CAN, ECAT
2205/2	Analog input	INT16	RO	**ECAT:** Sub-index1 TxMap; **CAN:** Sub-index 1 Sub-index 2 TxMap	CAN, ECAT. Sub-index 1: in mVolts Sub-index 2: 0 – 4095 (A2D ticks)
2206/0	5V DC supply	UINT16	RO	TxMap	CAN, ECAT. In mVolts
22A0/0	Digital output	UINT8	RW	RxMap	CAN only, GP output only
22A1/3	Extended outputs	UINT32	Sub 0: RO, Sub 1-3: RW	**CAN:** Sub 1 RxMap, **ECAT:** Sub 1 RxMap	CAN, ECAT.
22A2/0	Drive Temperature in °C	UINT16	RO	TxMap	CAN only, Legacy object
22A3/3	Temperature	UINT16	RO	CAN Sub 1 TxMap	CAN, ECAT. Similar to **TI[]**
2E00/0	Gain scheduling manual index	UINT16	RW	RxMap	CAN, ECAT
2E06/0	Torque window	UINT16	RW	No	CAN only, OF[50], TR[5]
2E07/0	Torque window time	UINT16	RW	No	CAN only, OF[51], TR[6]

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
2E10/0	Set HOME Position according to last Touch Probe capture.	UINT16	RW	No	CAN, ECAT
2E15/0	Gantry YAW offset	INT16	RW	No	CAN, ECAT. Reflected in **TW[14]**
2F00/24	General purpose User Integer array	INT32	RW	CAN: RxMap, TxMap ECAT: No	CAN,ECAT. Reflects **UI[]**
2F01/24	General purpose User Float array	FLOAT	RW	CAN: RxMap, TxMap ECAT: No	CAN, ECAT. Reflects **UF[]**
2F05/0	Get drive control board type	UINT16	RO	No	CAN, ECAT. Similar to **WS[8]**
2F20/4	TPDO Asynchronous events	UINT32	RW	No	CAN only
2F21/0	Emergency event mask	UINT16	RW	No	CAN only
2F41/0	Configuration object	UINT32	RW	No	CAN, ECAT
2F45/4	Threshold parameter object	INT32	Sub 0: RO, Sub 1-4: RW	No	CAN, ECAT
2F70/2	CAN encoder range	INT32	Sub 0: RO, Sub 1-2: RW	No	CAN only
2F75/0	Extrapolation Cycles Timeout	INT16	RW	No	CAN, ECAT, OV[63]
0x3000 to 0x3300	Elmo legacy commands	UINT32	RW	No	CAN, ECAT
6007/0	Abort connection option code	INT16	RW	No	CAN, ECAT
603F/0	Error Code	UINT16	RO	No	CAN, ECAT
6040/0	Control word	UINT16	RW	RxMap	CAN, ECAT
6041/0	Status word	UINT16	RO	TxMap	CAN, ECAT
605A/0	Quick stop option code	INT16	RW	No	CAN, ECAT
605B/0	Shut down option code	INT16	RW	No	CAN, ECAT
605C/0	Disable operation option code	INT16	RW	No	CAN, ECAT
605D/0	Halt option code	INT16	RW	No	CAN, ECAT
605E/0	Fault reaction option code	INT16	RW	No	CAN, ECAT
6060/0	Modes of Operation	INT8	RW	**CAN:** RxMap, TxMap **ECAT:** RxMap	CAN, ECAT
6061/0	Modes Of operation display	INT8	RO	TxMap	CAN, ECAT

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
6062/0	Position demand value	INT32	RO	TxMap	CAN, ECAT
6063/0	Position actual internal value	INT32	RO	TxMap	CAN, ECAT
6064/0	Position actual value	INT32	RO	TxMap	CAN, ECAT
6065/0	Following error window	UINT32	RW	No	CAN, ECAT
6066/0	Following error time out	UINT16	RW	No	CAN, ECAT
6067/0	Position Window	UINT32	RW	No	CAN, ECAT
6068/0	Position Window time	UINT16	RW	No	CAN, ECAT
6069/0	Velocity sensor actual value	INT32	RO	TxMap	CAN, ECAT
606A/0	Sensor selection code	INT16	RW	No	CAN, ECAT
606B/0	Velocity demand value	INT32	RO	TxMap	CAN, ECAT
606C/0	Velocity actual value	INT32	RO	TxMap	CAN, ECAT. In accordance with 606A
606D/0	Velocity window	UINT16	RW	No	CAN, ECAT
606E/0	Velocity window time	UINT16	RW	No	CAN, ECAT
606F/0	Velocity threshold	UINT16	RW	No	CAN, ECAT
6070/0	Velocity threshold time	UINT16	RW	No	CAN, ECAT
6071/0	Target Torque	INT16	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6072/0	Maxl torque	UINT16	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6073/0	Max current	UINT16	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6074/0	Torque Demand	INT16	RO	TxMap	CAN, ECAT
6075/0	Motor rated current	UINT32	RW	No	CAN, ECAT
6076/0	Motor rated torque	UINT32	RW	No	CAN, ECAT
6077/0	Torque actual value	INT16	RO	TxMap	CAN, ECAT
6078/0	Current actual value	INT16	RO	TxMap	CAN, ECAT
6079/0	DC link circuit voltage	UINT32	RO	**ECAT**:TxMap	CAN, ECAT
607A/0	Target Position	INT32	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
607B/2	Position range limit	INT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT
607C/0	Home offset	INT32	RW	No	CAN, ECAT

Object(Hex) /Hi Sub(Dec)	Name	Data Type	Attribute	Mappable?	Comment
607D/2	Software position limit	INT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT
607E/0	Polarity (speed & position)	UINT8	RW	**ECAT**: RxMap **CAN**: RxMap, TxMap	CAN, ECAT
607F/0	Max profile velocity	UINT32	RW	No	CAN, ECAT
6080/0	Max motor speed	UINT32	RW	No	CAN, ECAT
6081/0	Profile velocity	UINT32	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6082/0	End velocity	UINT32	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6083/0	Profile acceleration	UINT32	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6084/0	Profile deceleration	UINT32	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6085/0	Quick stop deceleration	UINT32	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6086/0	Motion profile type	INT16	RW	No	
6087/0	Torque slope	UINT32	RW	**ECAT:** RxMap **CAN:** RxMap, TxMap	CAN, ECAT
6089/0	Position notation index	UINT8	RO	No	CAN only
608A/0	Position dimension index	UINT8	RO	No	CAN only
608B/0	Velocity notation index	UINT8	RO	No	CAN only
608C/0	Velocity dimension index	UINT8	RO	No	CAN only
608D/0	Acceleration notation index	UINT8	RO	No	CAN only
608E/0	Acceleration dimension index	UINT8	RO	No	CAN only
608F/2	Position encoder resolution	UINT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT
6090/2	Velocity Encoder resolution	UINT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT
6091/2	Gear ratio	UINT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT
6092/2	Feed constant	UINT32	Sub 0: RO, Sub	No	CAN, ECAT

Object(Hex) /Hi Sub(Dec)	**Name**	**Data Type**	**Attribute**	**Mappable?**	**Comment**
			1,2: RW		
6093/0	Position factor of DS402	UINT32	RO	No	CAN only
6094/0	Velocity encoder factor of DS402	UINT32	RO	No	CAN only
6095/0	Velocity_factor_1 of DS402	UINT32	RO	No	CAN only
6096/2	Velocity factor	UINT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT
6097/2	Acceleration factor	UINT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT
6098/0	Homing Method	INT8	RW	No	CAN, ECAT
6099/2	Homing speeds	UINT32	Sub 0: RO, Sub 1,2: RW	No	CAN, ECAT
609A/0	Homing acceleration	UINT32	RW	No	CAN, ECAT
60B0/0	Position offset	INT32	RW	**ECAT:** RxMap **CAN:** RxMap	CAN, ECAT
60B1/0	Velocity offset	INT32	RW	**ECAT:** RxMap **CAN:** RxMap	CAN, ECAT
60B2/0	Torque offset	INT16	RW	**ECAT:** RxMap **CAN:** RxMap	CAN, ECAT
60B8/0	Touch probe function	UINT16	RW	RxMap	CAN, ECAT
60B9/0	Touch probe status	UINT16	RO	TxMap	CAN, ECAT
60BA/0	Touch probe 1 positive edge	INT32	RO	TxMap	CAN, ECAT
60BB/0	Touch probe 1 negative edge	INT32	RO	TxMap	CAN, ECAT
60BC/0	Touch probe 2 positive edge	INT32	RO	TxMap	CAN, ECAT
60BD/0	Touch probe 2 negative edge	INT32	RO	TxMap	CAN, ECAT
60C0/0	Interpolation sub mode select	INT16	RW	No	CAN only
60C1/2	interpolation data record	INT32	Sub 0:RO, Sub 1,2: RW	RxMap	CAN only
60C2/2	interpolation time period	INT8	Sub 0:RO, Sub 1,2: RW	**CAN:** RxMap **ECAT:** Sub 1: RxMap	CAN, ECAT
60C4/6	interpolation data configuration	INT16	RW	No	CAN only
60C5/0	Max acceleration	UINT32	RW	No	CAN, ECAT
60C6/0	Max deceleration	UINT32	RW	No	CAN, ECAT
60E3/33	Supported Homing	UINT8	RO	No	CAN, ECAT

Object(Hex) /Hi Sub(Dec)	**Name**	**Data Type**	**Attribute**	**Mappable?**	**Comment**
	Methods				
60E4/0	Additional Position Actual Value	INT32	RO	No	CAN only
60E5/0	Additional Velocity Actual Value	INT32	RO	No	CAN only
60F2/0	Positioning option code	UINT16	RW	No	CAN, ECAT
60F4/0	Following error actual value	INT32	RO	TxMap	CAN, ECAT
60FA/0	Control effort	INT32	RO	TxMap	CAN, ECAT
60FC/0	Position demand internal value	INT32	RO	TxMap	CAN, ECAT
60FD/0	Digital inputs	UINT32	RO	TxMap	CAN, ECAT
60FE/2	Digital outputs	UINT32	Sub 0: RO, Sub 1,2: RW	**CAN:** Sub 1 RxMap, **ECAT:** Sub1 RxMap	CAN, ECAT
60FF/0	target velocity	INT32	RW	**CAN:** RxMap **ECAT:** RxMap	CAN, ECAT
6502/0	Supported Drive Modes	UINT32	RO	No	CAN, ECAT

Table 3-2: Object Dictionary

Chapter 4: Addressing Elmo Parameters via CANopen Objects

The following describes the method which an EtherCAT or CANopen host can address any of Elmo's variables via SDO. Refer to section 17.46 Objects 0x3000 to 0x32A3: Elmo parameters objects.

The method is designed to be used in a simple way by a PLC function blocks of any high-level programming language that can translate characters to numbers and using it to modify or inquire the relevant desired object.

4.1. Elmo Parameters via SDO Interface (CoE \ CANopen)

A host can address Elmo legacy commands via general object as follows:

Object 0x3xxx is defined as a general object for Elmo's 2-letters commands. These commands include two alphabetical letters:

<Letter1> <Letter2>

Each one of these commands have a corresponding object. The object number is defined as follows:

$$Object\ Number = 0x3000 + 26 * (Letter1 - 65) + (Letter2 - 65)$$

Where:

Letter1 and *Letter2* are the ASCII numbers presentation for example 'A' is 65, 'B' is 66 etc.

Every command is treated as an array command regardless to the legacy meaning, for example the legacy **MO** command should be addressed as **MO[1]** and with an SDO as: 0x3146.1.

Sub index 0 is a read-only object. The reply for sub index 0 will be the size of the specific array, for example the **UI[x]** command is a general purpose command that can be used by the user for any need. The command has 24sub indices. An SDO read access of the alias object: 0x3210.0 will be replied as 24 which mean that setting of 0x3210 sub index 24 is permitted.

Note: **Parameters where they include more than 255 entries, the sub index 0 cannot be presented. In such cases the return value is not valid. The Gold Command reference manual includes the alias objects per command.**

The sub index of the legacy array commands, such as **CA[x]** or **AN[x]**, is the array pointer 'x'.

Example:

Addressing the scalar parameter: **OP**, the object will be:

0x3000+('O'-65)*26 + ('P'-65) = 0x3000+(79-65)*26+(80-65) = 0x317B.1

For example the parameter **KI[2]** will be:

0x310C sub index 2.

4.2. Method Limits

The following limits apply to object 0x3000 to 0x32A3:

- The **AA[]** command cannot be access via this method
- The objects are not map able to PDO
- It is limited to expedite transfer where the reply from the host is up to 4 bytes
- No string commands are supported (e.g. **VR**)
- No uploading or downloading is supported (e.g. **BH**)
- The EtherCAT ESI file and CANopen EDS do not include the object list
- The objects cannot be retrieved via EtherCAT SDO “get info” or “complete access” methods

Chapter 5: Service Data Objects (SDOs)

Gold servo drives use a single transmit server SDO (COB 581h - 5ffh) and a single receive server SDO (COB601h-67fh). This is in accordance with the CiA predefined object list for 11-bit addressing. An SDO provides direct access to objects of an Elmo Drive Object Dictionary.

SDOs may be used to transfer multiple data sets from a client to a server and vice versa. The client controls the data to be transferred via a multiplexer (index and sub-index of the object dictionary). The content of the data set is defined in the object dictionary.

An SDO may be transferred as a sequence of segments. However, before transferring the segments there is an initialization phase where client and server prepare themselves for transferring the segments. An SDO can transfer a data set up to four bytes during the initialization phase. This mechanism is called **SDO expedited transfer**. If the data to be transferred contains more than four bytes, **SDO segmented transfer** is used.

When using SDOs, it is important to remember that:

- An SDO has a lower priority than a PDO.
- An SDO session is not complete until it is confirmed.
  For example, if an SDO is used to change a PDO mapping, the SDO should be issued only after the last session in which the PDO is completed, and the newly-mapped PDO should not be used until the SDO mapping change is confirmed.
- In an SDO data exchange, each client message **must** be backed by one and only one server message.
- An SDO carries a toggle bit, which varies in every consecutive message of a domain transfer, so that the loss of a single message can be tracked.
- Any SDO transfer can be terminated using the special **SDO abort transfer** message.
- An SDO message carries a maximum of seven bytes of data. One byte (the header byte) is always dedicated to command specifier and other header data (will be explained below).
- The length of an SDO message is always eight bytes, even if some of them are unused. Unused data bytes are marked as such in the message header.
- The maximum length of payload data in an expedited SDO is four bytes.

5.1. Initiate SDO Download Protocol

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

Client to server

Bytes: 0					**1-3**	**4-7**
Bits: 7…5	4	3…2	1	0	All bits	All bits
css = 1	**x**	**n**	**e**	**s**	**m**	**d (data)**

Server to client

Bytes: 0		**1-3**	**4-7**
Bits: 7…5	4…0	All bits	All bits
scs = 3	**x**	**m**	**reserved**

where:

css	Client command specifier 1: Initiate download request
scs	Server command specifier 3: Initiate download response
n	Number of bytes in d that do not contain data. Only valid if e = 1 and s = 1; otherwise it is 0. 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. Represents index/sub-index of data to be transferred by SDO.
d	Data e = 0, s = 0: d is reserved for future use. e = 0, s = 1: d contains number of bytes to be downloaded. Byte 4 contains LSB and byte 7 contains MSB. e = 1, s = 1: d contains data of length 4-n to be downloaded. The encoding depends on the type of data referenced by index and sub-index. e = 1, s = 0: d contains an unspecified number of bytes to be downloaded.
x	Not used; always 0.
reserved	Reserved for future use; always 0.

5.2. Download SDO Protocol

This protocol is used to implement the Download SDO Segment service.

Client to server

Bytes: 0				**1-7**
Bits: 7…5	4	3…1	0	
css = 0	**t**	**n**	**c**	**Segmented data**

Server to client

Bytes: 0			**1-7**
Bits: 7…5	4	3…0	
scs = 1	**t**	**x**	**reserved**

where:

css	Client command specifier 0: Download segment request
scs	Server command specifier 1: Download segment response
Segmented data	Maximum seven bytes of segment data downloaded. Encoding depends on type of data referenced by index and sub-index.
n	Number of bytes in **seg-data** that do not contain segment data. Bytes [8-n, 7] do not contain segment data. n = 0 if no segment size is indicated.
c	Whether or not there are still more segments to be downloaded: 0: More segments to be downloaded 1: No more segments to be downloaded
t	Toggle bit, which alternates for each subsequent segment to be downloaded. First segment has toggle-bit set to 0. Toggle bit is equal for request and response message.
x	Not used; always 0.

5.3. Initiate SDO Upload Protocol

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

Client to server

Bytes: 0		**1-3**	**4-7**
Bits: 7…5	4…0		
css = 2	**x**	**m**	**reserved**

Server to client

Bytes: 0					**1-3**	**4-7**
Bits: 7…5	4	3…2	1	0		
scs = 2	**x**	**n**	**e**	**s**	**m**	**d**

where:

css	Client command specifier 2: Initiate upload request
scs	Server command specifier 2: Initiate upload response
n	Number of bytes in **d** that do not contain data. Only valid if **e** = 1 and **s** = 1; otherwise it is 0. Bytes [8-n, 7] do not contain segment 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. Represents index/sub-index of data to be transferred by SDO.
d	Data **e** = 0, **s** = 0: **d** is reserved for future use. **e** = 0, **s** = 1: **d** contains the number of bytes to be uploaded. Byte 4 contains LSB and byte 7 contains MSB. **e** = 1, **s** = 1: **d** contains data of length 4-n to be downloaded. The encoding depends on the type of data referenced by index and sub-index. **e** = 1, **s** = 0: **d** contains an unspecified number of bytes to be uploaded.
x	Not used; always 0.
reserved	Reserved for future use; always 0.

5.4. Upload SDO Segment Protocol

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

Client to server

0			1-7
7…5	4	3…0	
css = 3	**t**	**x**	**reserved**

Server to client

0				1-7
7…5	4	3…1	0	
scs = 0	**t**	**n**	**c**	**Segmented data**

where:

css	Client command specifier 3: Upload segment request
scs	Server command specifier 0: Upload segment response
t	Toggle bit, which alternates for each subsequent segment to be uploaded. First segment has toggle-bit set to 0. Toggle bit is equal for request and response message.
c	Whether or not there are still more segments to be uploaded: 0: More segments to be uploaded 1: No more segments to be uploaded
Segment data	Maximum seven bytes of segment data uploaded. Encoding depends on type of data referenced by index and sub-index.
n	Number of bytes in **seg-data** that do not contain segment data. Bytes [8-n, 7] do not contain segment data. **n** = 0 if no segment size is indicated.
x	Not used; always 0.
reserved	Reserved for future use; always 0.

5.5. Abort SDO Transfer Protocol

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

Client to server *or* server to client

0		1-3	4-7
7…5	4…0		
Cs = 4	**X**	**M**	**D (data)**

where:

Cs	Command specifier 4: Abort transfer request
X	Not used; always 0.
M	Multiplexor. Represents index (bytes 1,2) and sub-index (byte 3) of SDO.
D	Four-byte abort error code (see Table 5-1: SDO Abort Codes) code giving reason for abort, encoded as Unsigned32 value.

The SDO Abort codes are listed in the following table:

Abort Error Code(Hex)	Description
0x05030000	Toggle bit not alternated
0x05040001	Invalid or unknown client/server command specifier
0x05040002	Invalid block size
0x05040003	Invalid sequence number in SDO block upload
0x05040005	Out of memory
0x06010000	Unsupported access to an object
0x06010001	Attempt to read a write-only object
0x06010002	Attempt to write a read-only object
0x06020000	Object does not exist in object dictionary
0x06040041	Object cannot be mapped to PDO
0x06040042	Number and length of objects to be mapped exceeds PDO length
0x06040043	General parameter incompatibility
0x06060000	Access failed due to hardware error
0x06070012	Data type does not match, service parameter too long
0x06090011	Sub-index does not exist
0x06090030	Value range of parameter exceeded (only for write access)
0x06090031	Value of parameter written too high

Abort Error Code(Hex)	Description
0x06090032	Value of parameter written too low
0x06090036	Maximum value is less than minimum value
0x08000000	General error. When the abort code is 0x08000000, the actual error can be retrieved using the **EC** command
0x08000020	Data cannot be transferred to or stored in application
0x08000022	Data cannot be transferred to or stored in application due to present device state
0x08000024	There is no data available to transmit

Table 5-1: SDO Abort Codes

Example of SDO abort transfer

1. SDO client tries to reset sub index 10 (does not exist in object dictionary) of object 0x60C1:

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
67F	**22**	**C1**	**60**	**0A**	**00**	**00**	**00**	**00**

2. The SDO server (ELMO drive) responds with SDO abort transfer, error code 0x06090011 “Sub index does not exist”

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
5FF	**80**	**C1**	**60**	**0A**	**11**	**00**	**09**	**06**

5.6. Uploading Data Using an SDO

Data is uploaded in two basic formats:

- A short data item (up to four bytes) is uploaded by a single message conversation, called an expedited SDO.
- Longer data items require a longer conversation and are called segmented transfers.

Example of segmented SDO upload transfer

The SDO is used to read from the drive object 0x100A *SW version:*

1. Client sends to server request *Initiate upload object 0x100A* (all in hexadecimal mode)

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**40**	**0A**	**10**	**00**	**00**	**00**	**00**	**00**

2. Server responds, indicating the length of data 0x21

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**41**	**0A**	**10**	**00**	**21**	**00**	**00**	**00**

3. Client request with toggle bit =0

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**60**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

4. Server responds: “Whistle” (characters are presented in ASCII code)

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**00**	**57**	**68**	**69**	**73**	**74**	**6C**	**65**

5. Client request with toggle bit =1

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**70**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

6. Server responds: “ 01.01”

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**10**	**20**	**30**	**31**	**2E**	**30**	**31**	**2E**

7. Client request with toggle bit =0

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**60**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

8. Server responds: ".07.13 0"

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**00**	**30**	**37**	**2E**	**31**	**33**	**20**	**30**

9. Client request with toggle bit =1

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**70**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

10. Server responds: "9Oct201"

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**10**	**39**	**4F**	**63**	**74**	**32**	**30**	**31**

11. Client request with toggle bit =0

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**60**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

12. Server responds: "3B01G" and informing last message 'c' bit

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**05**	**33**	**42**	**30**	**31**	**47**	**00**	**00**

Example of an expedited upload SDO transfer

The SDO is used to read from the drive object 0x1000. The value of this object stored in the drive is 32-bit word 0x00020192. The client message expedited SDO is outlined in the following Table 5-2:

Byte	Value	Description	Comment
0	0x40	Header	Client command specifier for SDO Upload Initiate
1	0x00	Index (LO)	
2	0x10	Index (HI)	
3	0	Sub-index	No sub-index; therefore set to 0
4 - 8	0	Reserved	

Table 5-2: Expedited SDO - Client Message

The Gold servo drive response is outlined in the following Table 5-3:

Byte	Value	Description	Comment
0	0x43	Header	Bits 7…5 = 010b: is client command specifier for SDO Upload Initiate. Bits 3, 2 =00: bits that indicate that all data bytes are relevant. Bits 1, 0 = 11b: indicates expedite transfer and bytes 4-7 contain data.
1	0x00	Index (LO)	
2	0x10	Index (HI)	
3	0	Sub-index	No sub-index, so it is set to 0
4	0x92	Data: 0x00020192 in little endian format	
5	0x01		
6	0x02		
7	0		

Table 5-3: Expedited SDO - Server Response

5.7. Downloading Data Using an SDO

Data downloading with SDOs is very similar to data uploading. It can be handled in a single message conversation (expedited transfer) or in a segmented conversation.

Example of an expedited download SDO transfer

The SDO is used to load value 0x50000 to object 0x607F:

1. SDO client sends download initiate SDO request

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x22**	**0x7F**	**0x60**	**0x00**	**0x00**	**0x00**	**0x00**	**0x00**

2. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x581	**0x60**	**0x7F**	**0x60**	**0x00**	**0x00**	**0x00**	**0x00**	**0x00**

Chapter 6: Process Data Objects (PDOs)

The Gold drive supports 4 TPDO and 4 RPDO. Each PDO contains up to 8 bytes of data. The data must be defined during the configuration process (typically in PreOP state). The process of structuring the PDO is named PDO mapping.

6.1. PDO Mapping

PDO mapping is a convention that assigns (maps) an object from the object dictionary (data payload) to a PDO. Once mapped, the PDO can carry the assigned data items without explicit reference to the object dictionary, thereby saving on communication and CPU overhead.

Some of the objects in the object dictionary can be mapped to a PDO, which can either receive (RPDO) or transmit (TPDO). The mapping of an RPDO enables reception of commands and variables — for example, efficient transmission of high-speed online motion commands to the drive — whereas the mapping of a TPDO enables the drive to send a predefined message in response to an event such as end of motion.

A TPDO is considered synchronous if triggered by a SYNC signal, and asynchronous if triggered by another event.

An RPDO is buffered upon reception; it is sent for interpretation immediately (when defined as asynchronous) or upon receipt of the next SYNC signal (when defined as synchronous).

6.1.1. Transmission Type

The transmission of a TPDO and RPDO is triggered by an event, which is defined by the PDO communication parameters: sub-index 2 of objects 0x1800 to 0x1803 (TPDO) and sub-index 2 of objects 0x1400 to 0x1403 (RPDO). These object dictionary entries are transmission types. The data type of the PDO communication parameter is described in object 0x20.

PDO transmission types can be one of the following:

Transmission Type	Description
0	Synchronous transmission performed once, at next SYNC which follows an asynchronous event.
N=1…240	Synchronous transmission performed once per 1 to 240 accepted SYNC signals.
254	Asynchronous transmission in response to a manufacturer-specific event.
255	Asynchronous transmission in response to a device profile (such as DSP 402).

The Elmo drive treats type 254 and 255 similarly.

6.1.2. The Synchronous Trigger

Synchronous triggers are always related to the SYNC reception.

If the RPDO transmission type is 1, the received message is buffered but actually transmitted for execution at the next SYNC message. Only one RPDO can be buffered for synchronous trigger. If another RPDO arrives before the SYNC, it overrides the previous RPDO without any notification. This method enables the simultaneous synchronization of executing commands in several drives. When a SYNC arrives, the buffered message is performed in the next 250uSec.

With TPDOs, the message is transmitted according to transmission type value, which can range from 1 to 240, where 1 indicates on each single SYNC, 2 means every second SYNC message, and so on.

Objects can receive data from SDOs and RPDOs simultaneously. Be aware that when this occurs, the results are unpredictable. The final value of the object may be either the SDO or the RPDO data.

6.1.3. The Asynchronous Trigger

Asynchronous triggers are defined in the device-specific protocol (such as DSP-402) or by the Elmo manufacture-specific object 0x2F20 (Described in chapter *Object 0x2F20: PDO events*). When the device-specific protocol is used, the transmission type is 254,255 and the asynchronous behavior is defined in the object description.

A transmission type 0 means that the message is transmitted non-periodically, after the occurrence of the SYNC, only if an event occurred before the SYNC **(Is not supported)**.

When the Elmo manufacture-specific object is used, each sub-index of object 0x2F20 defines trigger events for a single TPDO. The settings for the object go into effect only if the TPDO communication parameters are set to transmission type 254, 255 and if the TPDO is correctly mapped. The Elmo drive treats transmission type 254 and transmission type 255 alike.

6.1.4. Mapping Parameter Objects

Objects 0x1A00 to 0x1A03 contain the objects mapped to TPDOs. Objects 0x1600 – 0x1603 contain the objects mapped to RPDOs. The data type of PDO mapping is described in object 0x21.

6.1.5. Default Values

Default values of PDO mapping parameters are used at:

- Power up
- NMT communication reset (NMT 82h)
- NMT node reset (NMT 81h)

Default values of PDO mapping parameters are presented in section 16.19 Objects 0x1600 - 0x1603: Receive PDO mapping.

6.2. Receive PDOs

A Receive Process Data Objects (RPDO) is used to receive predefined and unconfirmed messages. An RPDO is received through use of an event, which may be asynchronous (such as *Message Received*) or synchronous with the reception of a SYNC. Four receive PDOs are used with the Elmo drive. The following must be indicated:

- Objects that can be mapped and have write access can be mapped to each RPDO.
- Execution of the mapped objects begins in the lower index of the relevant mapping object (0x1600 - 0x1603).

The following general rules apply to RPDOs:

- Mapable objects are described in the Object Dictionary.
- Synchronous RPDOs are processed at the next 250uSec after the reception. Asynchronous RPDOs are processed at the background process (typically 500uSec after reception).
- The transmission type associated to the Rx mapping can be either synchronous (transmission type = 1)where the data of the RPDO is passed for executing after the next SYNC signal was received, or asynchronous (transmission type=254, 255) in which the data of the RPDO is passed to the application immediately after reception.
- At most one copy of each mapped RPDO can be stored for synchronous execution. If the same RPDO arrives before next SYNC, it overwrites the previous with no notification.
- A change in RPDO mapping wipes any pending synchronous or asynchronous queued RPDO s of that type. This might result in time out.
- Change of transmission type from synchronous to asynchronous does not wipe pending instances.
- Changing transmission type to synchronous does not wipe any queued asynchronous instances.
- Objects can receive data from SDO and RPDO at the same time. The user must be aware that the result of such a situation cannot be predicted. The final value of the object may be either the SDO or the RPDO data.
- RPDOs cannot be retrieved by a remote transmitting (RTR).

6.2.1. RPDO Error Handling

When an PDO fails to be interpreted or transmitted, an emergency message 0x6300 is transmitted.

Several objects may be mapped into the same RPDO. The EMCY message identifies the objects that failed. A failure occurs when the received data cannot be interpreted or executed. In some cases, the Elmo error code is produced and may be included in the EMCY message. See detailed description of EMCY 0x6300 in Chapter 7: Emergency (EMCY).

In some cases Elmo's error code is produced. The emergency message may contain this error code. Refer to the detailed description in Chapter 7:.

6.2.2. Mapping procedure

The following procedure is used for re-mapping, which may take place during the **NMT** state Pre-operational and during the **NMT** state Operational:

1. Destroy RPDO/TPDO by setting the respective RPDO/TPDO communication parameter sub-index 01h bit valid to 1_b.
2. Disable mapping by setting the sub-index 0 of the relevant PDO object to 0.
3. Modify mapping by changing the values of the corresponding sub-indices.
4. Enable mapping by setting sub-index 0 to the number of mapped objects.
5. Create RPDO/TPDO by setting the relevant RPDO/TPDO communication parameter sub-index 1bit valid to 0_b.

If the device detects during step 3, that the index and sub-index of the mapped object does not exist or the object cannot be mapped, the   responds with the SDO abort transfer service (abort code:0x 0602 0000 or 0x0604 0041h).

If device detects that during step 4, that the RPDO/TPDO mapping is not valid or not possible, the CANopen device responds with the SDO abort transfer service
(abort code: 0602 0000h or 0604 0042).

If the device receives a PDO that has more data bytes than the number (length) of mapped data bytes, the CANopen device uses the first data bytes up to the length and may initiate the EMCY write service, if supported.

If a CANopen device receives a PDO having less data bytes than the number(length) of mapped data bytes, then the CANopen device will send EMCY code 0x8210 0x21.

Example of RPDO mapping

Two objects are mapped to RPDO1 in this example: CW object 0x6040 and Interpolated data record object 0x60C1 sub index 1. Transmission type “synchronous” is set.

Disable RPDO1, COBID 0x201:

1. SDO client sends

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x22**	**0x00**	**0x14**	**0x01**	**0x01**	**0x02**	**0x00**	**0x80**

2. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x581	**0x60**	**0x00**	**0x14**	**0x01**	**0x00**	**0x00**	**0x00**	**0x00**

3. Clear mapping RPDO1 SDO client sends

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x22**	**0x00**	**0x16**	**0x00**	**0x00**	**0x00**	**0x00**	**0x80**

4. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x581	**0x60**	**0x00**	**0x16**	**0x00**	**0x00**	**0x00**	**0x00**	**0x00**

5. Set CW, object 0x6040, 16 bit length in sub index 1 of RPDO1

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x22**	**0x00**	**0x16**	**0x01**	**0x10**	**0x00**	**0x40**	**0x60**

6. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x581	**0x60**	**0x00**	**0x16**	**0x01**	**0x00**	**0x00**	**0x00**	**0x00**

7. Set Interpolated data record, object 0x60C1,. Sub index 1, 32 bit length in sub index 2 of RPDO1

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x22**	**0x00**	**0x16**	**0x02**	**0x20**	**0x01**	**0xC1**	**0x60**

8. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

0x581	**0x60**	**0x00**	**0x16**	**0x02**	**0x00**	**0x00**	**0x00**	**0x00**

9. Set transmission type synchronous, every SYNC

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x22**	**0x00**	**0x14**	**0x02**	**0x01**	**0x00**	**0x00**	**0x00**

10. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x581	**0x60**	**0x00**	**0x14**	**0x02**	**0x00**	**0x00**	**0x00**	**0x00**

11. Set 2 objects are mapped

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x22**	**0x00**	**0x16**	**0x00**	**0x02**	**0x00**	**0x00**	**0x00**

12. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x581	**0x60**	**0x00**	**0x16**	**0x00**	**0x00**	**0x00**	**0x00**	**0x00**

13. Enable RPDO1

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x22**	**0x00**	**0x14**	**0x01**	**0x01**	**0x02**	**0x00**	**0x00**

14. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x581	**0x60**	**0x00**	**0x14**	**0x01**	**0x00**	**0x00**	**0x00**	**0x00**

6.3. Transmit PDOs

Four transmit PDOs can be used in Elmo drives. TPDOs are used to retrieve an object (data) from the drive. Objects that have read access and are map-able can be mapped to each one of the TPDOs. The transmitted data inside the TPDO is ordered according to the mapping order. The data starting from the LSB data is mapped first - in the lower index of the relevant mapping object.

Example of TPDO mapping

Two objects are mapped to TPDO1 in this example: SW object 0x6041 and actual position object 0x6064. Transmission type “synchronous” is set:

1. Disable TPDO1, COBID 0x181

COB	Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5	Byte 6	Byte 7
601	22	00	18	01	81	01	00	80

COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
581	60	00	18	01	00	00	00	00

2. Clear mapping TPDO1

COB	Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5	Byte 6	Byte 7
601	22	00	1A	00	00	00	00	00

COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
581	60	00	1A	00	00	00	00	00

3. Set Status Word, object 0x6041, 16 bit length in sub index 1 of TPDO1

COB	Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5	Byte 6	Byte 7
601	22	00	1A	01	10	00	41	60

COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
581	60	00	1A	01	00	00	00	00

4. Set actual position, object 0x6064, 32 bit length in sub index 2 of TPDO1

COB	Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5	Byte 6	Byte 7
601	22	00	1A	02	20	00	64	60

COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
581	60	00	1A	02	00	00	00	00

5. Set 2 objects are mapped

COB	Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5	Byte 6	Byte 7
601	22	00	1A	00	02	00	00	00

COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
581	60	00	1A	00	00	00	00	00

6. Set transmission type synchronous, every SYNC

COB	Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5	Byte 6	Byte 7
601	22	00	18	02	01	00	00	00

COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
581	60	00	18	02	00	00	00	00

7. Enable TPDO1

COB	Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5	Byte 6	Byte 7
601	22	00	18	01	81	01	00	00

COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
581	60	00	18	01	00	00	00	00

Chapter 7: Emergency (EMCY)

Gold servo drives issue an emergency code in response to an abnormal condition. All emergencies can be masked. For a description of emergency codes that can be masked, refer to object 0x2F21.

The Emergency object COB-ID is 0x81 to 0xFF. The structure of the manufacturer-specific emergency message is as follows:

0	Error code
1	
2	Error register
3	Elmo error code
4	Error code data field 1
5	
6	Error code data field 2
7	

Note: **Unused bytes must be set to zero.**

7.1. Emergency Codes

The following table (Table 7-1) lists the Emergency Error codes and their description with the relevant Elmo code where applicable.

EMCY name	Error Code (Hex)	Error Register (Hex)	Elmo Error Code (Dec)	Data Field
Short circuit	**2340**	03	0	0
Under-voltage	**3120**	05	0	0
AC fail, loss of phase	**3130**	11	0	0
Over-voltage	**3310**	05	0	0
Temperature: drive overheating	**4310**	09	0	0
Gantry position error	**5280**	81	0	0
Motor disabled by: • INHIBIT or ABORT and • FLS and RLS are switched on simultaneously in IP or CSP operation modes. Note that FLS\RLS are ignored when **XA[4]**=4 The function of Abort\Inhibit\FLS\RLS is defined via **IL[]** command. Refer to the Command reference.	**5441**	21	0	0
Motor disabled by switch “additional abort motion”	**5442**	81	0	0
RPDO failed	**6300**	01	Elmo Error Code see section 7.2	0
Motor stuck	**7121**	21	0	0
Feedback error	**7300**	81	0	0
Two digital Hall sensors were changed at the same time	**7381**	81	0	0
Commutation process fail during motor on	**7382**	81	0	0
CAN message lost (corrupted or overrun)	**8110**	11	Elmo Error Code see	0x4000 (Sync lost)

EMCY name	Error Code (Hex)	Error Register (Hex)	Elmo Error Code (Dec)	Data Field
			section 7.2	0x2000 (rPDO lost) 0x200 (NMT lost) 0x100 (SDO lost)
Heartbeat event	**8130**	11	Elmo Error Code see section 7.2	Node ID
Recovered from bus off	**8140**	11	Elmo Error Code see section 7.2	0
Attempt to access a non-configured RPDO	**8210**	21	0	0
Peak current has been exceeded. Possible reasons are drive malfunction or bad tuning of the current controller.	**8311**	21	0	0
Speed tracking error **DV[2]** - **VX** (for **UM=2** or **UM=4**, **5**) exceeded speed error limit **ER[2]**. This may occur due to: • Bad tuning of the speed controller • Too tight a speed error tolerance • Inability of motor to accelerate to the required speed due to too low a line voltage or insufficient motor power	**8480**	81	0	0
Speed limit exceeded: **VX**<**LL[2]** or **VX**>**HL[2]**. (Compatibility only)	**8481**	81	0	0
Position tracking error **DV[3**] - **PX** (**UM=5**) or **DV[3]** - **PY** (**UM=4**) exceeded position error limit **ER[3]**. This may occur due to: • Bad tuning of the position or speed controller • Too tight a position error tolerance • Abnormal motor load, or reaching a mechanical limit	**8611**	21	0	0
Position limit exceeded: **PX**<**LL[3]** or **PX**>**HL[3]** (**UM=5**), or **PY**<**LL[3]** or **PY**>**HL[3]** (**UM=4**). (Compatibility only)	**8680**	81	0	0

EMCY name	Error Code (Hex)	Error Register (Hex)	Elmo Error Code (Dec)	Data Field
Request by user program EMCY(N) function	**FF01**	81	0	0…0xFFFF input parameter N of EMCY(N) function
Either of the possibilities: • IP mode underflow • Interpolation queue full (overfow) • Reference received in a wrong index. This EMCY is transmitted when operating in Profile Interpolated mode, sub mode 0 and RPDO with object 0x60C1 sub index 2 was received. • Bad PVT send order	**FF02**	81	Elmo Error Code see section 7.2	0
Failed to start motor	**FF10**	81	Elmo Error Code see section 7.2	0
Safety Torque Off in use	**FF20**	05	0	0
Gantry Slave Disabled	**FF40**	81	0	0

Table 7-1 Emergency error codes

7.2. ELMO Error Codes

The following table (Table 7-2: ELMO error codes) lists the ELMO Error codes and their description with the relevant Elmo code where applicable.

Description	ELMO Error Code (Dec)	Remark
Will not be updated	**1**	
Bad command	**2**	
Bad index	**3**	
PAL does not support this sensor	**4**	
Mode cannot be started - bad initialization data	**7**	
CAN message was lost – hardware error	**9**	
Cannot be used by PDO	**10**	
Cannot write to flash memory	**11**	
Cannot reset communication - UART is busy	**13**	
Array '[ ]' is expected, or empty expression in array	**16**	
Format of **UL** command is not valid - check the command definition	**17**	
Command syntax error	**19**	
Bad Set Point sending order	**20**	
Operand out of range	**21**	
Zero division	**22**	
Command cannot be assigned	**23**	
Bad operation	**24**	
Profiler mode not supported in this unit mode (**UM**)	**26**	
Bad ECAM setting, refer to **EE[6]**	**27**	
Out Of limit range	**28**	
CAN get object return an abort when called from interpreter	**31**	
Communication overrun, parity, noise, or framing error	**32**	
Bad sensor setting, check **CA[18]**, **CA[19]**	**33**	

Description	ELMO Error Code (Dec)	Remark
There is a conflict with another command	**34**	
Max bus voltage (**BV**) or max current (**MC**) is not valid	**35**	
Commutation method (**CA[17]**) or commutation table does not fit to sensor	**36**	
Hall sensors are defined to the same place, check **CA[4]**, **CA[5]** or **CA[6]**	**37**	
PORT C mux is with conflict with other **GO[]** or cannot be assigned to the output	**38**	
Operating in Wizard experimental mode	**40**	
Command is not supported by this product	**41**	
No Such Label	**42**	
An attempt to read a write only object	**45**	
Program does not exist or not compiled	**47**	
Motor could not start - fault reason in **CD**	**48**	
ECAM must be disabled during init (**RM=0**)	**49**	
Stack overflow	**50**	
Inhibit OR Abort inputs are active, cannot start motor	**51**	
PVT queue full	**52**	
Bad database	**54**	
Bad context	**55**	
Motor must be off (**MO=0**)	**57**	
Servo (**SO**) must be on	**58**	
Bad unit mode	**60**	
Database reset	**61**	
Socket change not allowed while capture is enabled	**62**	
Amplifier not ready	**66**	
Recorder is busy or data is uploading	**67**	
Required profiler mode is not supported	**68**	

Description	ELMO Error Code (Dec)	Remark
Recorder usage error	**69**	
Recorder data Invalid	**70**	
Homing is busy	**71**	
Modulo range must be even	**72**	
Please set position	**73**	
Bad profile database, see **EE[2]** or 0x2081 for the failed object number	**74**	
Download is in progress	**75**	
Error mapping is not allowed while Homing is in progress (**HM[1]\HF[1]**)	**76**	
Out of program range	**78**	
Sensor setting error, possible conflict with other socket settings	**79**	
ECAM data inconsistent	**80**	
Download failed see specific error in **EE[3]**	**81**	
Program is running	**82**	
Command is not permitted in a program	**83**	
STO is not active OR During STO Diagnostic	**85**	
Reserved	**87**	
Not allowed while Error mapping (**PC[1]**) is in progress	**94**	
User program time out	**96**	
RS232 receive buffer overflow	**97**	
Current offsets are beyond the allowed limit	**98**	
Bad auxiliary sensor configuration	**99**	
The requested PWM multiplication value is not supported	**100**	
Absolute encoder setting problem	**101**	
Output Compare (**OC[1] \ OC[21]**) or Emulation (**EA[1]**) are busy	**102**	

Description	ELMO Error Code (Dec)	Remark
Output compare sensor is not QUAD Encoder	**103**	
Output Compare Table Length OR Data	**104**	
Speed loop KP out of range	**105**	
Encoder emulation parameter (**EA[2] to EA[7]**) is out of range	**107**	
Encoder emulation (**EA[1]**) is already in progress	**108**	
Too long number	**110**	
Please wait until analog sensor initialized	**121**	
Motion mode is not supported or with initialization conflict	**122**	
Profiler queue is full	**123**	
Personality not loaded	**125**	
User Program failed - variable out of program size	**126**	
Bad variable index in database - internal compiler error	**128**	
Variable is not an array	**129**	
Variable name does not exist	**130**	
Cannot record local variable	**131**	
Variable is an array	**132**	
Number of function input arguments is not as expected	**133**	
Cannot run local label/function with **XQ** command	**134**	
Frequency identification failed	**135**	
Not a number	**136**	
Position Interpolation buffer underflow	**138**	
The number of break points exceeds maximal number	**139**	
An attempt to set/clear break point at the not relevant line	**140**	
Boot Identity parameters section is not clear	**141**	

Description	ELMO Error Code (Dec)	Remark
Checksum of data is not correct	**142**	
Numeric Stack underflow	**144**	
Numeric stack overflow	**145**	
Executable command within math expression	**147**	
Nothing in the expression	**148**	
Parentheses mismatch	**151**	
Bad operand type	**152**	
Overflow in a numeric operator	**153**	
Address is out of data memory segment	**154**	
Beyond stack range	**155**	
Bad op-code	**156**	
Out of flash memory range	**158**	
Flash verify error	**159**	
Program is not halted	**161**	
Not enough space in program data segment	**163**	
An attempt to access flash while busy	**165**	
Out of modulo range	**166**	
Speed too large to start motor	**168**	
Time out using peripheral.(overflow or busy)	**169**	
Cannot erase sector in flash memory	**170**	
Cannot read from flash memory	**171**	
Cannot write to flash memory	**172**	
Executable area of program is too large	**173**	
Program has not been loaded	**174**	
Cannot write program checksum - clear program (**CP**)	**175**	
User code, variables and functions are too large	**176**	
Capture/Compare conversion error in analog encoder.	**177**	

Description	ELMO Error Code (Dec)	Remark
CAN bus off	**178**	
Consumer **HB** event	**179**	
DF is not supported in this communication type	**180**	
Writing to Flash program area, failed	**181**	
PAL Burn Is In process or no PAL is burnt or incompatible PAL version	**182**	
Capture option already used by other operation	**184**	
This element may be modified only when interpolation is not active	**185**	
Interpolation queue is full	**186**	
Incorrect Interpolation sub-mode	**187**	
Gantry slave disable	**188**	
CAN message was lost - software	**189**	
Profile Acceleration is out of range	**190**	
Motor over temperature	**191**	
Main feedback error. Refer to **EE[1]**	**200**	
Commutation sequense failed	**201**	
Encoder-Hall sensor mismatch. Refer to **XP[7]**	**202**	
Current limit was exceeded	**203**	
External inhibit input detected	**204**	
AC fail: Loss of phase	**205**	
Digital halls run too fast or disconnected	**206**	
Speed error limit exceeded. Refer to **ER[2]**	**207**	
Position limit error exceeded. Refer to **ER[3]**	**208**	
Cannot start motor . Bad data base. Refer to **CD**	**209**	
Bad ECAM table	**210**	
Cannot find zero position without digital halls	**216**	
Over speed. Refer to **HL[2]**	**217**	

Description	ELMO Error Code (Dec)	Remark
Motor stack	**221**	
Out of position limits. Refer to **HL[3]**, **LL[3]**	**222**	
Numerical overflow	**223**	
Gantry slave is not enabled	**224**	
Cannot start motor because of internal problem	**229**	
Undervoltage protection	**233**	
Overvoltage protection	**235**	
Safety switch	**237**	
Short protection	**241**	
Over temperature protection	**243**	
Additional inhibit input	**245**	

Table 7-2: ELMO error codes

Example of Emergency message

Client tries to enable the motor while the power is too low, using Ds402 Control Word (CW, object 0x6040):

Sets Control Word =0xF (Motor On), but motor voltage is not switched on

COB	Byte 0	Byte 1	Byte 2	Byte3	Byte 4	Byte 5	Byte 6	Byte 7
67F	22	40	60	00	0F	00	00	00

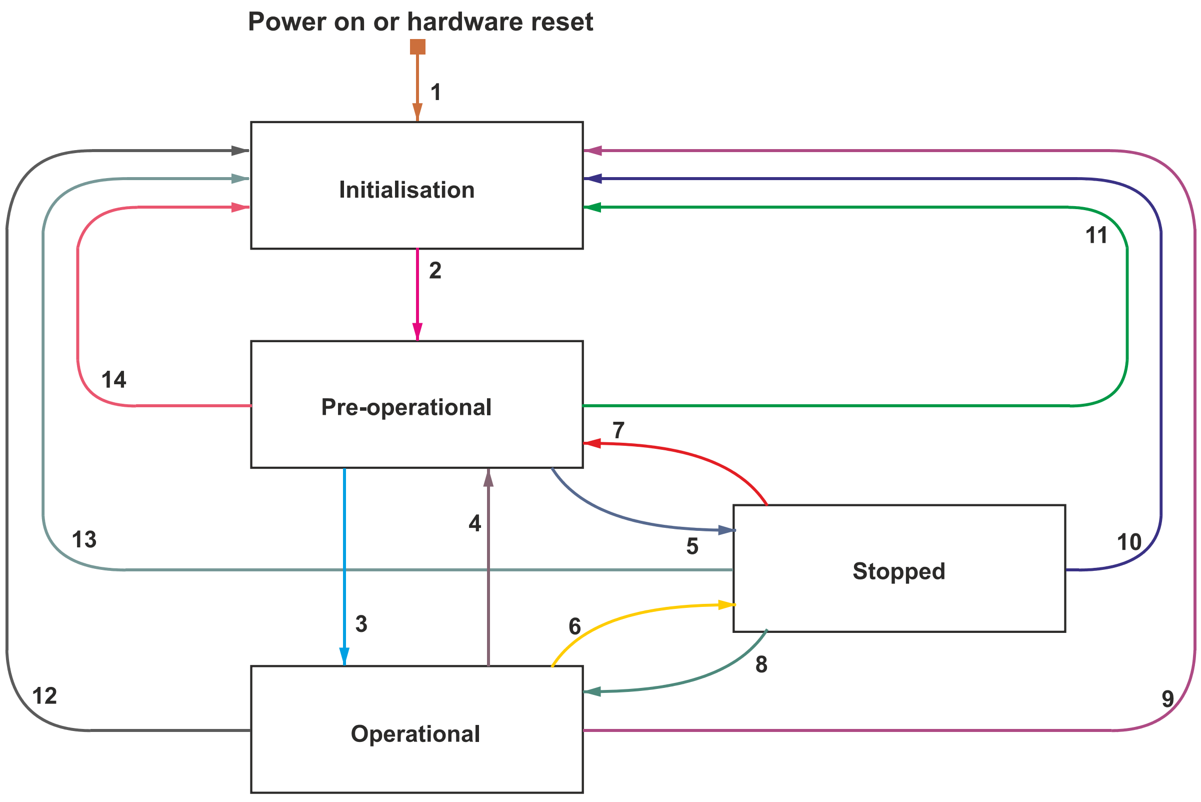
COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
5FF	60	40	60	00	00	00	00	00

ELMO drive (ID=0x7F) sends emergency message with COB ID= 0x80+ID= 0xFF and EMCY error code 0x3120, error register 0x05, means "Under Voltage"

COB	Byte0	Byte 1	Byte 2	Byte3	Byte 4	Byte5	Byte 6	Byte 7
FF	20	31	05	00	00	00	00	00

Chapter 8: Network Management (NMT)

The NMT state diagram of an ELMO drive is presented in Table 8-1. Device enters NMT state Pre-operational immediately after ending the initialization. In this NMT state it is possible to transfer SDOs. Then the device can be switched to NMT state Operational. The NMT state machine determines the behavior of the drive.



1	At Power on the NMT state Initialisation is entered autonomously
2	NMT state Initialisation finished - enter NMT state Pre-operational automatically
3	NMT service start remote node indication or by local control
4, 7	NMT service enter pre-operational indication
5, 8	NMT service stop remote node indication
6	NMT service start remote node indication
9, 10, 11	NMT service reset node indication
12, 13, 14	NMT service reset communication indication

G-DS301001A

Table 8-1 NMT State Diagram

8.1. NMT States

Only the minimum, required, set of network management (NMT) services is supported by the Gold Line. **NMT** commands are used to control the communication state of the servo drive and to broadcast manufacturer messages to all other connected servo drives.

The following network communication states are supported:

State	Description
Unpowered/Initialization	Servo drive is not ready, or it is booting. Drive will not respond to communication and will not transmit anything.
Pre-operational	Servo drive boot sequence is complete, but no command has been received to enter operational mode. The servo drive will respond to SDO and NMT messages, but not to PDOs.
Operational	Servo drive is fully operational, responding to PDO, SDO and NMT messages.
Stopped	Servo drive can respond only to NMT objects (including heartbeats).

Table 8-2: Network Management (NMT)

When the servo drive is powered on, it enters the initialization state. After completing the boot sequence, it automatically enters the pre-operational state. The transition between pre-operational, operational and prepared states is carried out according to Network Management (NMT) messages. The COB-ID of an **NMT** command is always 0.

An NMT message is always two bytes long: the first byte is the command specifier and the second byte is the ID of the units that are to respond to the message. If the ID is 0, the NMT message is executed by the entire set of connected servo drives.

The following NMT services are supported:

Command Specifier	Service
1	Start remote node (go to operational).
2	Stop remote node (go to prepared).
128 (0x80)	Enter pre-operational state.
129 (0x81)	Reset node (perform full software reset, non-volatile parameters are loaded from flash).
130 (0x82)	Reset communication (reload communication parameters from flash, reset CAN mapping, send boot up message and enter pre-operational state).

Table 8-3: Supported NMT Services

Note: **It is recommended that you turn off the motor and kill any user program before executing NMT 130.**

Examples of NMT message

1. Master sends NMT message *reset node ID 03*

COB	Byte 0	Byte 1
00	81	03

Where: 0x00 is COBID of NMT message;

0x81 is command *reset*;

0x03 is ID of node to be reset

2. Master sends NMT message *reset communication for all nodes*

COB	Byte 0	Byte 1
00	82	00

Where: 0x00 is COBID of NMT message;

0x82 is command *reset communication*;

0x00 means ''for all nodes''

Chapter 9: Boot-up Messages

9.1. Boot-up Event

Via this service, the ELMO drive indicates that a local state transition has occurred, from the NMT Initialization state to the NMT state Pre-operational.

The COB ID of the message must be **0x700 + Node ID**

Examples of boot-up message

After power up ELMO drive (ID=0x7F) sends

COB	Byte 0
77F	00

Where: 0x77F is COBID of boot-up message, 0x700 + 0x7F;

0x00 data, always 0;

Chapter 10: Heartbeat Messages

The Elmo drive can be configured as a Heartbeat Consumer by settings object 0x1016 or Heartbeat Producer by settings object 0x1017. When the drive is configured as **consumer**, it can indicate either of the following:

- A heartbeat error occurred
- A heartbeat error has been resolved
- An NMT state change has occurred

If the **heartbeat producer time** defined by 0x1017 object is configured on the Elmo drive, the producer heartbeat protocol begins immediately and transmits producer heartbeat messages periodically. If however, the Elmo drive starts with a value for the heartbeat producer time unequal to 0, the heartbeat protocol starts on the transition from the NMT Initialization state to the NMT Pre-operational state. In this case the boot-up message is regarded as the first heartbeat producer message.

One or more heartbeat consumer Elmo drives can receive the producer heartbeat messages. The heartbeat consumer Elmo drive checks the reception of the heartbeat messages within the heartbeat consumer time. If the heartbeat is not received within this **heartbeat consumer time** defined by the 0x1016 object, a heartbeat event is generated and heartbeat emergency is sent (if not masked). For a description of emergency codes that can be masked, refer to object 0x2F21.

An Elmo drive defined as a consumer, checks the reception of heartbeat producer messages in Operational, Pre-operational and Stopped states. When the consumer is in Stopped state and a heartbeat event occurs, the heartbeat emergency cannot be sent immediately, it will be sent after switching to Pre-operational state.

In addition, an Elmo drive defined as a consumer, starts monitoring of the heartbeat producer after the reception of the first heartbeat producer message. The consumer heartbeat time should be higher than the corresponding producer heartbeat time. Prior to the reception of the first heartbeat, the status of the heartbeat producer is unknown.

10.1. Producer message

The Producer message contains COB ID = **0x700 + Producer ID** and only one data byte that contains the producer state as follows:

0:	Boot-up
4:	Stopped
5:	Operational
127:	Pre-operational

Example of producer message

0x720 0x05

Where: Producer ID=0x20, Producer state = operational

10.2. Consumer HB EMCY message

The structure of EMCY messages is described in Chapter 7:Emergency (EMCY). The field “Elmo error code ” contains 0xB3, the field “data” contains producer ID.

Example of HB EMCY message that was sent from ELMO drive

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x81	**0x30**	**0x81**	**0x11**	**0xB3**	**0x20**	**0x00**	**0x00**	**0x00**

Where:

0x81	COB ID= 0x80+0x01 (Drive ID)
0x8130	EMCY error code *Heartbeat event*, see chapter *Emergency codes*
0x11	Error register
0xB3	ELMO error code *Consumer heartbeat event*, see chapter *ELMO error codes*
0x20	Producer ID

10.3. Support of two heartbeat producers

Elmo drive can be configured as consumer supporting two heartbeat producers with different IDs.

Example

1. Set Consumer period =10sec, Producer ID=0x20 to sub ind 1

Time	COBID	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0
06:39:03.849.2	601	22	16	10	01	10	27	20	00
06:39:03.849.5	581	60	16	10	01	00	00	00	00

2. Set Consumer period =15sec, Producer ID=0x30 to sub ind 2

   06:39:03.860.0 601 22 16 10 02 98 3A 30 00

   06:39:03.860.0 581 60 16 10 02 00 00 00 00

Time	COBID	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0
06:39:03.860.0	601	22	16	10	02	98	3A	30	00
06:39:03.860.0	581	60	16	10	02	00	00	00	00

3. delay 20000 ms
4. Here will not be EMCY because of the first HB is not received yet

   Producer sends heartbeat messages:

Time	COBID	Byte0
06:39:23.904.5	720	05
06:39:23.915.1	730	05
06:39:32.944.3	720	05
06:39:32.954	730	05

5. delay 20000 ms
6. When Producer stops sending its message the drive sends

   EMCY 81 30 81 11 B3 20 00 00 00 in 10 sec

   after the last producer message and

   EMCY 81 30 81 11 B3 30 00 00 00 in 15 sec

   after the last producer message

Time	COBID	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0	Byte 0
06:39:42.962.3	81	30	81	11	B3	20	00	00	00
06:39:47.981.0	81	30	81	11	B3	30	00	00	00

Chapter 11: SYNC Messages

An Elmo drive can receive a SYNC message and use it for two purposes:

- Synchronize the operation of synchronous TPDOs and RPDOs. Synchronous TPDOs can be sent by the Elmo drive upon only receiving a SYNC message. The data of received synchronous RPDOs can only be processed on receiving a SYNC message.
- Synchronize the motion clock of the Elmo drive with a clock of the network master. In this case, the drives are synchronized by the reception of a SYNC message, whose arrival time is captured by the drive. Upon reception of the SYNC message, the drive adjusts its internal timer. Refer to the MAN-G-DS402 chapter: IP mode implementation.

Chapter 12: Time Stamp

Elmo drives support read only Time Stamp object 0x2041 that can be read by SDO and can be mapped to TPDO. The object presents a Stamp Timer that counts microseconds (regardless of the sampling time of the drive).

The Time Stamp is cyclic and has 32 bits, and therefore completes a full cycle in 4,295 seconds (approximately 72 minutes). The timer is updated every 2*TS (sampling time), and by default the Time stamp is updated every 100 microseconds by a value of 100.

Chapter 13: Binary Interpreter Commands

Elmo commands can be addressed with CAN via the proprietary Binary Interpreter protocol. By using a simple 8 bytes format, the Binary Interpreter is designed to simplify the way by which CAN master can get and set Elmo commands. By default, PDO2 is mapped to the Binary Interpreter objects. In that way, the CAN host can address the Elmo commands immediately after power up.

The Binary Interpreter uses the 2-letters format where the first two bytes are the explicit command for example in order to get the **AC** command, the first two transmitted bytes will be ASCII value of 'A' & 'C' which is 0x41 & 0x43 respectively. For detailed information about the Elmo commands, please refer to the *Gold Command Reference manual*.

It should be noted that due to the simplified format of the Binary interpreter, it does not support visible string such as **VR** command. For the same reason, it does not support expressions such as:

UI[1]=UI[1]+100.

The following table summarizes the main differences between the binary interpreter used for CAN communication and the ASCII interpreter used for RS-232.

Feature	ASCII Interpreter	Binary Interpreter
Command length	Depends on data	Fixed: 8 bytes for Set commands; 4 bytes for Get commands
Delimiter	; or <CR> for commands and servo drive responds	None
Servo drive responds to Set commands	Always	Drive does not respond to Set commands. An emergency object is sent if command execution fails
Long response strings	Returned by certain commands, such as LS and BH	No support for returned long strings, which are read via SDOs

Table 13-1: Comparison of ASCII vs. Binary Interpreter Commands

RPDO2 is mapped by default to the receive binary interpreter object (0x2012) and TPDO2 is mapped by default to the transmit binary interpreter object (0x2013). TPDO2 is transmitted as an unsynchronized *Binary Interpreter complete* event.

The binary interpreter supports three types of commands:

- **Set value**
  These commands are eight bytes in length. The transmitted message includes either the reflection of the Set command or an error code, if a failure has occurred.
- **Get value**
  These commands can be four or eight bytes in length. An 8-byte response includes the reflection of the command and the resulting numerical value, and an error if a fault has occurred.
- **Execute command**
  This command can be four or eight bytes in length. An 8-byte response includes the

reflection of the command and the resulting numerical value, and an error if a fault has occurred.

If an interpreter command cannot be serviced for any reason, bit 6 in byte 3 of TPDO2 is set on, and byte 4 of the response contains the Elmo error code. Consider that RPDO and TPDO are used for the Binary Interpreter, meaning that to use the Binary Interpreter, the drive must be in OPERATIONAL NMT state (**NMT** start node command must be sent before).

13.1. Binary Interpreter Commands and Results

The sequences in this section illustrate the binary interpreter options for setting, querying and executing commands.

13.1.1. Set and Query Commands

The host (client) sends commands (**RPDO2**) for setting variables in eight bytes (DLC=8). The drive (server) transmits the reply (**TPDO2**) as an asynchronous event of the received object.

13.1.1.1. RPDO2 Structure

RPDO2 is used to set values for the drive and query (get) values from it. The structure of the command is as follows:

- Bytes 0 to 3 are the header, which includes the command, command index (when needed) and data type (float or integer).
- Bytes 4 to 7 are the data, which is always four bytes. The format can be integer or float. The bytes are interpreted in little endian format.

The following table describes the format in which the host sends commands to the drive:

Byte	**0**	**1**	**2**	**3**			**4 - 7**
Bits	0…7	0…7	0…7	0…5	6	7	
Description	First command character	Second command character	Index for array parameter. 0 for scalar command		See note	0: Integer 1: float	Data in little endian format

Note: **Bytes 0 and 1, which represent the command character in ASCII, must be uppercase.**

Byte 3, bit 6

When this bit is set to 1, the drive treats the command as a *query* and not as a *setting*. In this case, the rest of the data bytes are discarded and the drive replies to the command according to 4 bytes DLC. For compatibility reasons, bytes 4 to 7 should be 0.

In array commands in which the index is used (as in **ET[100]**), the lowest significant bits are in byte 2 (bits 0 to 7) and the most significant bits are in byte 3.

Always use the bit 7 of byte 3 to indicate the data type (float or integer) in the transmitted message, even if the numerical data type is known in advance and given in the reference manual. This enables Elmo to guarantee that the type of numerical data returned for any interpreter command will remain unchanged in future versions.

Example 1

```
CL[1] is set to 1.0, which is 3F800000h in hex IEEE format.
CL[1]=1.0
```

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Hex value	43	4C	01	80	00	00	80	3F

Note: **Bit 7 in byte 3 is set to 1 to indicate that the value is float.**

Example 2

`AC is set to 150,000 (0249F0h):`
`AC=150000`

Byte	0	1	2	3	4	5	6	7
Hex value	41	43	0	0	F0	49	02	0

Example 3

`AC is queried and the DLC is 4:`
`AC`

Byte	0	1	2	3
Hex value	41	43	0	0

Example 4

`This is the same query as in Example 3, but with a` **`DLC of 8`** `(`*`Byte 3, bit 6`* `note).`
`AC`

Byte	0	1	2	3	4	5	6	7
Hex value	41	43	0	40	0	0	0	0

In both Example 3 and Example 4, the reply from the server is:

Byte	0	1	2	3	4	5	6	7
Hex value	41	43	0	0	F0	49	02	0

Example 5

`CA[18] = 4096` (0x1000) (18 in decimal - 12h in hex)

Byte	0	1	2	3	4	5	6	7
Hex value	43	41	12	0	0	10	0	0

Example 6

In this example, the server replies to the command **ET[992]** (3E0h), assuming that the value is 32121 (7D79h). This is done to query the DLC 8 format (bit 6 in byte 3 is set):
ET[992] (3E0h)

Byte	0	1	2	3	4	5	6	7
Hex value	45	54	E0	43	0	0	0	0

The server replies as follows:

Byte	0	1	2	3	4	5	6	7
Hex value	45	54	E0	03	79	7D	0	0

13.1.1.2. TPDO2 Structure

The server (drive) replies (TPDO2) to query and set requests in eight bytes (DLC=8):

- Bytes 0 to 3 are the header, which includes the responding command, command index (when needed) and data type (float or integer). It also indicates whether the response data is true data or an error code.
- Bytes 4 to 7 are data, which is either a reflection of the host Set command or an error code according to the EC command.

Byte	0	1	2	3			4 - 7
Bits	0…7	0…7	0…7	0…5	6	7	
Description	First character	Second character	Index for array parameter. 0 for scalar command		See note.	0: Integer 1: Float	Valid data or error code. Little endian format.

Note: **Bytes 0 and 1 represent the command character and must be uppercase.**

Byte 3, bit 6

When this byte is 1 for TPDO, the data in bytes 4 to 7 should be interpreted as an error code. Refer to the EC command section in the Gold Line *Reference Manual* for details.

Note: **In array commands in which the index is used (as in ET[100]), the lowest significant bits are in byte 2 (bits 0 to 7) and the most significant bits are in byte 3.**

Example 1

The server replies to the command **CA[1]=4**, which is out of range: error code 21 (15h).

Byte	0	1	2	3	4	5	6	7
Hex value	43	41	01	40	15	0	0	0

Example 2

In this example, the server replies to the command **ET[992]** (0x3E0), assuming that the value is 32121 (7D79h).

The server replies as follows:

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Hex value	45	54	E0	03	79	7D	0	0

13.1.2. Execute Command

These commands are used to instruct the drive to perform a sequence. The reply to these commands is only an acknowledgement or an error code; there is no value for executing command. Execute commands are a unique case of RPDO2, which can be used with a DLC of either 4 or 8.

Example

BG command, to start a motion.

DLC4:

Byte	**0**	**1**	**2**	**3**
Hex value	42	47	0	0

DL8:

Byte	**0**	**1**	**2**	**7**
Hex value	42	47	0	0

The reply is always eight bytes long and indicates either success or failure (error).

Success

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Hex value	42	47	0	0	0	0	0	0

Failure: error code 58 (3Ah) for *Motor must be on*

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Hex value	42	47	0	40	3A	0	0	0

13.2. ASCII Interpreter Commands not Supported by Binary Interpreter

Commands that deal with strings are not accessible using the binary interpreter. In most cases, these strings may be accessed using the OS interpreter prompt. The binary interpreter cannot handle expressions, which must be dealt with using the OS interpreter.

Command	Description	Alternative
VR	Detailed software version string	Use OS prompt instead of SDO to read object 0x100a
CD	CPU dump in case of fatal exception	Use OS prompt
LS/DL	List/download from serial flash	Use OS prompt
DF	Download firmware version	Use SDO to write object 0x2090
BH	Bring recorded value	Use SDO to read object 0x2030
XC##\XQ##	Execute user program	Use OS prompt

Chapter 14: The OS Interpreter

The OS interpreter is used to process any *Gold Line* interpreter string command, and to return the string results. The only limitation in its use is that the returned strings cannot exceed 500 characters in length, a limit that must be considered when uploading recorded data. A more efficient — and unlimited — method to upload recorder data is to use object 0x2030.

To issue an OS interpreter command:

1. Set OS mode to evaluate the string immediately, by writing 0 to object 0x1024.
2. Write the command string to object 0x1023, sub-index 1.
   The command execution can be resolved by an event-driven PDO (object 0x2F20) or by polling object 0x1023, sub-index 2. The polling may return:
   - 0xFF: Command still executing (can be aborted by writing 3 to object 0x1024).
   - 0x1: Command successfully executed. Result is waiting for read.
   - 0x3: Command rejected. Error code waiting for read.

When the response is ready, it can be read from object 0x1023, sub-index 3.

Example

The following describes the use of the OS interpreter to send the command **PX=1234**.

Client initiates OS Evaluate Immediately mode:

RSDO	Object 0x1024		Sub-index	Data			
23	24	10	00	00	00	00	00

Server replies:

TSDO	Object 0x1024		Sub-index	Data			
60	24	10	00	00	00	00	00

Client initiates segmented SDO download:

RSDO	Object 0x1023		Sub-index	Data			
21	23	10	01	00	00	00	00

Server replies:

TSDO	Object 0x1023		Sub-index	Data			
60	23	10	01	00	00	00	00

Client sends PX=1234 in one SDO:

RSDO	P	X	=	1	2	3	4
01	50	58	3D	31	32	33	34

Server acknowledges that the RSDO was received OK:

TSDO			Sub-index	Data			
20	00	00	00	00	00	00	00

Client gets PX value from OS interpreter (assuming OS was already defined as *Evaluate Immediately*:

RSDO	Object 0x1023		Sub 1	P	X		
23	23	10	01	50	58	00	00

Server acknowledges that the RSD0 was received OK:

TSDO	Object 0x1023		Sub-index	Data			
60	23	10	01	00	00	00	00

Client queries status of command:

RSDO	Object 0x1023		Sub-index	Data			
40	23	10	02	00	00	00	00

Server replies that command was executed and that the result is waiting:

TSDO	Object 0x1023		Sub-index	Execute OK			
42	23	10	02	01	00	00	00

Client queries for reply value:

RSDO	Object 0x1023		Sub-index	Data			
42	23	10	03	00	00	00	00

Server replies with value of valid PX:

TSDO	Object 0x1023		Sub-index	1	2	3	4
43	23	10	03	31	32	33	34

Chapter 15: The EDS

The Electronic Data Sheet (EDS) assists CANopen configuration personnel in determining which objects a CAN slave supports. The EDS has a standard format that is explained in CiA DS-301, version 4. This document defines an optional read-only object used to upload the EDS directly from the CAN slave.

- Object 0x1021 is the EDS, stored as an ASCII string – *future implementation*
- Object 0x1022 defines the EDS compression style, which must be 0, for No compression - *future implementation*

The EDS is loaded to the internal serial flash memory of the Gold digital servo drive as part of the firmware download process – *future implementation*

Chapter 16: Communication Profile Objects

16.1. Object 0x1000: Device type

This object contains information about the device type and functionality. It is comprised of a 16-bit field that describes the device profile used, and a second 16-bit field that gives additional information about optional functionality of the device.

MSB			**LSB**
Byte 4	**Byte 3**	**Byte 2**	**Byte 1**
Additional information		Device profile number	

- Object description:

Index	**0x1000**
Name	Device type
Object code	VAR
Data type	UNSIGNED32
Category	Mandatory

- Entry description:

Access	Read only
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x020192

16.2. Object 0x1001: Error register

This object is an error register for the device.

- Object description:

Index	0x1001
Name	Error register
Object code	VAR
Data type	UNSIGNED8
Category	Mandatory

- Entry description:

Access	Read only
PDO mapping	No
Value range	0…255
Default value	0

- Data description (M for Mandatory and O for Optional):

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

If a bit is set to 1, the specified error has occurred. The only mandatory error that must be signaled is the generic error, which is signaled in any error situation.

16.3. Object 0x1002: Manufacturer status register

This object is a common status register for manufacturer-specific purposes. It returns the status similar to the **SR** command.

- Object description:

Index	0x1002
Name	Manufacturer status register
Object code	VAR
Data type	UNSIGNED32
Category	Optional

- Entry description:

Access	Read only
PDO mapping	Yes
Value range	0…(2^{32})-1
Default value	

16.4. Object 0x1003: Pre-defined error field

The object contains the last 16 errors that were reported by the drive via the EMCY message.

For details about the EMCY Error Code refer to the Chapter 7: Emergency (EMCY).

Sub-index 0 contains the number of actual errors recorded in the array, starting at sub-index 1. Up to 16 errors can be retrieved.

If there are no errors, the value of sub-index 0x00 is 0x00 and a read access of any of the other allowed sub-indexes is responded by an SDO abort message (abort code: 0x08000024).

Writing a 0 to sub-index 0 empties the array. An Abort message (error code: 0x06090030) is transmitted on attempt to write values higher than 0. Writing to object 0x2F21 also empties the array.

Every new error is stored in sub-index 0x01, older errors are moved to the next higher sub-index.

If an error is present in sub index 1…16, the read SDO returns with a data field containing the predefined error field that has the following structure:

MSB	**LSB**
Bits 31…16	**Bits 15…0**
Not used, filled with 0	EMCY error code

- Object description:

Index	**0x1003**
Name	Pre-defined error history
Object code	ARRAY
Data type	UNSIGNED32
Category	Mandatory

- Entry description:

Sub-index	**0**
Description	Number of actual errors
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…16
Default value	0

Sub-index	**1 - 16**
Description	Standard error field
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0…65535
Default value	0

16.5. Object 0x1006: Communication Cycle Period

This object is relevant for SYNC Producers only. This object provides the communication cycle period. This period defines the SYNC interval in microseconds. If the value is set to 0000 0x0000 the transmission of SYNC messages is disabled.

Elmo drive does not support the SYNC producer feature. The object is implemented for compatibility reason.

- Object description:

Index	0x1006
Name	Communication Cycle Period
Object code	VAR
Data type	UNSIGNED32
Category	Conditional, Mandatory for SYNC products

- Entry description:

Sub-index	**0**
Description	
Entry category	
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0000 0x0000

16.6. Object 0x1008: Manufacturer device name

This object contains the manufacturer device name, such as *Guitar*, *Trombone* and *Whistle*.
The manufacturer device name can be read from the segmented SDO

- Object description:

Index	0x1008
Name	Manufacturer device name
Object code	VAR
Data type	VISIBLE STRING
Category	Optional

- Entry description:

Access	Constant
PDO mapping	No
Value range	
Default value	

16.7. Object 0x1009: Manufacturer hardware version

This object contains the version number of the manufacturer's hardware. The **WS[30]** command contains the hardware version as a 32-bit unsigned integer, while this object conveys the information as a hexadecimal number. For example, if **WS[30]** is equal to 0x1400A, the string returned by this object will be 0x1400A.

- Object description:

Index	**0x1009**
Name	Manufacturer hardware version
Object code	VAR
Data type	VISIBLE STRING
Category	Optional

- Entry description:

Access	Constant
PDO mapping	No
Value range	
Default value	No

Example (in Hexadecimal)

1. Client sends SDO Get object 1009

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**40**	**09**	**10**	**00**	**00**	**00**	**00**	**00**

2. Drive responds with segmented SDO upload, marking 7 characters to be uploaded

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**41**	**09**	**10**	**00**	**07**	**00**	**00**	**00**

3. Client sends

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**60**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

4. Drive responds with **WS[30]**=0x1400A

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**01**	**30**	**78**	**31**	**34**	**30**	**30**	**41**

16.8. Object 0x100A: Manufacturer software version

This object contains the version identification of the manufacturer's software, similar to **VR** command.

- Object description:

Index	0x100A
Name	Manufacturer software version
Object code	VAR
Data type	VISIBLE STRING
Category	Optional

- Entry description:

Access	Constant
PDO mapping	No
Value range	No
Default value	No

The object can be read by segmented SDO. See example of segmented SDO upload transfer in chapter 4.6 Uploading Data Using SDPO.

16.9. Object 0x100B: Node ID

This object contains the node ID of the drive. If the node ID is changed, the object will return the updated value only after Reset Communication and Start Communication NMT messages have been sent.

- Object description:

Index	**0x100B**
Name	Node ID
Object code	VAR
Data type	UNSIGNED8
Category	Optional

- Entry description:

Access	Read only
PDO mapping	No
Value range	1…127
Default value	127

16.10. Object 0x1010: Store parameters

This object is used to save parameters in non-volatile memory. Through read access, the drive provides information about its save capabilities, using:

- Sub-index 0: Largest supported sub-index
- Sub-index 1: Save all parameters
- Read access to sub index 1 returns 1

Write access is protected. In order to avoid accidental storage, it is only executed when a specific signature save (Lower case only can be used) is written to the sub index 1.

Example of write SDO to object 0x1010 sub index 1

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**22**	**10**	**10**	**01**	**73**	**61**	**76**	**65**

- Object description:

Index	0x1010
Name	Store parameters
Object code	RECORD
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Largest supported sub-index
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	1
Default value	1

Sub-index	**1**
Description	Save all parameters
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	
Default value	No

16.11. Object 0x1011: Restore parameters

This object is used to restore parameters from non-volatile memory. Through read access, the drive provides information about its restore capabilities, using:

- Sub-index 0: Largest supported sub-index
- Sub-index 1: Restore all parameters
- Read access to sub index 1 returns 1

Write access is protected. In order to avoid accidental restore, it is only executed when a specific signature load (Lower case only can be used) is written to the appropriate sub index 1.

Example of write SDO to object 0x1011 sub index 1

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**22**	**11**	**10**	**01**	**6c**	**6f**	**61**	**64**

- Object description:

Index	**0x1011**
Name	Restore parameters
Object code	RECORD
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Largest supported sub-index
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	1
Default value	1

Sub-index	**1**
Description	Restore all parameters
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	
Default value	No

16.12. Object 0x1016: Consumer heartbeat time

The consumer heartbeat time defines the expected heartbeat cycle time and thus has to be higher than the corresponding producer heartbeat time configured on the device producing this heartbeat. Monitoring starts after the reception of the first heartbeat. If the consumer heartbeat time is 0, the corresponding entry is not used. The time period must be a multiple of 1 ms.

The bit placement of the object entries is displayed in the table.

UNSIGNED32 **MSB**			**LSB**
Bits	31…24	23…16	15…0
Value	0	Node-ID	Heartbeat time, in milliseconds
Encoding	-	Unsigned8	Unsigned16

If more than one consumer heartbeat entry is configured for the same Node ID, the device aborts the SDO download with abort code 0604 0043h.

- Object description:

Index	**0x1016**
Name	Consumer heartbeat time
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Largest supported sub-index
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	1, 2
Default value	2

Sub-index	**1, 2**
Description	Consumer heartbeat time
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…0x7FFFFF
Default value	0

Refer to chapter 9 Heartbeat messages for more details.

16.13. Object 0x1017: Producer heartbeat time

This object defines the cycle time of the heartbeat, which must be a multiple of 1 millisecond. It is 0 if not used.

- Object description:

Index	0x1017
Name	Producer heartbeat time
Object code	VAR
Data type	UNSIGNED16
Category	Mandatory

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	0…65535
Default value	0

Refer to chapter 9 Heartbeat messages for more details.

16.14. Object 0x1018: Identity object

This object stores the LSS address used for the CAN ID and baud rate setting.

- Object description:

Index	0x1018
Name	Identity object
Object code	RECORD
Data type	UNSIGNED32
Category	Mandatory

- Entry description:

Sub-index	0
Description	Number of entries
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	4
Default value	4

Sub-index	1
Description	Vendor ID
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x9A

Sub-index	2
Description	Product code
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x30924 or 0x30925

Sub-index	**3**
Description	Revision number
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0…(2^{32})-1
Default value	None

Sub-index	**4**
Description	Serial number
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0…(2^{32})-1
Default value	None

16.15. Object 0x1023: OS command and prompt

This object is used with the **OS** interpreter (see **Chapter 10**).

- Object description:

Index	0x1023
Name	**OS** command
Object code	RECORD
Data type	**OS** command record
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of entries
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	3
Default value	3

Sub-index	**1**
Description	Command
Entry category	Optional
Access	Write only
PDO mapping	No
Value range	String octet
Default value	None

Sub-index	**2**
Description	Status: 1: Last command completed, reply ready 3: Last command rejected, reply ready 255: Executing
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	1, 3, 255
Default value	1

Sub-index	**3**
Description	Reply
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	String octet
Default value	None

16.16. Object 0x1024: OS command mode

This object is used with the **OS** interpreter (see **Chapter 10**).

- Object description:

Index	**0x1024**
Name	**OS** command mode
Object code	VAR
Data type	UNSIGNED8
Category	Optional

- Entry description:

Access	Write only
PDO mapping	No
Value range	0…3 0: Execute next command immediately 1 - 2: Not supported 3: Abort execution
Default value	0

16.17. Object 0x1029: Error behavior

This object reports the CAN communication state after a heartbeat failure. The value of the object asserts that after such a failure, the CAN communication state is:

0: If current state is Operational then go to Pre-operational

1: No state change

2: Stopped

- Object description:

Index	0x1029
Name	Error behavior
Object code	ARRAY
Data type	UNSIGNED8
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of error classes
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	1
Default value	1

Sub-index	**1**
Description	Communication error
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…255
Default value	1

16.18. Objects 0x1400 - 0x1403: Receive PDO communication parameter

- Object description:

This object contains the details about the Receive PDOs communication method and data.

- Value definition:

Sub-index 0x00 contains the number of valid object entries within the record.

Sub-index 0x01 contains the COB-ID of the RPDO, see table below.

MSB					LSB
Bits	**31**	**30**	**29**	**28 - 11**	**10 - 0**
meaning	valid=0, Not valid=1	Reserved	Frame, always 0	0	11 bit CAN-ID

The bit valid (bit 31) enables RPDOs.

There may be PDOs configured (e.g. by default) but not used, and therefore set to *not valid* (deleted).

Attempt to set bit 29 (frame) to 1_b is responded with the SDO Abort Transfer service (abort code: 0609 0x0030).

It is not allowed to change bit 0 to 30 while the PDO exists and is valid (bit 31 = 0). In this case, when attempting to change the values from bit 0 to bit 30, the device responds with the SDO abort transfer service (abort code: 0609 0x0030).

Changes to CAN-ID field , bits 0…28 is allowed only to the value 0 or default value. When attempting to change the CAN-ID to another value, the device responds with the SDO abort transfer service (abort code: 0609 0x0030).

Index	**0x1400 – 0x1403**
Name	Receive PDO Parameter
Object code	RECORD
Data type	PDO communication parameter record (object 0x20)
Category	Conditional: mandatory for each supported PDO

- Entry description:

Index	**0x1400…0x1403**
Sub-index	**0**
Description	Number of entries
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	2
Default value	2

Index	**0x1400**
Sub-index	**1**
Description	COB-ID used by PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0….(2^{32})-1
Default value	0x27F

Index	**0x1401**
Sub-index	**1**
Description	COB-ID used by PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0….(2^{32})-1
Default value	0x37F

Index	**0x1402**
Sub-index	**1**
Description	COB-ID used by PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0….(2^{32})-1
Default value	0x47F

Index	**0x1403**
Sub-index	**1**
Description	COB-ID used by PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0….(2^{32})-1
Default value	0x57F

Index	**0x1400…0x1403**
Sub-index	**2**
Description	Transmission type
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	1…255
Default value	255

Sub-index 02h defines the transmission type of the RPDO (see Table below). Transmission type may be 255, 254 or 1. An attempt to change the value of the transmission type to any unsupported value is responded with the SDO abort
transfer service (abort code: 0609 0x0030).

16.18.1. Description of RPDO transmission type, sub index 2

Value	Description
0x00	Synchronous , acyclic
0x01	Synchronous, cyclic, every 1 SYNC
0x2…0xFD	Reserved
0xFE	Event driven (manufacturer specific)
0xFF	Event driven (device profile and application profile specific)

The Elmo drive treats type 254 and 255 similarly

Synchronous type means that the device will actuate the received data with the reception of the next SYNC (see Figure below). Take into account: when PDO is defined as synchronous it may be sent by producer in time more than 250 microseconds before next sync (See Chapter 11: SYNC Messages).

Event-driven means PDO may be received at any time the device will actuate the data immediately. See description of object 0x2F20 for detailed information related to defined events.

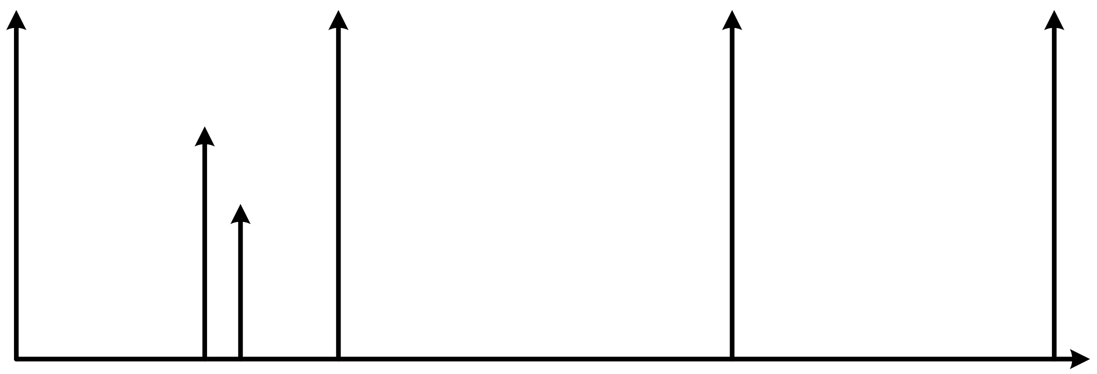


Table 16-1: Synchronous PDO timing

16.19. Objects 0x1600 - 0x1603: Receive PDO mapping

These objects contain the mapping for the PDOs that the Elmo drive is able to receive. The sub-index 0 contains the number of valid entries within the mapping record. This number of entries is also the number of the application variables that are received with the corresponding PDO.

- Maximum 8 object can be mapped to single PDO, so the length of PDO data can be 8X8=64 bits
- The sub-indexes from 1 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 PDO mapping element is a 32-bit field (object 0x21), divided as follows:

MSB		LSB
Index: 16-bit	Sub-index: 8-bit	Object length: 8 bits

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

Sub-index 0 determines the valid number of objects that have been mapped. To change the PDO mapping, sub-index 0 must be set to 0 (mapping is deactivated). Only then can the objects be remapped.

When a new object is mapped by writing a sub-index between 1 and 8, if the object does not exist or it cannot be mapped, the SDO transfer is aborted with the Abort SDO Transfer Service.

After all objects are mapped, sub-index 0 is set to the valid number of mapped objects. Then the drive rechecks the mapping integrity. Finally, the PDO is created by writing to its communication parameter COB-ID.

When sub-index 0 is set to a value greater than 0, the device may validate the new PDO mapping before transmitting the response of the SDO service. If an error is detected, the Elmo drive transmits the Abort SDO Transfer Service with abort codes 0602 0x0000, 0604 0x0041 or 0604 0x0042.

When the sub-index 0 is read, the actual number of valid mapped objects is returned.

- Object description:

Index	0x1600 – 0x1603
Name	Receive PDO Mapping
Object code	RECORD
Data type	PDO Mapping
Category	Conditional: mandatory for each supported PDO

- Entry description:

Index	**0x1600**
Sub-index	**0**
Description	Number of mapped application object in PDO
Entry category	Optional
Access	Read/write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	1(CAN), 3 (ECAT)

Index	**0x1601, 0x1602**
Sub-index	**0**
Description	Number of mapped application object in PDO
Entry category	Optional
Access	Read/write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	1(CAN), 2 (ECAT)

Index	**0x1603**
Sub-index	**0**
Description	Number of mapped application object in PDO
Entry category	Optional
Access	Read/write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 4 (ECAT)

Index	**0x1600**
Sub-index	**1**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x60400010 (CAN), 0x607A0020 (ECAT)

Index	**0x1600**
Sub-index	**2**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60FE0120 (ECAT)

Index	**0x1600**
Sub-index	**3**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60400010 (ECAT)

Index	**0x1600**
Sub-index	**4-8**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), none (ECAT)

Index	**0x1601**
Sub-index	**1**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x20120040 (CAN) , 0x60FF0020 (ECAT)

Index	**0x1601**
Sub-index	**2**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60400010 (ECAT)

Index	**0x1601**
Sub-index	**3-8**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), none (ECAT)

Index	**0x1602**
Sub-index	**1**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60710010 (ECAT)

Index	**0x1602**
Sub-index	**2**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60400010 (ECAT)

Index	**0x1602**
Sub-index	**3-8**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), none (ECAT)

Index	**0x1603**
Sub-index	**1**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x607A0020 (ECAT)

Index	**0x1603**
Sub-index	**2**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60FE0120 (ECAT)

Index	**0x1603**
Sub-index	**3**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60B10020 (ECAT)

Index	**0x1603**
Sub-index	**4**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60400010 (ECAT)

Index	**0x1603**
Sub-index	**5-8**
Description	PDO mapping for application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), none (ECAT)

Up to 8 objects can be mapped to a single RPDO.

Dummy entries are supported by the Elmo drive.

The object can be mapped during the PRE OPERATIONAL & Operational state.

The SDO client is responsible for data consistency.

16.20. Objects 0x1800 - 0x1803: Transmit PDO communication parameter

This object contains the communication parameters for the PDOs that Elmo drive is able to transmit.

- Value definition:

Sub-index 0 contains the number of valid object entries within the record. Its value is at least 5.

Sub-index 0x1 contains the COB-ID of the TPDO (see table below).

MSB		**Sub-index 1**			**LSB**
Bits	**31**	**30**	**29**	**28 - 11**	**10 - 0**
meaning	valid=0, Not valid=1	RTR 0 – RTR allowed on this PDO 1 – no RTR allowed	Frame, always 0	0	11 bit CAN-ID

The bit valid (bit 31) enables RPDOs. There may be PDOs configured (e.g. by default) but not used, and therefore set to *not valid*.

Attempting to set bit 29 (frame) to 1_b or bit 30 (RTR) to 0_b is responded with the SDO abort transfer service (abort code: 0609 0x0030).

It is not allowed to change bit 0 to 30 while the PDO exists and is valid (bit 31 = 0b). In this case, when attempting to change the values from bit 0 to bit 30, the device responds with the SDO abort transfer service (abort code: 0609 0x0030).

Changes to CAN-ID field , bits 0…28 is allowed only to the value 0 or default value. When attempting to change the CAN-ID to another value, the device responds with the SDO abort transfer service (abort code: 0609 0x0030).

- Object description:

Index	**0x1800 – 0x1803**
Name	Transmit PDO parameter
Object code	RECORD
Data type	PDO communication parameter record (object 0x20)
Category	Conditional: mandatory for each supported PDO

- Entry description:

Sub-index	**0**
Description	Number of entries
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	5
Default value	5

Index	**0x1800**
Sub-index	**1**
Description	COB-ID used by PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	Not applicable
Default value	0x400001FF

Index	**0x1801**
Sub-index	**1**
Description	COB-ID used by PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	Not applicable
Default value	0x400002FF

Index	**0x1802**
Sub-index	**1**
Description	COB-ID used by PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	Not applicable
Default value	0xC00003FF

Index	**0x1803**
Sub-index	**1**
Description	COB-ID used by PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	Not applicable
Default value	0xC00004FF

Index	**0x1800**
Sub-index	**2**
Description	Transmission type
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…240, 254,255
Default value	0xFF

Index	**0x1801**
Sub-index	**2**
Description	Transmission type
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…240, 254,255
Default value	0xFE

Index	**0x1802, 0x1803**
Sub-index	**2**
Description	Transmission type
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…240, 254,255
Default value	0

Sub-index	**3**
Description	Inhibit time
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…65536
Default value	0 (no inhibit time between messages)

Sub-index	**4**
Description	Reserved
Entry category	
Access	
PDO mapping	
Value range	
Default value	

Sub-index	**5**
Description	Event timer
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…65536
Default value	0

- **COB ID used by PDO**
  Only the default COB ID can be written to the drive. An attempt to write another COB ID will result in an abort (Abort Code 0609 0x0030).
- RTR is not supported
- Sub-index 0x02 defines the transmission type of the TPDO (see Table). An attempt to change the value of the transmission type to any not supported value shall be responded with the SDO abort transfer service (abort code: 0609 0x0030).

16.20.1. Description of RPDO transmission type, sub index 2

Value	Description
0x00	Synchronous , acyclic
0x01	Synchronous, cyclic, every 1 SYNC
--	--
0xF0	Synchronous, cyclic, every 240 SYNC
0xF1	Reserved
--	--
0xFD	Reserved
0xFE	Event driven (manufacturer specific)
0xFF	Event driven (device profile and application profile specific)

The Elmo drive treats type 254 and 255 similarly

Synchronous cyclic (sub index 2 = 1…240) means that the TPDO is transmitted after every N number of SYNCs are received, where N is the value of sub index 2.
Synchronous acyclic transmission (sub index 2 =0)means that the TPDO is transmitted once only after receiving the first sync after TPDO is enabled.
Event-driven means that the PDO may be transmitted at any time based on the occurrence of drive internal event. See the description of object **0x2F20** for detailed information related to defined events.

16.20.1.1. Inhibit Time

Inhibit time (sub index 3) defines the minimum time interval to transmit a TPDO. The resolution of the inhibit time is 100 microseconds. If sub index 3 is set to 0, then the minimum time interval to transmit TPDO is disabled.

16.20.1.2. Event Timer

When a TPDO transmission type is 254 or 255, an event timer defined by sub index 5 can be used. The timer defines the maximum interval for TPDO transmission. The event occurs when the time is elapsed. The event timer resolution is 1ms. The event causes the transmission of this TPDO in addition to other asynchronous events. The occurrence of an event sets the timer again. A value of 0 disables this function.

16.21. Objects 0x1A00 - 0x1A03: Transmit PDO mapping parameter

These objects contain the mapping parameter for the PDOs that the Elmo drive is able to transmit. Sub-index 0 contains the number of valid entries within the mapping record. This number of entries is also the number of the application variables that are transmitted with the corresponding TPDO.

- Object description:

Index	0x1A00- 0x1A03
Name	Transmit PDO mapping
Object code	RECORD
Data type	PDO mapping parameter record (object 0x21)
Category	Conditional: mandatory for each supported PDO

- Entry description:

Index	**0x1A00**
Sub-index	**0**
Description	Number of mapped application objects in PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…8
Default value	1, (CAN), 3 (ECAT)

Index	**0x1A01**
Sub-index	**0**
Description	Number of mapped application objects in PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…8
Default value	1, (CAN), 4 (ECAT)

Index	**0x1A02**
Sub-index	**0**
Description	Number of mapped application objects in PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…8
Default value	0 (CAN), 5 (ECAT)

Index	**0x1A03**
Sub-index	**0**
Description	Number of mapped application objects in PDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…8
Default value	0 (CAN), 4 (ECAT)

Index	**0x1A00**
Sub-index	**1**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x60410010 (CAN), 0x60640020 (ECAT)

Index	**0x1A00**
Sub-index	**2**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60FD0020 (ECAT)

Index	**0x1A00**
Sub-index	**3**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60410010 (ECAT)

Index	**0x1A00**
Sub-index	**4-8**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), none (ECAT)

Index	**0x1A01**
Sub-index	**1**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x20130040 (CAN), 0x60640020 (ECAT)

Index	**0x1A01**
Sub-index	**2**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x606B0020 (ECAT)

Index	**0x1A01**
Sub-index	**3**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60740010 (ECAT)

Index	**0x1A01**
Sub-index	**4**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60410010 (ECAT)

Index	**0x1A01**
Sub-index	**5-8**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), none (ECAT)

Index	**0x1A02**
Sub-index	**1**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN) , 0x60640020 (ECAT)

Index	**0x1A02**
Sub-index	**2**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN) , 0x60770010 (ECAT)

Index	**0x1A02**
Sub-index	**3**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN) , 0x60410010 (ECAT)

Index	**0x1A02**
Sub-index	**4**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN) , 0x60610008 (ECAT)

Index	**0x1A02**
Sub-index	**5**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x00020008 (ECAT)

Index	**0x1A02**
Sub-index	**6-8**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN) , none (ECAT)

Index	**0x1A03**
Sub-index	**1**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0 (CAN), 0x60640020 (ECAT)

Index	**0x1A03**
Sub-index	**2**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0, (CAN), 0x60FD0020 (ECAT)

Index	**0x1A03**
Sub-index	**3**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0, (CAN), 0x606C0020 (ECAT)

Index	**0x1A03**
Sub-index	**4**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0, (CAN), 0x60410010 (ECAT)

Index	**0x1A03**
Sub-index	**5-8**
Description	PDO mapping for nth application object to be mapped
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0, (CAN), none (ECAT)

Up to eight objects can be mapped to a single TPDO.

Dummy entries are supported by the Elmo drive.

The object can be mapped during PRE OPERATIONAL and OPERATIONAL stage.

Dynamic mapping is allowed during the OPERATIONAL stage. The SDO client is responsible for data consistency.

Sub-index from 1 to 8 contains the information of the mapped application objects. The object describes the content of the PDO by their index, sub-index and length (see Figure below).

The length contains the length of the application object in bit. This may be used to verify the mapping.

MSB $\longleftrightarrow$ LSB		
Index: 16-bit	Sub-index: 8-bit	Object length: 8 bits

An attempt to change the value of an object entry to any value that is not supported is responded with the SDO abort transfer service. The cause for a not supported value could be the mapping (index and sub-index) of a non-existing application object, a wrong length for the mapped application object, or a wrong length for the PDO at all.

Chapter 17: Manufacturer-specific Objects

17.1. Object 0x2005: Fast Reference

The reference object allows a direct access to a socket. Sockets are virtual entities which allow access to feedback functions. These functions include for example speed calculations, commutation calculations, position, filters etc. The socket can also be used as a reference for follower \ ECAM mode or virtual profiler which can be used as emulation to other drives. The object can be accessed by the **OV[48]** command.

- Object description:

Index	0x2005
Name	Fast reference
Object code	VAR
Data type	INTEGER32
Category	Mandatory

- Entry description:

Access	Read/Write
PDO mapping	Yes
Value range	No
Default value	No

17.2. Object 0x2012: Binary interpreter input

This object is a binary interpreter object (refer to Chapter 13: Binary Interpreter Commands concerning the byte stream).

- Object description:

Index	0x2012
Name	Binary interpreter
Object code	VAR
Data type	Binary interpreter query (object 0x42)
Category	Mandatory

- Entry description:

Access	Write only
PDO mapping	Yes
Value range	No
Default value	No

17.3. Object 0x2013: Binary interpreter output

This object is a binary interpreter object (refer to Chapter 13: Binary Interpreter Commands concerning the byte stream).

- Object description:

Index	0x2013
Name	Binary interpreter
Object code	RECORD
Data type	Binary interpreter command (object 0x43)
Category	Mandatory

- Entry description:

Access	Read only
PDO mapping	Yes
Value range	No
Default value	No

17.4. Object 0x2020: Home on block limit parameters

The object detailed description is presented in the *Gold Homing On Block* document and in the *Administrative Software Manual* chapter *PLC open Homing on Block*.

- Object description:

Index	**0x2020**
Name	Home on Block Limit Parameters
Object code	RECORD
Data type	Home On Block Limits, Object 0x45
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED8
Default value	5

Sub-index	**1**
Description	Torque limit - Maximum torque
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…50000
Default value	0

Sub-index	**2**
Description	Time Limit
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…1073741823
Default value	0

Sub-index	**3**
Description	Distance limit
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…2147483647
Default value	0

Sub-index	**4**
Description	Detection Velocity Limit
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	UNSIGNED32
Default value	0

Sub-index	**5**
Description	Detection Velocity Time Limit
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	UNSIGNED32
Default value	0

The value of the object **0x2020** sub index 1 can be read and written with the **OV[55]** command. The value of **0x2020**, sub index 1 is restricted by value 50000. When trying to set value larger than 50000, the SDO abort returns with error code 0x06090030.

The torque limit value set in **object 0x2020** sub index 1 is relative to the rated torque and presented in thousands of rated torque. When set to 1000, it means that the full rated torque is applied. Rated torque is defined by **object 0x6076**. Since the relation between torque and current is linear, the units are **considered as mA. Note the relation between 0x6076 and 0x6075.**

The value of the object **0x2020** sub index 2 can be read and write with **OV[56]** command. The value of **0x2020**, sub index 2 is non-negative. When trying to set a negative value, the SDO abort returns with error code 0x06090030. The time limit is defined in milliseconds.

The value of the object **0x2020** sub index 3 can be read and write with **OV[57]** command. The value of **0x2020**, sub index 3 is non-negative. When trying to set negative value, the SDO abort returns with error code 0x06090030. Distance limit is defined in user units.

The value of the object **0x2020** sub index 4 can be read and write with **OV[64]** command. Detection velocity limit is defined in user units.

The value of the object **0x2020** sub index 5 can be read and write with **OV[65]** command.

Time Limit (sub index 2) and Detection Velocity Time Limit (sub index 5) are presented in milliseconds, the max value to be entered is 1000000000 ms.

17.5. Object 0x2030: Recorder data

This object is used to retrieve recorder parameters according to **RC** and the sub-index field. If N is sub index, the 0x2030.N fetches the parameter, recorded in RC = (1 << (N-1)), the same as **BH=N** command.

- Object description:

Index	0x2030
Name	Bring recorded data
Object code	RECORD
Data type	Recorder Data
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	16
Default value	16

Sub-index	**1**
Description	Main speed
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**2**
Description	Main position
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**3**
Description	Position command
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**4**
Description	Digital input
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**5**
Description	Position error for UM=4, UM=5
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**6**
Description	Torque command
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**7**
Description	Bus voltage
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**8**
Description	Velocity Command
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**9**
Description	Field Angle
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**10**
Description	Active current
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**11**
Description	Reactive current
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**12**
Description	Analog input 1
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**13**
Description	Analog input 2
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**14**
Description	Current Phase A
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	R Refer to Table 17-1
Default value	

Sub-index	**15**
Description	Current Phase B
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

Sub-index	**16**
Description	Current Phase C
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	Refer to Table 17-1
Default value	

- **The bring data upload process:**
  The SDO Upload protocol is initiated as described in chapter 4. After confirmation, a character stream is transmitted. Each octet must be answered according to the Upload SDO segmented protocol, except sub-index 0, which returns object-supported entries in expedited SDO format.

The segmented response is built from the header and data stream.

Note: **The sub-index in this object is a value of the BH command, means upload object 0x2030 sub index 2 is the same as to send *BH=2* command. Refer to the Gold Drive Command Reference guide.**

The header byte sequence is as follows:

Byte Number	Description	Value	Type
0	Variable type for user. Field has no practical significance.	0: Integer 1: float	Byte
1	Data width: number of hex character of single transmitted data item.	4: Short integer 8: Long integer	Byte
2-3	Data length: actual number of transmitted data items in according to **RL** command.		Word
4-5	Record gap in accordance with **RG** command		Word
6-9	Floating factor to convert the data from internal units to physical units (not user units). Factors are used to convert the following from internal units: Current: (e.g., *Active Current*) to Amperes Velocity: (e.g., *Velocity Command*) to counts/sec Voltage: (e.g., *Bus Voltage*) to Volts The user must multiply the data by the factor to obtain the actual value.		Double word

Table 17-1: Upload SDO Header Byte

The rest of the byte sequence is the data stream. Sub-index 0 uploads the supported object entries.

If the recorder variables are changed during upload, the process will be aborted.

Example of uploading recorded data

Suppose px=7174 (0x1C06)

Set **RL=2;** to get two points only

Set **RC=66;** to record 2 signals: VBus ($2^{(7-1)}$) and main position ($2^{(2-1)}$)

Set **RR=2;** to start recorder

If RR==0. We can upload recorded data.

To upload main position via object 0x2030 sub index 2:

1. SDO client sends upload initiate SDO request

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x601	**0x40**	**0x30**	**0x20**	**0x02**	**0x00**	**0x00**	**0x00**	**0x00**

2. Server responses

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x581	**0x41**	**0x30**	**0x20**	**0x02**	**0x12**	**0x00**	**0x00**	**0x00**

Means: Segmented, there are 18 chars in string

3. SDO client sends

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**60**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

4. Server responses:

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**00**	**00**	**08**	**02**	**00**	**02**	**00**	**00**

Where (header starts from Byte 1):

byte 1 = 00, means “integer”;

byte 2= 08 means “Long integer”,

bytes 3,4 = 0002 are the word means recorded 2 points,

bytes 5,6 = 0002 are the word means record gap is 2, i.e. data recorded every 2 sampling times

byte 7 = 0 is the part of the next segmented data

5. SDO client sends

COBID/Byte	COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Value	**601**	**70**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

6. Server responses:

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**10**	**00**	**80**	**3F**	**06**	**1C**	**00**	**00**

Where:

Byte 6 of previous segment, byte 0,1,2 = 3F800000, means factor is float value which equals 1.0.

byte 3-6 = 00001C06 means position is 0x00001C06, which is the first recorded point

7. SDO client sends

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**60**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

8. Server responses:

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**07**	**06**	**1C**	**00**	**00**	**00**	**00**	**00**

Where:

byte 0-3 = 00001C06, means position is 0x00001C06, which is the second recorded point

07 – means “it is the last segmented data and 3 last bytes 00 00 00 do not contain data”

17.6. Object 0x2035: Upload Data Parameters

The object for ELMO internal use only, not for user

- Object description:

Index	0x2035
Name	Upload data parameters
Object code	RECORD
Data type	UNSIGNED32
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	
Default value	

17.7. Object 0x2036: Upload Data

The object for ELMO internal use only, not for user

- Object description:

Index	0x2036
Name	Upload data
Object code	RECORD
Data type	UNSIGNED64
Category	Optional

- Entry description:

Access	Read only
PDO mapping	No
Value range	
Default value	

17.8. Object 0x2041: Timestamp (free running timer)

This object transmits the accurate 32-bit timer of the drive. The timer has a 1-microsecond resolution that is updated once in every second sample time (default is every 100 microseconds). The accuracy of the report is 1 microsecond and the resolution is real time.

This object is typically used when the host wants to synchronize records for several slave nodes. It returns the precise time when the synchronous PDO data has been sampled (refer to Chapter 11).

- Object description:

Index	**0x2041**
Name	Drive free-running timer
Object code	VAR
Data type	UNSIGNED32
Category	

- Entry description:

Access	Read only
PDO mapping	Yes
Value range	No
Default value	No

17.9. Object 0x2045

This object is for ELMO internal use only, and not for user

The object contains the delay to be inserted between uploaded blocks. The accuracy of the delay is 1 microsecond and the resolution is HS cycle (250 us).

- Object description:

Index	**0x2045**
Name	Block upload Inhibit time parameter
Object code	VAR
Data type	UNSIGNED16
Category	optional

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	0…65535
Default value	0

17.10. Object 0x2051: Download Data

This object is for Elmo internal use only, not for user

- Object description:

Index	**2051h**
Name	Download data
Object code	RECORD
Data type	UNSIGNED64
Category	Optional

- Entry description:

Access	Write only
PDO mapping	No
Value range	
Default value	

17.11. Object 0x2060: Drive Parameters Checksum

This object includes the checksum of the parameters which are saved in the drive FLASH.

This checksum is calculated during save procedure (*SV*, object 0x1010 or save() function)

- Object description:

Index	0x2060
Name	Parameters Checksum
Object code	VAR
Data type	UINTEGER16
Category	Optional

- Entry description:

Access	Read only
PDO mapping	No
Value range	0-65535
Default value	None

In case of invalid parameters checksum, the object will return general abort message. (0x80000000)

In case checksum value is valid, the object will return the checksum value.

The checksum reflects the value of the parameters stored in the drive FLASH. Changing parameters on the fly will not change the checksum value.

17.12. Object 0x207B: Additional Position Range Limit

The object specifies additional position range limit, the same as **YM[1]** (0x207B.1) and **YM[2]** (0x207B.2) commands.

- The position of the external reference (object 0x20A0, PY) is counted cyclically, and therefore after the position is counted to its maximum value, the next position count will reset the position counter back to its minimum value. The speed reading is not affected by the position jump.
  For example: if 0x607B.1=-5 and 0x607B.2=5, the external reference is counted in a cycle length of 10. The external reference will always be in the range [-5…4].
  If the external reference rotates in the positive direction, the external reference count will proceed from 0, 1, 2, 3, 4 to -5, -4, -3, -2, -1, 0, 1.
- A new 0x607B setting is activated after the setting of 0x607B.2.
- 0x607B.2 must be bigger than 0x607B.1.
- If 0x607B.2 = 0x607B.1 = 0, the external reference modulo functionality is disabled. Actually, this means that additional position reference counts in the Integer32 counting range (−231 to (231 − 1)).
- If 0x607B.2 and 0x607B.1 are set so that 0x607B.2 > 0x607B.1, but object 0x20A0 is out of the range [0x607B.1…0x607B.2], 0x20A0 will be set to modulo range in accordance with:
  PY=(PY-YM[1])Mod(YM[2]-YM[1]) + YM[1]

Example (all messages in hexadecimal)

1. Check **YM[1]**

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**40**	**7B**	**20**	**01**	**00**	**00**	**00**	**00**
581	**43**	**7B**	**20**	**01**	**00**	**36**	**65**	**C4**

2. Check **YM[2]**

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**40**	**7B**	**20**	**02**	**00**	**00**	**00**	**00**
581	**43**	**7B**	**20**	**02**	**00**	**CA**	**9A**	**3B**

3. set **YM[1]**=-10

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**22**	**7B**	**20**	**01**	**F6**	**FF**	**FF**	**FF**
581	**60**	**7B**	**20**	**01**	**00**	**00**	**00**	**00**

4. set **YM[2]**=100

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**22**	**7B**	**20**	**02**	**64**	**00**	**00**	**00**
581	**60**	**7B**	**20**	**02**	**00**	**00**	**00**	**00**

5. set **PY**=100

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**22**	**A0**	**20**	**00**	**64**	**00**	**00**	**00**

6. Server returns abort message with error code 06090030 “Value range of parameter exceeded”.

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**80**	**A0**	**20**	**00**	**30**	**00**	**09**	**06**

7. set **PY**=99

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**22**	**A0**	**20**	**00**	**63**	**00**	**00**	**00**
581	**60**	**A0**	**20**	**00**	**00**	**00**	**00**	**00**

8. check **PY**

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**40**	**A0**	**20**	**00**	**00**	**00**	**00**	**00**

9. **PY**=99

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**43**	**A0**	**20**	**00**	**63**	**00**	**00**	**00**

10. set **YM[2]**= 90

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

601	**22**	**7B**	**20**	**02**	**5A**	**00**	**00**	**00**
581	**60**	**7B**	**20**	**02**	**00**	**00**	**00**	**00**

11. Check PY, PY will be converted into modulo range in accordance with formula
PY=(PY-YM[1])Mod(YM[2]-YM[1]) + YM[1] = (99-(-10))Mod(90-(-10)) + (-10) = -1

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**40**	**A0**	**20**	**00**	**00**	**00**	**00**	**00**
581	**43**	**A0**	**20**	**00**	**FF**	**FF**	**FF**	**FF**

- Object description:

Index	**0x207B**
Name	Additional Position Range Limit
Object code	ARRAY
Data type	INTEGER32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED8
Default value	2

Sub-index	**1**
Description	Min additional position range limit (YM[1])
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	-2^{31} to $(2^{31} - 1)$
Default value	-1000000000

Sub-index	**2**
Description	Max additional position range limit (YM[2])
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	-2^{31} to $(2^{31} - 1)$
Default value	1000000000

17.13. Object 0x2081: Extended Error Code

The object reflects Elmo **EE[N]** command. See Gold Line Command Reference Guide

- Object description:

Index	0x2081
Name	Extended Error Code
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED8
Default value	6

Sub-index	**1**
Description	Feedback error
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED32
Default value	0

Sub-index	**2**
Description	Profiler initialization error
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED32
Default value	0

Sub-index	**3**
Description	Download procedure error
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED32
Default value	0

Sub-index	**4**
Description	Elmo error code returned when SDO returns an abort message
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED32
Default value	0

Sub-index	**5**
Description	Motor Fault Reason
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED32
Default value	0

Sub-index	**6**
Description	ECAM initialization error
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	UNSIGNED32
Default value	0

17.14. Object 0x2082: CAN controller status

This object provides the status of the CAN controller status register. **OV[60]** is an alias of the object.

- Object description:

Index	**0x2082**
Name	CAN controller status
Object code	VAR
Data type	UNSIGNED32
Category	Optional

- Entry description:

Access	Read only
PDO mapping	Yes
Value range	UNSIGNED32
Default value	0

- Entry bit field placement is:

Bits	**31…25**	**24…16**	**15…8**	**7…0**
Description	Reserved	Network status	CAN transmit error counter	CAN receive error counter

The bit fields are described in following table:

Bit	**Meaning**	**Remarks**
0-7	CAN receive error counter (REC)	The Rx error counter is increased when the receive channel detects an error. When the counter reaches 96, the warning bit is set (bit 16). When the counter reaches 127, the drive is in Error Passive state (bit 17) and does not produce any more Error Frames.
8-15	CAN transmit error counter (TEC)	The Tx error counter is increased when the transmitter channel detects an error frame. When the counter reaches 96, the warning bit is set (bit 16). When the counter reaches 127, the drive is in Error Passive state (bit 17) and does not produce any more Error Frames. When the counter reaches 255, the drive enters Bus Off state

Bit	Meaning	Remarks
16	EW, Warning status	1 One of the two error counters (CANREC or CANTEC) has reached the warning level of 96. 0 Values of both error counters (CANREC and CANTEC) are less than 96.
17	EP, Error-passive state	1: The CAN module is in error-passive mode. CANTEC has reached 128. 0: The CAN module is in error-active mode.
18	BO, Bus-off status. The CAN module is in bus-off state.	1: There is an abnormal rate of errors on the CAN bus. This condition occurs when the transmit error counter (CANTEC) has reached the limit of 256. During Bus Off, no messages can be received or transmitted. The bus-off state can be exited by clearing the CCR bit in CANMC register or if the Auto Bus On (ABO) (CANMC.7) bit is set and after 128 * 11 receive bits have been received. After leaving Bus Off, the error counters are cleared. 0: Normal operation
19	ACKE, Acknowledge error.	1: The CAN module received no acknowledge. 0: All messages have been correctly acknowledged.
20	SE, Stuff error.	1: A stuff bit error occurred. 0: No stuff bit error occurred.
21	CRCE, CRC error.	1: The CAN module received a wrong CRC. 0: The CAN module never received a wrong CRC.
22	SA1, Stuck at dominant error	The SA1 bit is always at 1 after a hardware reset, a software reset, or a *Bus-Off* condition. This bit is cleared when a recessive bit is detected on the bus. 1: The CAN module never detected a recessive bit. 0: The CAN module detected a recessive bit.
23	BE, Bit error flag	1: The received bit does not match the transmitted bit outside of the arbitration field or during transmission of the arbitration field, a dominant bit was sent but a recessive bit was received. 0: No bit error detected.

Bit	Meaning	Remarks
24	FE, Form error flag	1: A form error occurred on the bus. This means that one or more of the fixed-form bit fields had the wrong level on the bus. 0: No form error detected; the CAN module was able to send and receive correctly.

Table 17-2: CAN Receiver Flag Bit 0…7

The following graph (Table 17-3) displays the method that the Frame error detection is counted:

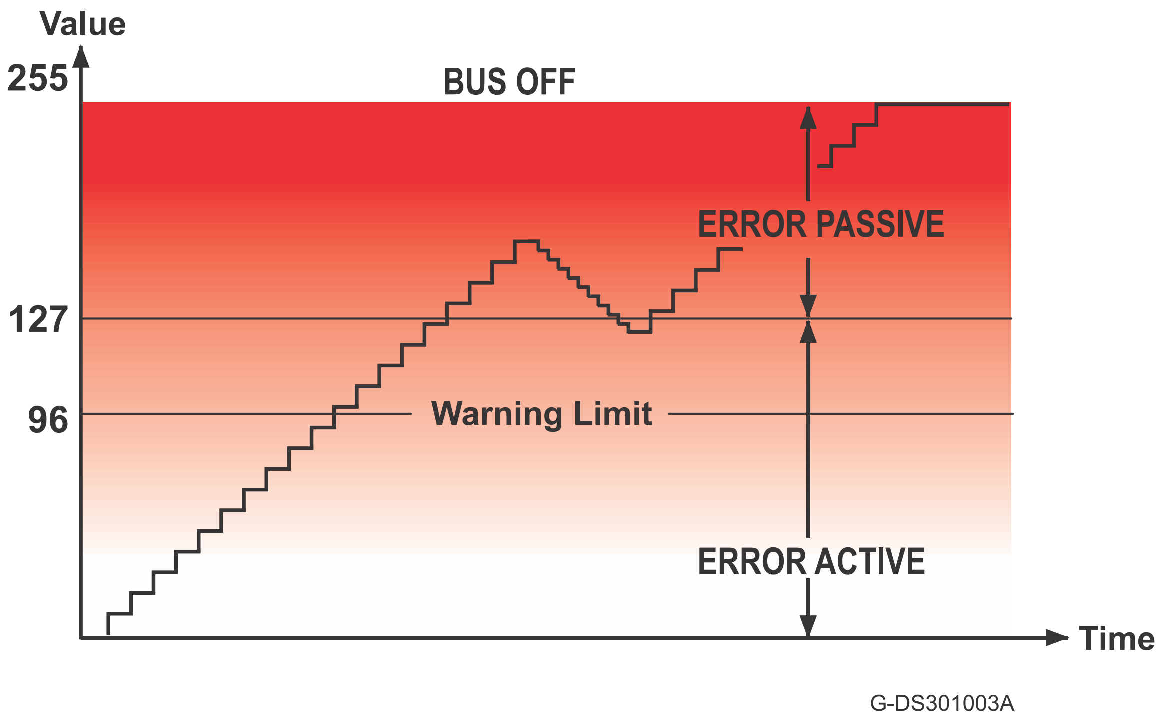


Table 17-3: Frame error

17.15. Object 0x2085: Extra status register

This object provides the extra status for various of signals according to the table below. **OV[61]** is an alias of the object.

- Object description:

Index	**0x2085**
Name	Extra status register
Object code	VAR
Data type	UNSIGNED16
Category	Optional

- Entry description:

Access	Read only
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No

- The bit-field object (Bits 0, 1 are also reported by **WS[16]**) reports as shown in the following table.

Bits	**Description**
0	For a planar motor, this bit informs that the commutation has performed successfully. 0: Commutation is not known. Next motor enable will perform commutation sequence on Y axis according to **CA[89]** and **CA[90]**. Yaw error in X is reset to 0. 1: Commutation known.
1	Informs that the levels of the analog signals used for feedback (sine/cosine) are above the minimum defined range. The minimum range is selected via the **CA[52]** command. If the range is too low, the bit will be set to 0. **Note** that in this case, bit 0 (Planar commutation status) will also be reset to 0. Signal levels, in this indication, are **only** checked **during motor disable** (MO=0). **Note: CA[48]** and **CA[49]** are used to monitor the feedback amplitude when the motor is enabled. 0: Analog signals amplitude is lower than the defined in **CA[52]** 1: Analog signals amplitude is above or equals to the requested.
2	Low voltage sensed, The bit is set if the voltage drops below the value set in **XT[1]** or object **0x2F45** sub index 1.

Bits	Description
3	High Voltage sensed. The bit will be set if the voltage is higher than the value set in **XT[2]** or object **0x2F45** sub index 2.
4	Analog sensor low voltage. The bit is set if the voltage of any sine \ cosine analog sensor drops below the level set in **XT[3]** or object **0x2F45** sub index 3.
5	High temperature sensed. The bit is set If the temperature of the drive (**TI[1]**) is higher than the temperature that was set in **XT[4]** or object **0x2F45** sub index 4
6	Battery alarm. Bit is set if there is a battery alarm (low voltage or no battery) in the digital absolute sensor. The battery alarm indication, as comes from the sensor, is reflected in this bit.
7	Yaskawa absolute serial encoder was reset
8	Auto focus error bit. The Auto Focus is Elmo's proprietary additional sensor that can be read via the drive inputs.
9	Gurley absolute sensor data valid
10	Auto focus mode out of limit. The Auto Focus is Elmo's proprietary additional sensor that can be read via the drive inputs.

17.16. Object 0x2086 STO status register

The STO status register includes the state machine data at the time the STO error is detected.

OV[62] is an alias of the Object.

- Object description:

Index	**0x2086**
Name	STO status register
Object code	VAR
Data type	UNSIGNED32
Category	Optional

- Entry description:

Access	Read only
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0

The register includes the following formats:

Bit	**Description**
0-4	The present state number when an STO error is detected. Refer to the *State* column in Table 17-5.
5-6	0 – STO full diagnostics is in progress. 1 – STO full diagnostics passed, and STO periodic diagnostics is in progress. 2 - Error is detected.
7	Unused
8-12	Error code, see Table 17-5 below
13-15	Unused
16-20	Current STO diagnostics state. Refer to the *State* column in Table 17-5.
21-31	Unused

Table 17-4 Object 0x2086 (OV[62]) STO Status Register

The following table describes the address space of the STO diagnostic function in the CPLD.

Error Value	Description
0	No error
1	Error in state 0, 'DAIG_STOx_P1' or PWMx status != 0
2	Error, short between 'STO1_EN' and STOx IN
3	Error, short between 'STO2_EN' and STOx IN
4	Error in state 1, at least one of PWMx are active
5	Error in state 1, Incorrect resistor R702/R802 value
6	Error in state 2, short between 'STOx_EN' and 'DAIG_STOx_P1'
7	Error in state 3, PWMx test (toggle) failed
8	Error in state 4, STO minimum filter time
9	Error in state 4, STO filter time is too long
10	Error in state 4, 'STOx_EN' is not active but one or more of 'DAIG_STOx_P1', 'DAIG_STOx_P2',PWMx are active
11	Error in state 6, PWM Letch is active
12	Error in state 6, 'DAIG_STOx_P2' is still active after fall time
13	Error in state 5, periodic test, short between STOx_EN and STOx_P1
14	Error in state 5, periodic test, PWM active while MO==0
15	Error in state 5, periodic test, 'DAIG_STOx_P2' is disconnected
16	Error in state 5, periodic test did not execute for more than 1[sec]

Table 17-5 STO Error Codes

17.17. Object 0x2087: PAL Version

This object indicates the burned PAL version.

- Object description:

Index	0x2087
Name	PAL Version
Object code	VAR
Data type	UINTEGER16
Category	Optional

- Entry description:

Access	Read only
PDO mapping	No
Value range	0-255
Default value	0

Value	Description
20	GCON Rev-A PAL version
40-69	GCON Rev-C PAL version
70-127	GCON Rev-E PAL version
128-255	Reserved for future usage
All others	Error, one of the following reasons: • PAL not burn • Incompatible PAL

17.18. Object 0x2090: Firmware download

This object is for Elmo internal use only, not for user

- Object description:

Index	0x2090
Name	Firmware download
Object code	
Data type	
Category	Optional

- Entry description:

Access	Write only
PDO mapping	No
Value range	
Default value	

17.19. Object 0x20A0: Auxiliary position actual value

This object returns the actual position of the auxiliary axis (PY). **OV[53]** is an alias of the object.

- Object description:

Index	0x20A0
Name	Auxiliary position
Object code	VAR
Data type	SIGNED32
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	Yes
Value range	-2^{31} to $(2^{31} - 1)$
Default value	0

17.20. Object 0x20B0: Socket Additional Function

Object 0x20B0 controls the functionality of the sockets.

This allows the CANopen master to control the socket functionality using trivial object via SDO. Sub index 8 can be accessed by the **OV[54]** command.

- Object description:

Index	**0x20B0**
Name	Socket Additional Function
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0…255
Default value	9

Sub-index	**1**
Description	Socket used for Position Loop, **CA[45]**
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	1…4
Default value	1

Sub-index	**2**
Description	Socket used for Velocity Loop, **CA[46]**
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	1…4
Default value	1

Sub-index	**3**
Description	Socket used for Commutation, **CA[47]**
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	1…4
Default value	1

Sub-index	**4**
Description	Socket used for Position reference, **CA[68]**
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…4
Default value	0

Sub-index	**5**
Description	Socket used for Velocity reference, **CA[69]**
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…4
Default value	0

Sub-index	**6**
Description	Socket used for Current reference, **CA[70]**
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…4
Default value	0

Sub-index	**7**
Description	Socket is used for Touch probe capturing, **CA[87]**
Entry category	
Access	Read/Write
PDO mapping	No
Value range	0…4
Default value	0

Sub-index	**8**
Description	Socket is used for Homing reference, **OV[54]**
Entry category	
Access	Read/Write
PDO mapping	No
Value range	0, 1
Default value	0

Sub-index	**9**
Description	Socket used for Additional Sensor 0x20A0 read out, **CA[79]**
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…4
Default value	1

17.21. Object 0x20FC: Absolute Sensor Functions

The object performs the following operations on absolute sensors:

- Resets Panasonic, Tamagawa or EnDAT2.2 absolute multi turn position
- Resets EnDat 2.2 warning and error.
- Object description:

Index	0x20FC
Name	Absolute Sensor Function
Object code	ARRAY
Data type	UNSIGNED16
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0……255
Default value	2

Sub-index	**1**
Description	Resets Panasonic, Tamagawa, EnDAT2.2 absolute multi turn position. The same as **TW[19]** command
Entry category	Mandatory
Access	Write only
PDO mapping	No
Value range	0………65535
Default value	

Sub-index	**2**
Description	Resets EnDat 2.2 warning and error. The same as **TW[20]** command
Entry category	Mandatory
Access	Write only
PDO mapping	No
Value range	0………65535
Default value	

17.22. Object 0x20FD: Digital Inputs

- Object description:

Index	0x20FD
Name	Digital Inputs
Object code	VAR
Data type	UNSIGNED32
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	Bitfield, range is not applicable
Default value	No

The object indicates the digital inputs status on read access. Bit placement is presented in the tab below (1: means logical active state, 0: logical not active):

31…16	**15…4**	**3**	**2**	**1**	**0**
Logical state of digital inputs 1 to 16	Reserve, 0	Interlock	Main Home Switch	FLS	RLS

Interlock bit is set to 1 when one (or both) of the STO inputs is disabled and the drive is in safety state. (from firmware version 1.1.10.7 B00)

The object is also indicated by **OV[31]** command.

If any digital input was defined as sticky general purpose digital input (see **IL[]** command description in Gold Line Command Reference Guide) writing '1' to this input clears the bit (the same **OV[31]** command).

The digital inputs can also be read via **IP** command.

Object is alias to 0x60FD with optionally write access.

Example

1. Set IL[4]=263 (GPI, sticky)
2. Turn on and then off input switch 4
3. To check digital inputs SDO client sends:

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**40**	**FD**	**20**	**00**	**00**	**00**	**00**	**00**

4. Server responds:

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**43**	**FD**	**20**	**00**	**00**	**00**	**C8**	**17**

Where: Bit 19 =1 (bit 4 in High 16 bit word), means sticky GPI 4 is active

5. To Clear sticky bit 4 SDO server sends

COBID	Byte 0	Byte1	Byte2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**22**	**FD**	**20**	**00**	**00**	**00**	**08**	**00**

6. Server responds: 60 FD 20 00 00 00 00 00

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**60**	**FD**	**20**	**00**	**00**	**00**	**00**	**00**

7. To check Digital Inputs SDO server sends

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	**40**	**FD**	**20**	**00**	**00**	**00**	**00**	**00**

8. Server responds

COBID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
581	**43**	**FD**	**20**	**00**	**00**	**00**	**C0**	**17**

Where: Bit 19 =0 (bit 4 in High 16 bit word), means sticky GPI 4 is cleared

17.23. Object 0x2201: Digital input low byte

This object defines least 8 bits of object 0x60FD, simple digital inputs for drives.

- Object description:

Index	0x2201
Name	Digital inputs low byte
Object code	VAR
Data type	UNSIGNED8
Category	Optional

- Entry description:

Access	Read only
PDO mapping	Yes
Value range	0…0xFF
Default value	No

- Bit placement (1: means logical active state, 0: logical not active):

MSB				LSB
7 4	3	2	1	0
Reserved=0	Interlock	Main Home Switch	FLS	RLS

From firmware version 1.1.10.7 B00 the Interlock is set to '1' when at least one of the STO inputs is disabled and the drive is in safety state.

17.24. Object 0x2202: Extended input

The object presents extended inputs similar to **XI[n]** command.

- Object description:

Index	**0x2202**
Name	Extended input
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	3
Default value	3

Sub-index	**1**
Description	Extended input value bits 0…31
Entry category	Mandatory
Access	Read/Write (Write access has no effect, but does not return error)
PDO mapping	TxMap
Value range	Bitfield, range is not applicable
Default value	No

Sub-index	**2**
Description	Logic/Polarity
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	Bitfield, range is not applicable
Default value	0

Sub-index	**3**
Description	PDO Event Mask
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	Bitfield, range is not applicable
Default value	0

17.25. Object 0x2203: Application Object

This read-only object is design for specific application needs.

For EtherCAT usage, the object is mapped to tPDO: **0x1A24**.

- Object description:

Index	**0x2203**
Name	Application Object
Object code	VAR
Data type	UNSIGNED32
Category	Optional

- Entry description:

Access	Read only
PDO mapping	Yes
Value range	0…(2^{32})-1
Default value	No

The **Object 0x2203** returns values according to the setting of **Object 0x2F41** bit 16..19 as follows:

Bit 16..19 of 0x2F41		
0 (default)	Depends on drive hardware	In Duet drives if hardware is supported, **Object 0x2203** returns the 16 bits external A2D value: (16 bits Channel A << 16 | 16 bits Channel B) In all other drives, returns the Drive temperature (0x22A3.1)
1	Analog encoder amplitude	The sum of the $\sin^2 + \cos^2$ of analog feedback. Values are in A2D internal level. Typically for 1Vp-p sensor: ~2,380,000
2	Analog encoder signals	The cosine signal << 16 | sine signal.
3	Drive temperature	Similar to object 0x22A3.1
4	Analog input 2	Similar to object 0x2205.2

The object can be retrieved by **WS[38]** command.

17.26. Object 0x2205: Analog Input Object

This object returns the value of the analog input 1 in mVolts, and analog input 2 in A2D ticks.

- Object description:

Index	0x2205
Name	Analog Input Object
Object code	ARRAY
Data type	INTEGER16
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of entries
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	2
Default value	2

Sub-index	**1**
Description	Analog Input 1
Entry category	Optional
Access	Read only
PDO mapping	Yes
Value range	+/- 10,000 mVolts
Default value	-

Sub-index	**2**
Description	Analog Input 2
Entry category	Optional
Access	Read only
PDO mapping	CAN: Yes ECAT: No
Value range	0 – 4095 A2D ticks
Default value	-

Notes:

Sub index 1 retrieves the analog input 1 value in millivolts in the range of +/-10,000.
Sub index 2 retrieves the analog input 2 value in A2D ticks: 0 - 4095

Example:

Host sends SDO request for analog input 1:

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Hex value	40	05	22	01	00	00	00	00

Assume that the drive answer an SDO expedite message with the following value:

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Hex value	42	05	22	01	BD	FF	00	00

Note: **Byte 4 & byte 5 are 16 bits, representing the internal value of analog input 2.**

Then the reading obtained is:

The value of analog input 1 = 0xFFBD (INTEGER 16) = -67mVolts

AN[1] commands returns the same value in volts

17.27. Object 0x2206: 5V DC supply

The object returns the value of 5V DC supply in mVolts.

- Object description:

Index	0x2206
Name	5 V DC Supply
Object code	VAR
Data type	INTEGER16
Category	Optional

- Entry description:

Access	Read only
PDO mapping	Yes
Value range	INTEGER16
Default value	No

17.28. Object 0x22A0: Digital Outputs

he object is used to set general purpose digital output via CANopen interface. The function of the digital output is set via **OL[]** command. The object affects the General purpose function.

- Object description:

Index	0x22A0
Name	Digital Outputs
Object code	VAR
Data type	UNSIGNED8
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	Yes
Value range	0……255
Default value	0

17.29. Object 0x22A1: Extended Outputs

The object presents extended outputs similar to **XO[n]** command. The Extended Outputs are presented in special hardware configuration only.

- Object description:

Index	0x22A1
Name	Extended Outputs
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0……255
Default value	3

Sub-index	**1**
Description	Extended output value bits 0…31
Entry category	Mandatory
Access	Read/Write
PDO mapping	RxMap
Value range	0…(2^{32})-1
Default value	0

Note: **Write access to this object is allowed but has no affect.**

Sub-index	**2**
Description	Logic/Polarity
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0xFFFFFFFF

Sub-index	**3**
Description	Extended Output Mask
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0xFFFFFFFF

Extended output bit (sub index 1) cannot be set if relevant bit in extended output mask is 0.

17.30. Object 0x22A2: Drive Temperature in ˚C

The object returns the value of drive temperature in degree of Celsius.

- Object description:

Index	**0x22A2**
Name	Drive Temperature in C
Object code	VAR
Data type	UNSIGNED16
Category	Optional

- Entry description:

Access	Read only
PDO mapping	TxMap
Value range	0……65535
Default value	0

17.31. Object 0x22A3: Temperature Array

The object presents temperature array similar to **TI[n]** command.

- Object description:

Index	**0x22A3**
Name	Temperature Array
Object code	ARRAY
Data type	UNSIGNED16
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of supported elements
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0……255
Default value	3

Sub-index	**1**
Description	Drive temperature in Celsius
Entry category	Mandatory
Access	Read only
PDO mapping	TxMap (CAN), No (ECAT)
Value range	0……..65535
Default value	No

Sub-index	**2**
Description	Drive temperature in Fahrenheit
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0……..65535
Default value	No

Sub-index	**3**
Description	Absolute encoder temperature in Celsius
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	0……..65535
Default value	No

Note: **The sensor temperature is relevant only if the sensor supports it.**

17.32. Object 0x2E00: Gain Scheduling Manual Index

This object allows manually selection of the gain scheduling index, and permits selection of two types of scheduling in the same object. The object's 8 low bits (LSB) are used for one of the selected gain scheduling tables while the high bits (MSB) are used to select an index from another gain scheduling table.

By setting the relevant **GS[]**, according to the desired scheduling, to either 67 or 68, the Low or the High bytes are used respectively. The optional **GS[]** can be:

- **GS[2]** for control loop (**KI[2]**,**KP[2]**, and **KP[3]**) scheduling
- **GS[16]** and **GS[17]** for speed advance filter 1 and filter 2 scheduling
- **GS[18]** for position advanced filter scheduling
- Object description:

Index	**0x2E00**
Name	Gain Scheduling Manual Index
Object code	VAR
Data type	UNSIGNED16
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	RxMap
Value range	0……..65535
Default value	0

17.33. Object 0x2E06: Torque Window

This object indicates the configured torque window. The value is given in 1/1000 of rated torque. **TR[5]** is an alias of the object **0x2E06**. The object is also accessible via **OF[50]**.

- Object description:

Index	0x2E06
Name	Torque Window
Object code	VAR
Data type	UNSIGNED16
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	0……..65535
Default value	40

When the actual torque reaches the Torque Window (object **0x2E06**) for Torque Window time (object **0x2E07**), the Target Reached bit (object **0x6040** bit 10) will be set. The **MS** command will be set to 0.

17.34. Object 0x2E07: Torque Window Time

This object shall indicate the configured torque window time. The value shall be given in milliseconds. **TR[6]** is an alias of the object **0x2E07**. The object is also accessible via **OF[51]**.

- Object description:

Index	0x2E07
Name	Torque Window
Object code	VAR
Data type	UNSIGNED16
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	0……..65535
Default value	20

When the actual torque reaches the Torque Window (object **0x2E06**) for Torque Window time (object **0x2E07**), the Target Reached bit (object **0x6040** bit 10) will be set. The **MS** command will be set to 0.

17.35. Object 0x2E10: Home on Touch Probe

Adjust the position of the touch probe socket according to the touch probe captured value and the homing offset (object **0x607C**). In other words, treat the touch probe captured value as home.

Position adjustment completion is indicated by returning of the SDO. This object is not mappable.

The object includes the following:

Value	Details
0	Adjust the position of the touch probe socket according to the touch probe rising edge value and the homing offset (object **0x607C**)
1	adjust the position of the touch probe socket according to the touch probe falling edge value and the homing offset (object **0x607C**)

- Object description:

Index	0x2E10
Name	Home on touch probe
Object code	Variable
Data type	Integer32
Category	Optional

- Entry description:

Attribute	Value
Sub-index	0
Access	Write-only
PDO mapping	NA
Value range	Integer16
Default value	Not defined

17.36. Object 0x2E15: Gantry YAW offset

The object performs write and read access to Gantry yaw offset in encoder counts (**FP[1]**-**FP[3]**).

The object is reflected in the **TW[14]** command.

- Object description:

Index	**0x2E15**
Name	Gantry yaw offset
Object code	VAR
Data type	INTEGER16
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	0……..65535
Default value	0

17.37. Object 0x2F00: User Integer

This object provides an array of 24 integer numbers for general-purpose use similar to **UI[N]** command.

- Object description:

Index	0x2F00
Name	User Integer array
Object code	ARRAY
Data type	Integer32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of entries
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	24
Default value	24

Sub-index	**1-24**
Description	User Array
Entry category	Optional
Access	Read/Write
PDO mapping	Yes
Value range	0…(2^{32})-1
Default value	0

17.38. Object 0x2F01: User Float Array

This object provides an array of 24 floating numbers for general-purpose use similar to the **UF[N]** command.

- Object description:

Index	0x2F01
Name	User Float Array
Object code	ARRAY
Data type	Floating Point (Float)
Category	Optional

- Entry description:

Sub-index	0
Description	Number of entries
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	24
Default value	24

Sub-index	1-24
Description	User Array
Entry category	Optional
Access	Read/Write
PDO mapping	Yes
Value range	0…(2^{32})-1
Default value	0

17.39. Object 0x2F05: Get drive control board type

This object retrieves HW drive control board type similar to **WS[8]** command The possible values are presented in the following table.

HW type	Value
Score	0
GCON Rev. A	1
GCON Rev. C	2
GCON Rev. E	3

- Object description:

Index	0x2F05
Name	Get drive control board type
Object code	VAR
Data type	UNSIGNED16
Category	Optional

- Entry description:

Access	Read only
PDO mapping	No
Value range	0……65535
Default value	No

17.40. Object 0x2F20: TPDO asynchronous events

This object is used to select the events that cause asynchronous TPDOs to be transmitted, with transmission type 254, 255. PDOs with other transmission types are ignored.

Sub-indices 1 to 4 define events for transmitting TPDO1, TPDO2, TPDO3 and TPDO4, respectively. The event definition for the PDO transmission is a bit field, as follows:

Bit	Event
0	Motion complete: **MS = 0**
1	Main homing complete: **HM[1] = 0**
2	Auxiliary homing complete: **HY[1] = 0**
3	Motor shut down by exception: **MO = 0**
4	Motor started: **MO = 1**
5	User program *emit* command
6	OS interpreter execution complete
7	Reserve
8	Motion Started event
9	External Digital Input Event (for internal use only, not for user)
10…23	Reserved
24	PDO data changed
25	Timer event (for internal use only, not for user)
26	Digital input event in according to **0x60FD**
27	Status word event
30	Reserve
31	Binary interpreter command processing complete

Asynchronous TPDOs obey the inhibit time restrictions, as explained in section 16.20 Objects 0x1800 - 0x1803:
Transmit PDO communication parameter.

- Object description:

Index	**0x2F20**
Name	PDO events
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of sub-indices
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	4
Default value	4

Sub-index	**1**
Description	Events for PDO1 trigger
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x08000000

Sub-index	**2**
Description	Events for PDO2 trigger
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0x80000000

Sub-index	**3**
Description	Events for PDO3 trigger
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0

Sub-index	**4**
Description	Events for PDO4 trigger
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0

17.41. Object 0x2F21: Emergency events

This object selects events as the cause for transmitting emergency objects (see Chapter 6).

Updating the object **0x2F21** resets the object **0x1003** sub index 0 and object **0x603F** sub index 0.

The driving event definition for an emergency is a bit field, as follows:

Bit	Event/EMCYname	EMCY Error Code (Hex)	Details and Resolution
0	CAN message lost	8110	HW or SW buffer is overflowed. Check the rate of the messages for Sync, NMT or RPDO.
2	Attempt to access non configured RPDO	8210	RPDO received is not mapped or contains incorrect number of bytes. Check the RPDO mapping and the RPDO sent by the host. Check if number of RPDO bytes equals to the bytes that were mapped
3	Heartbeat event	8130	Consumer heartbeat time elapsed. Check if producer message was sent in time. Check the consumer time in object 0x1016.
7	Motor shut down by fault		See Table 17-7 Motor Faults and EMCY
8	RPDO failed	6300	Object mapped to an RPDO returned an error during interpretation. If the RPDO is used as a motion set-point and the motion failed to perform. Check if data of RPDO is correct with respect to the mapped object. Check the reason for the failure as indicated in “Elmo Error Code” (byte 3 aliases to the EC command). Optional reasons may be: parameter is out of range, operating mode is not supported, error mapping in a progress, motor failed to start, etc.
9	DS 402 IP Emergency	FF02	IP mode failure: • Underflow: Interpolated periods defined via object 0x2F75 is elapsed • Interpolation queue full (overflow): the new set point is received before previous set point was fully processed. • Bad sub-index • Wrong set-points order in case of IP sub mode is -1. In case of underflow: • Check that a new set-point (object 0x60C1) arrived prior to the time defined by 0x2F75.

Bit	Event/EMCYname	EMCY Error Code (Hex)	Details and Resolution
			• Check master Sync or RPDO message rate: it might be lower than the period time (0x60C2). In case of overflow: • Check master Sync or RPDO message rate: it might be higher than the period time (0x60C2). In case of bad index: • Check that the mapped set-point object (sub index of 0x60C1) is relevant to the IP sub-mode. In case of wrong set-point order: • Check that the order of set points 0x60C1.1 and 0x60C1.2 are according to the IP sub-mode definition.
10	Recovered from bus off	8140	EMCY is sent after the drive recovered from bus-off state • Check the physical connection of the CAN • Check that the baud rate of all nodes is the same as the master’s • Check the CAN 120 Ω terminators
11	Cannot enable motor		An attempted to start the motor via Binary Interpreter or DS-402 control word (0x6040) failed. See Table 17-8 Amplifier Status for indication and possible resolution.

Table 17-6 Emergency Events

- Object description:

Index	0x2F21
Name	Emergency events
Object code	VAR
Data type	UNSIGNED16
Category	

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	No
Default value	0xFFFF (all emergencies enabled)

The following table (Table 17-7) details the bit-field structure with respect to the fault reason (To be updated after change in **MF** command description).

MF Value (Hex)	Event/EMCY name	EMCY error code(Hex)	Details and Resolution
1 (0x1)	Main feedback error	7300	Feedback error: Resolver feedback is not ready – Resolver angle was not located yet. Analog encoder or Resolver feedback is either lost or with too low amplitude. Battery Alarm: Absolute Position may be incorrect due to battery power loss. For analog feedbacks check the threshold level in **CA [48]** and **CA[49]** For an absolute encoder check the reason in **EE[1]**. The fault causes a commutation search on the next motor enable. Check the feedback settings
2 (0x2)	Commutation process fail during motor on	7382	Commutation process fail during motor on for the reasons: • For locking the phase • Planar motor when on alignment process Fault reaction can be selected via 0x605E (**OF[6]**) Check settings of encoder type and encoder resolution. Try running the expert tuner commutation process again.

MF Value (Hex)	Event/EMCY name	EMCY error code(Hex)	Details and Resolution
8 (0x8)	Current exceeded peak limit	8311	The peak current has exceeded the value of **MC** but has not yet reached the level of a short. This is typically caused by instability of the current loop. Check peak current, **KP[1]**, **KI[1]** settings. Try to run current loop tuning again
16 (0x10)	Motor disabled by switch	5441	Motor disabled by: • INHIBIT or ABORT, and • FLS and RLS are switched on simultaneously in IP or CSP operation modes. Note that FLS, RLS are ignored when **XA[4]**=4. The function of Abort\Inhibit\FLS\RLS is defined via **IL[]** command. Refer to the Command reference.
32 (0x20)	AC fail, loss of phase	3130	Activated in specific HW version of the drive. Refer to the drive User manual. Check hardware configuration dependent on drive. Check extended digital input **XI** command. Check that the motor phases are connected properly.
64 (0x40)	Halls sensor speed is too high	7381	Two digital Hall sensors were changed at the same time. Error occurs because digital Hall sensors must be changed one at a time. Digital halls run too fast or disconnected. Check hall settings and hall connections. Try to run the commutation wizard again.

MF Value (Hex)	Event/EMCY name	EMCY error code(Hex)	Details and Resolution
128 (0x80)	Speed tracking error	8480	The difference between the commanded speed to the control loop and the feedback exceeded the value defined in **ER[2]**. Speed tracking error = (**DV[2]** – **VX)** (for **UM=2** or **UM=4**, **5**) This may occur due to: • Bad tuning of the speed controller • Too tight a speed error tolerance • Inability of motor to accelerate to the required speed due to too low a line voltage or not a powerful enough motor Resolution: • Check that the value of **ER[2]** is appropriate to your sensor and profile • Try to run tuning again, and consider using feed forward • Check that you use optimal commutation • Lower the acceleration or speed of your motion profile
256 (0x100)	Position tracking error	8611	Position tracking error **DV[3**] - **PX** (**UM=5**) or **DV[3]** - **PY** (**UM=4**) exceeded position error limit **ER[3]**. This may occur due to: • Bad tuning of the position or speed controller • Too tight a position error tolerance • Abnormal motor load, or reaching a mechanical limit Resolution: • Check that the value of **ER[3]** is appropriate to your sensor and profile • Try to run tuning again • Check that you use optimal commutation • Lower the acceleration or speed of your motion profile

MF Value (Hex)	Event/EMCY name	EMCY error code(Hex)	Details and Resolution
1024 (0x400)	Gantry position error	5280	Gantry yaw or stepper closed loop position error. Resolution: • Check that the value of **ER[5]** is appropriate to your system and profile • Try to run tuning again • In case of gantry, check that you use optimal commutation for both master and slave • Lower the acceleration or speed of your motion profile
2048 (0x800)	Heartbeat event	8130	Consumer heartbeat time is elapsed. The motor was shut down due to a heartbeat event according to CANopen DS-301 object **0x1016**. Check settings of consumer heartbeat time. Check if producer message was sent in time. **For CAN users:** The motor was shut down due to a heartbeat event or Bus Off state according to CANopen DS301 object 0x1016. **For EtherCAT users:** The motor was shut down due to a loss of frame or loss of synchronization to the EtherCAT master.
4096 to 61440 (0x1000 to 0xF000)	Amplifier problem		Indicates the problem that the power section of the drive has encountered. (See Table 17-8 "Amplifier Status bits indication")
61440 (0xF000)	Additional abort	5442	Motor disabled by switch "additional abort motion" See **IL[]** for more details about the Inhibit/Abort functions. The External Abort input should be reset.
131072 (0x20000)	Over speed	8481	Over speed indication . (Compatibility only) The motor speed has exceeded the value which is defined in **HL[2]** or **LL[2]**. **VX**<**LL[2]** or **VX**>**HL[2]**The motor main speed is reported in **VX**. Perform tuning. To cancel this protection, set **HL[2]**=0. **HL[2]** is in user units

MF Value (Hex)	Event/EMCY name	EMCY error code(Hex)	Details and Resolution
2097152 (0x200000)	Motor is stuck	7121	Motor stuck - the motor is powered but is not moving according to the definition of **CL[2]** and **CL[3].** A stuck motor indication can be requested by using **CL[2]**, **CL[3]** and **CL[4]** according to the following logic: If the motor speed is lower than **CL[2]** (in counts/sec) and the measured current is higher than **CL[3]** (in amperes), and if this is observed for more than **CL[4]** msec, the motor is considered to be in the *Motor Stuck* state. Resolution: • Check **CL[2]**, **CL[3]**, **CL[4]** settings. • Check for physical obstructions • Check sensor wiring
4194304 (0x400000)	Feedback is out of position limits	8680	Position limit exceeded: **PX**<**LL[3]** or **PX**>**HL[3]** (**UM=5**), or **PY**<**LL[3]** or **PY**>**HL[3]** (**UM=4**). (Compatibility only) The main feedback is reported in **PX**. Check **HL[3]**, **LL[3]** Motor position exceeded feedback's limits. To cancel this protection set **HL[3]**=**LL[3]=**0. These parameters are in user units
16777216 (0x1000000)	Gantry slave disabled	FF40	Gantry master disables the slave because gantry slave is not enabled at current mode. Check the slave`s operating mode. Re-enable the gantry slave in Current Mode.

MF Value (Hex)	Event/EMCY name	EMCY error code(Hex)	Details and Resolution
536870912 (0x20000000)	Failed to start motor	FF10	Commutation auto-phasing failed, and the motor could not be started. A request to initiate the motor using a CANopen Controlword failed. Possible problems may be: • Inhibit/abort switches are active. • Commutation auto-phasing failed. • The PAL is not initiated/burned. • Too little time has passed since the last fault (typically 7.5 msec) or the last motor disable. • Profiler initiation failed due to conflicts between one of the profiler parameter/objects (reason in **EE[2]**). Check the reason in **EC** command

Table 17-7 Motor Faults and EMCY

The following table (Table 17-8) details the Amplifier Status bits indication.

MF Value (Hex)	Event/Emergency name	EMCY error code(Hex)	Details and resolution
0	All OK		All OK
12288 (0x3000)	Undervoltage	3120	Power supply is shut down or it has too high output impedance. The amplifier is not measuring the minimum required voltage. Check minimum allowed value in **WI[37]** (burnt) and **WI[38]** (actual) command. Check if personality is suitable to HW. Check bus voltage connections (**VL**) or bus voltage reading **AN[6]**
20480 (0x5000)	Overvoltage	3310	Over-voltage: power-supply voltage is too high or servo drive could not absorb kinetic energy while braking a load. A shunt resistor may be required. The amplifier is measuring a voltage which is higher than the allowed threshold. Check maximum allowed voltage in **WI[35]** (burnt) and **WI[36]** (actual) command. Check if personality is suitable to HW. Check bus voltage connections (**VL**) or bus voltage reading **AN[6]**

MF Value (Hex)	Event/Emergency name	EMCY error code(Hex)	Details and resolution
28672 (0x7000)	Safety	FF20	One or two of the safety inputs are in safety state. Check safety indications that are reported in **SR** bits 14 and 15. Check safety inputs. Turn on the safety inputs
45056 (0xB000)	Short Protection	2340	Motor or its wiring may be defective, or drive is faulty. The current has exceeded a range which is considered as a phase-to-phase or phase-to-ground short. Check HW. Check if personality is suitable to HW. Check current controller tuning, of feedback noise in case of analog sensor.
53248 (0xD000)	Over-temperature	4310	Drive overheating. The environment is too hot or heat removal is not efficient. Could be due to large thermal resistance between drive and its mounting. The drive is sensing a temperature which exceeds the maximum allowed temperature limit. Check actual temperature in **TI[1]** (in Celsius), **TI[2]** in Fahrenheit. Check if personality is suitable to HW Increase **TS** or decrease **XP[2]**
61440 (0xF000)	Additional Abort	5442	The drive sensed an input switch that is defined as Additional abort (refer to **IL[]** command). External Abort input should be reset.

Table 17-8 Amplifier Status

17.42. Object 0x2F41: DS402 Configuration object

This bit field object provides several configuration options to the DS402 protocol.

It resets to 0 after boot reset and must be set again in such cases.

- Object description:

Index	0x2F41
Name	DS402 Configuration object
Object code	VAR
Data type	UNSIGNED32
Category	

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	No
Default value	0

- Data description:

Bits	Description	
0..1	Reserved	
2	**0:**	Perform commutation once after power on
	1:	Perform commutation every motor enable
3..15	Reserved	
16…19		Defines the data to **Object 0x2203**:
	0:	If 16 bit A2D supported, points to A2D data (see **Object 0x2203**). If 16 bit A2D is not supported, points to drive temperature
	1:	Analog feedback amplitude. For 1Vp-p ~2,380,000
	2:	Analog input signals: cosine << 16 | sine (in A2D units) For 1Vp-p about +/-1600.
	3:	Drive temperature similar to 0x22A3.2
	4:	Analog input 2
20..31	Reserved	

17.43. Object 0x2F45: Threshold parameter

This object performs read and write access to threshold parameters similar to the **XT** command. Object **0x2085** reports the status of the threshold signals.

- Object description:

Index	**0x2F45**
Name	Threshold parameters
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of sub-indices
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	4
Default value	4

Sub-index	**1**
Description	Under voltage threshold in mVolts
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…**BV***1000
Default value	0

Sub-index	**2**
Description	Over voltage threshold in mVolts
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…**BV***2000
Default value	Drive dependent **BV***1000

Sub-index	**3**
Description	Analog Encoder Amplitude Threshold
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…10000000
Default value	0

Sub-index	**4**
Description	Over temperature Threshold in Celsius
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…125
Default value	85

17.44. Object 0x2F70 – CAN Encoder Range

This object defines the range of the CAN Encoder. The low limit is stored in sub-index 1 and the high limit in sub-index 2. It should be noted that the difference of the limits must be an even number. Object is used to accumulate the position of the CAN encoder and to calculate the reference for the follower mode used via the CAN encoder.

- Object description:

Index	0x2F70
Name	Can Encoder Range
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

- Entry description:

Sub-index	**0**
Description	Number of entries
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	2
Default value	2

Sub-index	**1**
Description	Low Limit
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	Zero value only is permitted
Default value	0

Sub-index	**2**
Description	Hi Limit
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	1…0x7FFFFFFF
Default value	0x0FFFFFFF

17.45. Object 0x2F75 – Extrapolation Cycles Timeout

This object defines the number of extrapolation cycles to be performed by the drive before it stops the motion in time depended motion mode such as Cyclic Position and Interpolated Position. In case of Interpolated Position mode, an EMCY message will be transmitted when the cycle counts are exhausted. The object can be accessed by the **OV[63]** command.

- Object description:

Index	0x2F75
Name	Extrapolation Cycles Timeout
Object code	VAR
Data type	INTEGER16
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	1…32767
Default value	1

17.46. Objects 0x3000 to 0x32A3: Elmo parameters objects

The 0x3000 to 0x32A3 objects can be used to address any of Elmo's parameters via SDO. The reference list from Elmo command to object index is indicated in the Gold Command Reference.

Note that the object is relevant only if the alias Elmo's command exists. Otherwise an appropriate Abort message is transmitted.

The maximum sub-index that can be accessed via objects 0x3XXX is 255. Some Elmo parameters have more sub-indexes which cannot be accessed via the objects.

Refer to Chapter 4: Addressing Elmo Parameters via CANopen Objects for more details.

- Object general description

Index	0x3000 – 0x32A3
Name	Elmo parameters objects
Object code	ARRAY
Data type	UNSIGNED32
Category	O

- Entry Description

Sub-index	**0**
Description	Highest sub-index supported
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	None
Default value	None

Sub-index	**1…255**
Description	Local Error Reaction
Entry category	Optional
Access	R/W
PDO mapping	No
Value range	0…(2^{32})-1
Default value	None

Note that a specific object and the sub index is subjected to the restrictions of the alias Elmo command. Please refer to the Command reference manual.

When access to object 0x30XX via CANopen SDO fails, the drive returns an SDO abort message that contains the SDO abort code. Detailed reason of the abort can be retrieved with uploading object 0x2081.4 or **EE[4]** via terminal.

Example 1

Try to set "MO=1" by sending **Object 0x3146** sub index 0 value 1 under condition of undervoltage. In this case the drive returns SDO abort message "General error", **EE[4]** and **0x2081.4** return error 233 "Undervoltage"

The following are the CANopen messages:

Client sends MO=1

```
601     SDO     22 46 31 00 01 00 00 00
```

Server responds with abort message "General error"

```
581             80 46 31 00 00 00 00 08
```

Client retrieves object 0x2081.4 (alias EE[4])

```
601     SDO     40 81 20 04 00 00 00 00
```

Server responds "Undervoltage"

```
581             43 81 20 04 E9 00 00 00
```

Example 2

Try to set UI[50]=200 by sending **Object 0x3120** sub index 50 value 200 (UI max sub index is 24). In this case the drive returns SDO abort message, **EE[4]** and **0x2081.4** return error 3 "Sub index does not exists"

he following are the CANopen messages:

Client sends UI[50]=200

```
601     SDO     22 20 31 32 C8 00 00 00
```

Server responds with abort message "Sub index does not exist"

```
581             80 20 31 32 11 00 09 06
```

Client retrieves object 0x2081.4 (alias EE[4])

```
601     SDO     40 81 20 04 00 00 00 00
```

Server responds "Bad index"

```
581             43 81 20 04 03 00 00 00
```

Chapter 18: ECAT only Objects

18.1. Object 0x10E0: Device ID reload

The Device ID reload object is used to reload registers in EtherCAT Slave Controller that are available using SDO. The object is defined in ETG1020.

The purpose of the entry sub index 1 *Configured Station Alias Register* is to permit remote change of the Device ID in Configured Station Alias register 0x0012 without power cycle of device.

A *Reload EEPROM* command to the ESC register 0x0502 is not sufficient, since all registers except register 0x0012 and 0x0140.9 are reloaded. Read access returns the current value of register 0x0012. Write access permits write value into register 0x0012.

Sub-index 2 *Write Configured Station Alias Persistent* may be set to 0 only.

- Object description

Index	0x10E0
Name	Device ID reload
Object code	ARRAY
Data type	INT16
Category	O

- Entry Description

Sub-index	**0**
Description	Number of entries
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	2
Default value	2

Sub-index	**1**
Description	Configured Station Alias Register
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…32767
Default value	0

Sub-index	**2**
Description	Write Configured Station Alias Persistent
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0
Default value	0

18.2. Object 0x1C12: SM2 (Outputs) PDO assignments

This object presents the Sync Manager 2 (Outputs) PDO assignment.

Note that up to 32 bytes can be mapped to a single SM.

- Object description

Index	0x1C12
Name	SM2 (Outputs) PDO assignment
Object code	ARRAY
Data type	UNSIGNED16
Category	M

- Entry Description

Sub-index	**0**
Description	Number of assigned RxPDOs
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…30
Default value	1

Sub-index	**1-30**
Description	PDO Mapping object index of assigned RxPDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…65535
Default value	Sub-index 1 – 0x1600 Sub-index 2 – 0x1601 Sub-index 3 – 0x1602 Sub-index 4 – 0x1603 Sub-index 5 – 0x1604 Sub-index 6 – 0x1605 Sub-index 7 – 0x1606 Sub-index 8 – 0x1607 Sub-index 9 – 0x160A Sub-index 10 – 0x160B Sub-index 11 – 0x160C Sub-index 12– 0x160D

	Sub-index 13 – 0x160E Sub-index 14 – 0x160F Sub-index 15 – 0x1611 Sub-index 16 – 0x1612 Sub-index 17 – 0x1613 Sub-index 18 – 0x1614 Sub-index 19 – 0x1615 Sub-index 20 – 0x1616 Sub-index 21 – 0x1617 Sub-index 22 – 0x1618 Sub-index 23 – 0x1619 Sub-index 24 – 0x161A Sub-index 25 – 0x161C Sub-index 26 – 0x161D Sub-index 27 – 0x161E Sub-index 28 – 0x161F Sub-index 29 – 0x1620 Sub-index 30 – 0x1621

18.3. Object 0x1C13: SM3 (Inputs) PDO assignments

This object presents the Sync Manager 3 (Inputs) PDO assignment.

Note that up to 32 bytes can be mapped to a single SM.

- Object description

Index	0x1C13
Name	SM3 (Inputs) PDO assignment
Object code	ARRAY
Data type	UNSIGNED16
Category	M

- Entry Description

Sub-index	**0**
Description	Number of assigned TxPDOs
Entry category	Mandatory
Access	Read/Write
PDO mapping	No
Value range	0…35
Default value	1

Sub-index	**1-30**
Description	PDO Mapping object index of assigned TxPDO
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…65535
Default value	Sub-index 1 – 0x1A00 Sub-index 2 – 0x1A01 Sub-index 3 – 0x1A02 Sub-index 4 – 0x1A03 Sub-index 5 – 0x1A04 Sub-index 6 – 0x1A07 Sub-index 7 – 0x1A0A Sub-index 8 – 0x1A0B Sub-index 9 – 0x1A0C Sub-index 10 – 0x1A0D Sub-index 11 – 0x1A0E Sub-index 12– 0x1A0F

	Sub-index 13 – 0x1A10 Sub-index 14 – 0x1A11 Sub-index 15 – 0x1A12 Sub-index 16 – 0x1A13 Sub-index 17 – 0x1A14 Sub-index 18 – 0x1A15 Sub-index 19 – 0x1A16 Sub-index 20 – 0x1A17 Sub-index 21 – 0x1A18 Sub-index 22 – 0x1A19 Sub-index 23 – 0x1A1A Sub-index 24 – 0x1A1B Sub-index 25 – 0x1A1C Sub-index 26 – 0x1A1D Sub-index 27 – 0x1A1E Sub-index 28 – 0x1A1F Sub-index 29 – 0x1A20 Sub-index 30 – 0x1A21 Sub-index 31 – 0x1A22 Sub-index 32 – 0x1A23 Sub-index 33 – 0x1A24 Sub-index 34 – 0 Sub-index 35 – 0

18.4. Object 0x10F1: Sync Error Setting

This object defines the error setting used Error Reaction behavior of the slave.

The entry sub index 1 *Local Error Reaction* cannot be changed, it is set by default to 2, meaning; Device specific state.

If sub index 2 Sync Error Counter Limit. A weighted internal sync error counter is used to count missed and received SM events. The drive increments this counter by 3 in case of missed events and decrements by 1 when SM events are received.

If the counter exceeds the value defined by sub index 2 a multiply error is detected and the drive changes its ECAT state to SAFEOP with AL status 0x1A. The sync error counter is reset when the AL error is acknowledged.

If sub index 2 is set to 0 the drive does not change its ECAT state to SAFEOP in case of SYNC error.

- Object description

Index	**0x10F1**
Name	Sync error setting
Object code	ARRAY
Data type	UNSIGNED32
Category	O

- Entry Description

Sub-index	**0**
Description	Highest sub-index supported
Entry category	Mandatory
Access	Read only
PDO mapping	No
Value range	2
Default value	2

Sub-index	**1**
Description	Local Error Reaction
Entry category	Optional
Access	Read only
PDO mapping	No
Value range	2
Default value	2

Sub-index	**2**
Description	Sync Error Counter Limit
Entry category	Optional
Access	Read/Write
PDO mapping	No
Value range	0…(2^{32})-1
Default value	0

18.5. Object 0x2046: DC Clock inhibit time

This object is reserved for compatibility reasons and presents distributed clock inhibit time in milliseconds.

- Object description:

Index	0x2046
Name	DC Clock inhibit time
Object code	VAR
Data type	UINT16
Category	O

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	0-65535
Default value	0

18.6. Object 0x2061: FoE Download Parameters Error

In case FoE download parameters process fails, the object includes the last FoE download parameters error value.

- Object description:

Index	2061h
Name	FoE Download Parameters Error
Object code	VAR
Data type	UINTEGER16
Category	Optional

- Entry description:

Access	Read only
PDO mapping	No
Value range	0-65535
Default value	0

Value	Description
0	No error
1-255	See ELMO **EC** command description. The command that caused the error can be read via object 0x2062
1001	Parameters file XML format error
1002	Timeout waiting for response from DSP
1003	Timeout between two consecutive FoE messages
1004	FoE Download cannot start in case motor is on or user program is in progress
All other values	Reserved for future usage

18.7. Object 0x2062: FoE Parameters Last Processed Command

This object includes the last parameter string received during parameters download via FoE.

The object can be used in case the FoE download parameters failed. In this case the object shows the command that returned with error.

- Object description:

Index	0x2062
Name	FoE Parameters Last String Send To Drive
Object code	VAR
Data type	string
Category	Optional

- Entry description:

Access	Read only
PDO mapping	No
Value range	No
Default value	Empty string

Value	Description
Empty string	No command processed, or FoE parameter download not used.
Non Empty string	The last command processed during FoE parameters download.

18.8. Object 0x20E0: ECAT alias ID object

Elmo alias register (same logic **as AA[30]**, see Gold drive Command Reference Guide).

- Object description:

Index	0x20E0
Name	Ecat alias object
Object code	VAR
Data type	UNSIGNED16
Category	Optional

- Entry description:

Access	Read/Write
PDO mapping	No
Value range	0-65535
Default value	0

Chapter 19: Little and Big Endians

The *end* in *endians* refers to the address of the most significant or least significant byte in a multiple-byte data type (such as short, long or float). The address of big endians is the most significant byte (the *big* end) while the address of little endians is the least significant byte (the *little* end).

Example using standard C conventions:

```
long lType;
short sType[2]
char cType[5] = “ABCD”;

lType = 0x12345678;
sType[1] = 0x1234 ;
sType[2] = 0x5678 ;
```

Memory structure: big endians (starts with MSB):

00   08
12 34 56 78 - lType
12 34 56 78 - sType
AB CD 0   - cType

Memory structure: little endians (starts with LSB):

00   08
78 56 34 12 - lType
34 12 78 56 - sType
AB CD 0   - cType

The CANopen protocol supports the little endian method; for example, a node replies to a TPDO with three objects:
Object 1: Signed24 - 0x12ABCD
Object 2: Unsigned32 - 0x123456AB
Object 3: Unsigned8 - 0x1F

The CAN octet will be as follows:
CD AB 12 AB 56 34 12 1F

Inspiring Motion
Since ***1988***

For a list of Elmo's branches, and your local area office, refer to the Elmo site www.elmomc.com