Inspiring Motion
Since 1988

EtherCAT Application Manual

January 2016 (Ver. 1.507)

www.elmomc.com

Elmo
Motion Control

Notice

This guide is delivered subject to the following conditions and restrictions:

- This guide contains proprietary information belonging to Elmo Motion Control Ltd. Such information is supplied solely for the purpose of assisting users of the EtherCAT Application Manual in its installation.
- The text and graphics included in this manual are for the purpose of illustration and reference only. The specifications on which they are based, are subject to change without notice.
- Information in this document is subject to change without notice.

Elmo Motion Control and the Elmo Motion Control logo are registered trademarks of Elmo Motion Control Ltd.

EtherCAT Conformance Tested. EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Document no. G-ETHERCATAM (Ver. 1.507)

Revision History

Version	Date	Details
1.40	21/05/2012	Document completely revised. Initial new document
1.401	12/11/2012	Updated to include EtherCAT and CAN Object tables
1.402	27/11/2012	Corrections to EtherCAT explanation and objects
1.403	29/11/2012	Addition of EtherCAT switch details
1.404	10/03/2013	Correction in tables of objects, emergency and abort codes
1.500	31/07/2013	Changes to CAN and EtherCAT Object tables
1.501	16/09/2013	Changes to template. Update of EtherCAT errors list Changes to captures. Changes by Vladyslav: Section 5.1: Object 0x20FD type changed from INT32 to UNSIGNED INT32

		Section 5.1: Object 0x2E00 “Gain scheduling manual index” added Section 5.1: Object 0x2E06 “Torque window” added Section 5.1: Object 0x2E07 “Torque window time” added Section 5.1: Objects 0x6072 0x6073 type changed to UINT16 Section 5.1: Objects 0x2060,0x2061,0x2062 are added Section 5.1: Object 0x2087 “PAL version” added Section 5.1: Object 0x3000 “ELMO command object” added Table 7: Object 0x1A08 added Table 7: Object 0x1A24 added Table 5: Object 0x1608 added Table 5: Object 0x161A added Section 5.2.1 and 5.2.2 are removed, reference added to the text in 5.2 Section 2.3.2: Text was changed, the new timing diagram was placed Section 2.3.3 was added Section 5.1 Object list was updated
1.502	March 2014	Corrections to sections 2.2.1.1 tables 4 and 5
1.503	April 2014	Updated Chapter 5 Gold Drive Object List
1.504	April 2015	Updated Chapter 3 Object Dictionary Table
1.505	Sep 2015	Updates to the Table 11 Object List
1.506	Dec 2015	Updated Chapter 3 Object Dictionary Table Update Tab.6, tab 8 Changes to Chapter 5.2
1.507	Jan 2016	Addition of Platinum Maestro to Gold Maestro family reflected in Chapter 2, Chapter 4, and throughout document where G-MAS is replaced by Maestro

Chapter 1: Introduction

The ELMO EtherCAT environment is extensive and abounding in features. ELMO provides a comprehensive solution for the EtherCAT system, which includes the Maestro Network Motion Controller family EtherCAT Master, Servo Drives, EtherCAT Slaves and EtherCAT configuration tools.
ELMO EtherCAT is standard compliant and is successfully EtherCAT conformance tested.

This manual describes the installation, setup, range of functions and EtherCAT protocols for Elmo's EtherCAT-ready products.

1.1. What is EtherCAT?

Ethernet for Control Automation Technology (EtherCAT) is an open high performance Ethernet-based fieldbus system, which uses the family of industrial computer network protocols used for real-time distributed control, now standardized as IEC 61158. It is a highly flexible Ethernet network protocol, running over a fast real time Master–Slave network.

The EtherCAT communication speed is up to 100 Mbps full duplex and can include a maximum of 65,535 stations in a single network configuration such as Ethernet star, line or tree without using switches.

Figure 1 describes a network of EtherCAT slaves in a ring topology. The Master controls the traffic in the network by initiating the transactions.

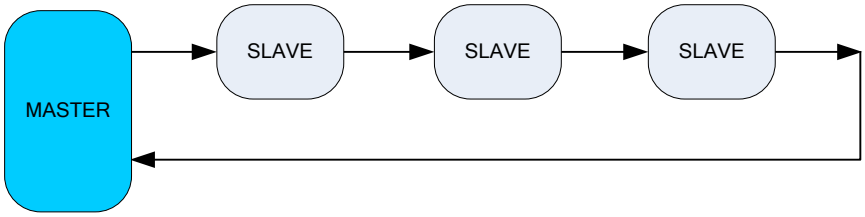


Figure 1: EtherCAT Network Configuration

Usually, a control system requires the following in periodic time intervals:

- **Inputs**
  Messages from the ECAT device to the Master latches data such as Positions, Velocities, Currents, System Status, IO's etc,
- **Outputs**
  Messages from the ECAT Master to the device with data or commands such as Control word commands, Trajectory Information (set point), or Higher Drive Level Commands.

The specific nature of the data transferred via the network depends on the operation mode of the slave drive. The Device Profile describes the application parameters and the functional behavior of the devices including the device class-specific state machines. A common standard for the Servo Drive is the DS-402 for Drive and Motion Control device Profile, which can be addressed via CoE (Can Over EtherCAT).

The EtherCAT protocol is optimized for process data and is transported directly within the standard IEEE 802.3 Ethernet frame. Each Ethernet frame can include several EtherCAT frames, each serving another slave.

EtherCAT network uses a processing on the fly, whereby the Ethernet frame is received and processed, while the telegram passes through the device. The frames only delay by a fraction of a microsecond in each node. Using EtherCAT, the entire network can be addressed with just one frame.

The data sequence is independent of the physical order of the nodes in the network; addressing can be in any order. Broadcast, multicast and communication between slaves are possible and must be performed by the master device.

The EtherCAT protocol can be inserted into UDP/IP datagrams. This also enables any control with an Ethernet protocol stack to address EtherCAT systems.

Using the Master configuration tool, the Master scans the EtherCAT network and uses the EtherCAT Slave library (ESI: EtherCAT Slave Information in XML format) to compare the slave memory area that includes information about the slave such as Vendor ID, Product Code, and Slave Configuration.

1.2. Terms and Abbreviations

The following table (Table 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: Shortened Terms

1.3. Elmo EtherCAT

For a complete motion solution, the ELMO environment comprises of three levels:

- **EAS**; EtherCAT configuration tools
- **Maestro**; EtherCAT Gold or Platinum Maestro master
- **Gold Servo Drive**; Elmo EtherCAT slave drives

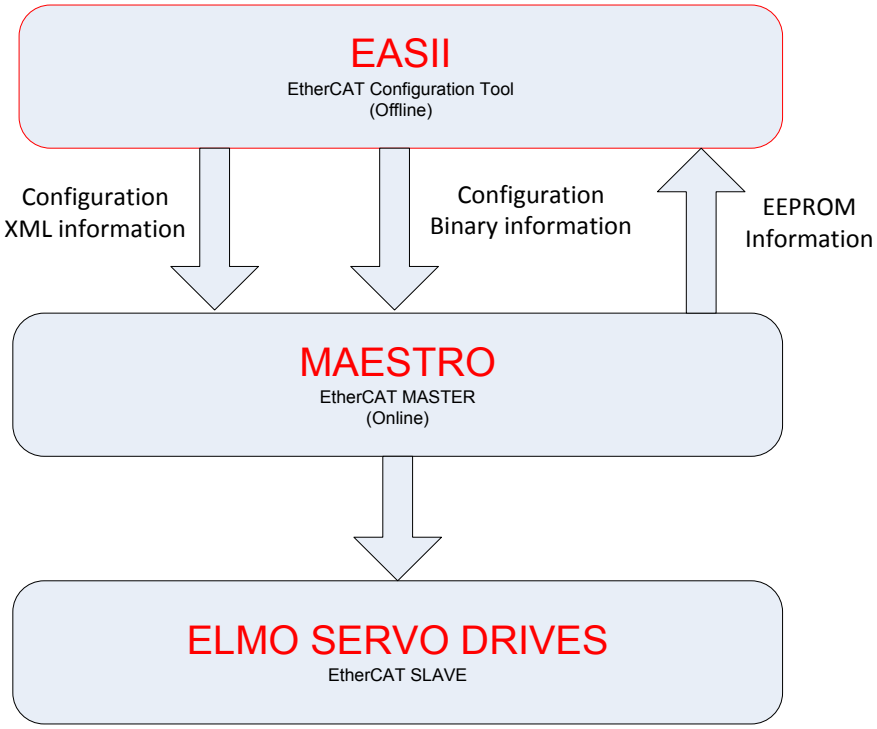


Figure 2: EtherCAT Environment

1.4. Elmo Slave Drives

The following diagram describes the EtherCAT communication of the drive.

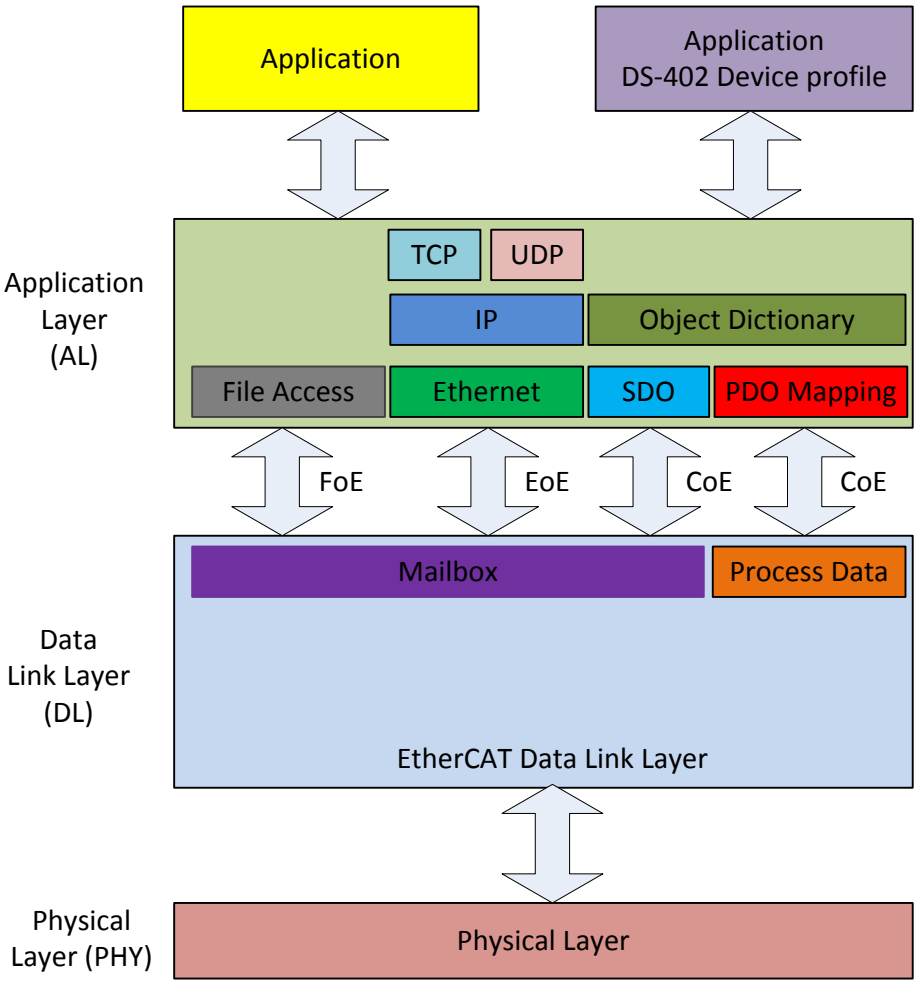


Figure 3: Layered Communication protocol in EtherCAT

Physical Layer

The Physical layer of the EtherCAT is a 100Mbits/sec Ethernet port over twisted per cable.

Data Link Layer

Supports two mechanisms of data transfer:

- **Process data**
  Allows writing and reading data simultaneously. This mode is used to transfer the Process data objects (PDO). The PDO transfers via SYNC Manager 2 (rPDO) and SYNC Manager 3 (tPDO)
- **Mailbox**
  The mailbox mechanism assures that the data will reach to the target without overlapping previous data. The mailbox is used to transfer the SDOs. The SDO transfers via SYNC Manager 0 (MailboxOut) and SYNC Manager 1 (MailboxIn). SDO objects are used for user triggered access. With SDO services, all of the OD's entries can be accessed. The SDO transport works in asynchronous mode.

The Elmo drive supports the following communication protocols:

CoE (CANopen over EtherCAT)	Defines a standard way to access the CANopen protocol and includes an object dictionary, SDO, PDO and emergency messages.
EoE (Ethernet over EtherCAT)	Fully Ethernet compatible, defines a standard way to exchange or tunnel standard Ethernet frames. This is typically used to address the drive inherent parameters such as control parameters
FoE (File over EtherCAT)	Similar to TFTP, defines a standard way to download firmware to the servo drive.

The **Object Dictionary (OD)** contains parameters, application data and the mapping information between the Process Data Interface (PDI) and application data. Its entries can be accessed via the Service Data Object (SDO).

An Object Dictionary is a naming system that provides a unique identifier to each data item or "object" communicated over the CoE protocol. An object is identified by an index and sub-index. It contains variables, arrays and complex objects. CoE and EoE protocols require a set of mandatory objects. Elmo's OD is compliant with the DS402 V3 object list for Drive and Motion control device. The DS402 defines standard objects for the following motion modes; Profile Position, Profile Velocity, Profile Torque, Homing mode, Synchronous Cyclic Position, Synchronous Cyclic Velocity and Synchronous Cyclic Torque.

Elmo drive supports distributed clock in order to synchronize between the Master and Slaves on the EtherCAT network. Refer to the section 2.3 Synchronization Modes for details.

1.4.1. CTT (Conformance Test Tool)

The CTT (Conformance Test Tool) is the official tool used by the EtherCAT Technology Group (ETG) for EtherCAT conformance certification. The CTT includes thousands of tests to verify that the device complies with the EtherCAT definition, requirements and standard. The Gold drive firmware version has passed the CTT in several quality assurance (QA) stations, making sure that the ETG Certificate given to Elmo is valid.

1.5. Elmo Maestro Master

While a single servo drive can run as a stand-alone drive using its inner profiler and filter, in order to perform synchronized multi axis motions in the system (such as circle, line etc.), a real time communication protocol must be used, and all drives must be synchronized to a specific SYNC signal in the system. The Gold or Platinum Maestro Network Motion Controller performs this task and operates as a master, independent of any host system. In operational mode, it periodically sends data to the slaves that may override the data that a user sends from a host system. Therefore, for example, the user cannot tune an axis if the axis is in operational mode.

Figure 4 describes the Maestro general communication and API architecture. This diagram describes two types of communication mechanisms:

- Inter-process communication (IPC) which is a set of techniques to exchange data among multiple threads in one or more processes. C Programs located on the Maestro use the IPC mechanism to communicate, and the MDS can communicate directly with the Maestro via C programs.
- A remote procedure call (RPC) is an inter-process communication with the Maestro host allowing a program to initiate a subroutine or procedure. The programmer essentially writes the same code whether the subroutine is local to the executing program, or remote. For example, the EASII uses RPC to communicate with the Maestro family motion controllers.

Both these mechanisms allow the GDS and Elmo Application Studio (EASII) to communicate using TCP/IP and EtherCAT over TCP/IP, with the Maestro to perform master operations.

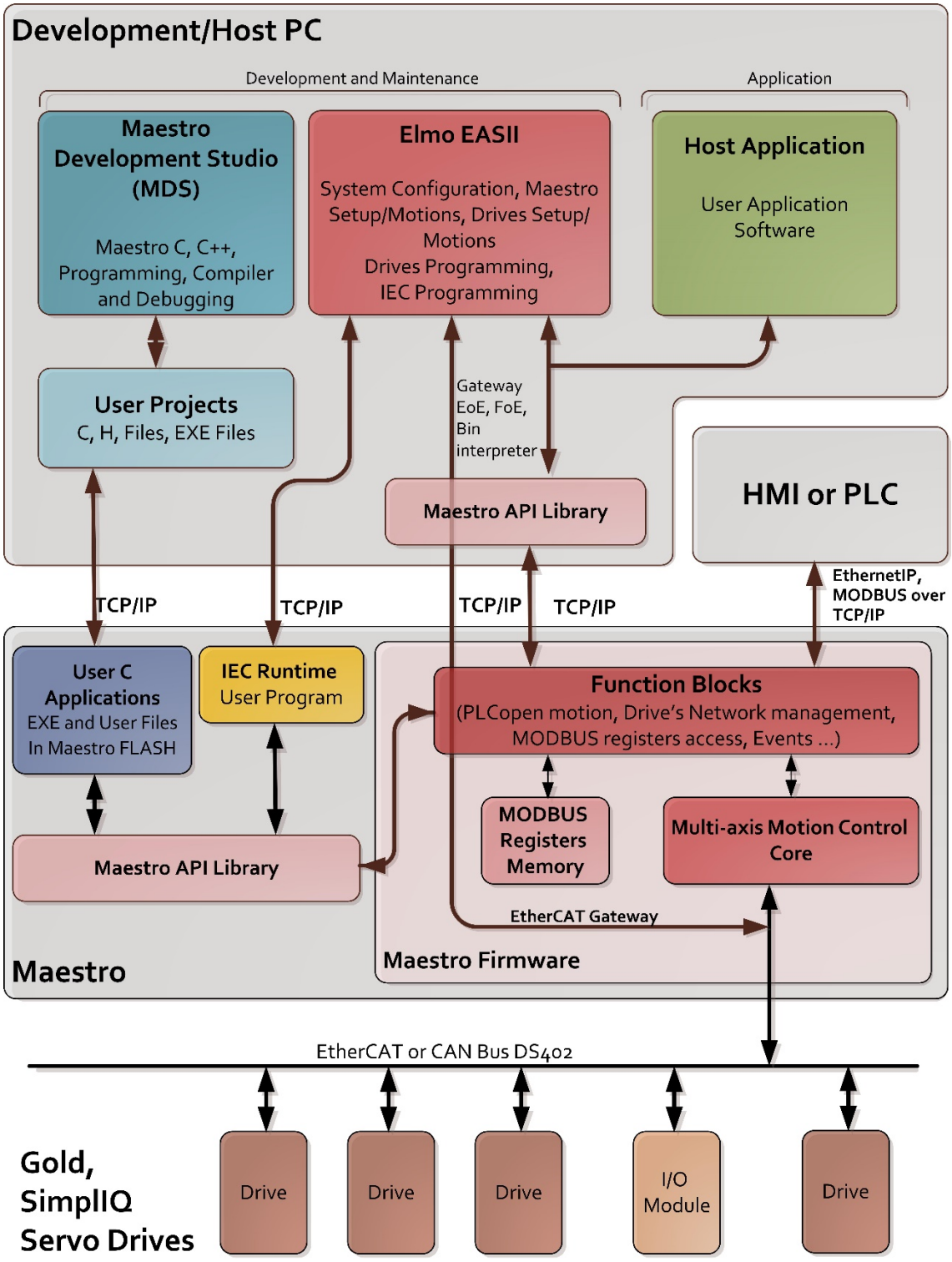


Figure 4: GoS System Software Structure –Host PC Development

In addition, since the Maestro operates as a master, independent of any host system, in operational mode, it periodically sends data to the slaves, that may override the data that a user sends from a host system. Therefore, for example, the user cannot tune an axis if the axis is in operational mode. To prevent this and allow the EAS application (Figure 4) to operate via the Maestro CANbus and EtherCAT, specific API functions are called to change the Maestro operation and allow these applications to function.

1.5.1. For EASII and MDS Applications

For the EASII and Maestro Developer Studio (MDS) applications to monitor and perform motions, the Maestro cannot operate in the background. A special API function changes the EtherCAT and CANbus communication from the Maestro to Pre-Operation mode, causing the following:

EtherCAT **No process cycle operates**

CANbus **No outputs via CAN, and no state machines run**

These API functions change the GMAS mode so that the GMAS operation is transparent and no messages transfer between the GMAS and the drives.

In order to configure the EtherCAT network (EtherCAT Configuration Mode) via the EASII application, the Maestro must be set to EtherCAT Configuration mode. The user is then able to perform the operations. The API then employs the specific functions to change the Maestro back to operational mode.

1.6. Maestro Operation Modes

To optimize the device network usage, the Maestro supports two modes of operating axes present on the Device Network:

- NC Axes – for Numeric Control Axes
- Distributed – for axes not under strict numeric control

The main difference between these modes is the way the motion profile is calculated, and as a result, the synchronization level achieved.

In general, for axes not requiring low level (network) motion synchronization, the Distributed mode should be used, allowing the servo drives to generate their own motion trajectory, thus reducing network load. In this case, synchronized motions like ECAM, based on an external master encoder can still be executed. For highly synchronized motions, generated by the Master controller (referred to under the PLCopen definitions as group vector motions), the NC mode should be used.

1.6.1. NC and Distributed / Standard DS-402 (stand-alone) Axes Motion Modes

In NC mode, the Maestro controls the motion, handling the axis (and motion) State (as defined by the PLCopen Standard), and calculating the motion profile as part of its real-time loop process (NC Cycle). Servo drives operating with a Maestro master under this mode will run under the DS-402 motion modes e.g.; Interpolated position, or one of the Cyclic Sync modes (Position/Velocity).

In Distributed/Standard DS-402 mode, the Maestro uses the servo drives own DS-402 operation modes, where the drive itself controls its own profiling as part of its Real Time process. The Maestro only synchronizes start/stop and general activation functions, but is not responsible to the low-level real-time profile generation.

The Maestro can mix NC and Distributed axes in the same network configuration, thus optimizing usage of network and processor resources.

1.6.2. Maestro to Servo Drive Interfaces

The Maestro manages all motion commands sent to the servo drives, via the CANopen DS-402 standard (Refer to Figure 4). This is relevant to the Maestro CAN hardware interface, and to the EtherCAT protocol implementing CoE (CAN Over EtherCAT).

For axes (Nodes) that operate in NC mode, Maestro uses the DS-402 motion modes: Interpolated position, or one of the Cyclic Sync modes (Position/Vel).

For axes (Nodes) operating in Distributed mode, where the servo drive manages its own profiler and real-time motion execution, it is assumed that the servo drive supports the relevant requested motion modes.

Motion Modes that are part of the PLC Motion API definition, but are **not** supported by the DS-402 interface, will not be available in standard DS-402 servo drives when working in Distributed mode (unless specific Vendor Types objects are defined, e.g. ECAM in drive level, etc. as implemented for example in Elmo servo drives).

The Maestro uses the EtherCAT communication protocol to enable synchronized motion of all the controllers to the same SYNC signal. Thus, all drives in the system are synchronized to the master clock, and all generate an interrupt at exactly the same time.

A profiler can run in the Maestro, on the condition that the axis (axes) is defined as a vector axis (axes). A vector axis may consist of 1 - 16 axes. The Multi Axis Indexer (MAI) is the profiler that runs within the Maestro, which sends (via a high priority interrupt routine) a calculated set point to the axes in the system and can perform vector calculations for up to 16 axes. The profiler EtherCAT outputs are points that are to be sent to the specific drives belonging to the vector. Therefore, a number of combination options are available:

- 1 x 16 axes (One vector profiler performing profiles for 16 axes),
- 16 x 1 axes (16 profilers for 16 vector axes), or,
- Any combination of M x N axes – as long as M x N < 16

The SYNC interrupt signal to the drives is based on the ET1100 component in the servo drive. The master Maestro does not receive this signal, but can calculate when the SYNC signal is generated. This is because the master EtherCAT in the Maestro is responsible for updating the SYNC cycle time in the servo drives, and therefore knows when the SYNC is generated. The MAI can operate at varying cycle times, dependent on a number of parameters, such as the:

- Desired response from the system
- Number of axes participating in the MAI. The more axes, the higher the cycle rate

1.7. ELMO Application Studio (EASII) – Configuration Tool

The EtherCAT configuration tools enable configuration and monitoring of the network from the EAS application.

Using the Master configuration tool, the Master scans the EtherCAT network and uses the EtherCAT Slave library to compare the slave memory area that includes information about the slave such as Vendor ID, Product Code, and Slave Configuration.

Chapter 2: Indicators

This section describes the drive and EtherCAT indicators used to support visual inspection and troubleshooting of the drive and networks

2.1. Drive Indicators

2.1.1. Status LED Indicator

Figure 5 shows the position of the red/green dual LED in the Gold Bassoon, a typical servo drive, used for immediate indication of the Initiation and Working states, i.e. the drive status.

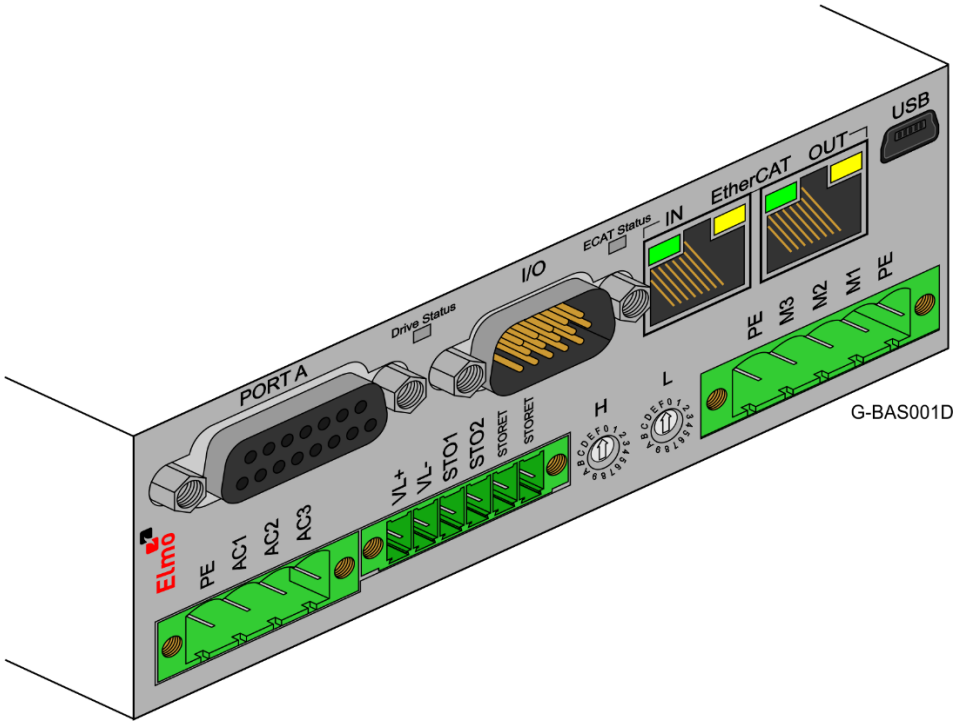


Figure 5: Gold Bassoon Drive Status Indicator

The LED color varies between green or red. The LED states are defined in the table below.

Indicator state/color	Definition
Off	No power supply to drive
Temporary blinking red and then Off	The drive is in BOOT state. Firmware download is required and should be performed now.
Red	The drive is in fault state e.g. Safety switches or Low bus voltage)
Green	Drive is ready

2.1.2. EtherCAT Link/Activity Indicators

Each EtherCAT slave device includes two RJ-45 connectors; EtherCAT IN and EtherCAT OUT as shown in Figure 5:

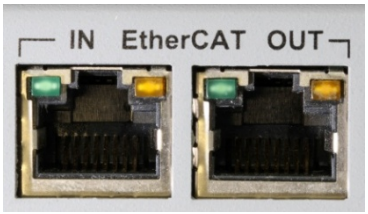


Figure 6: EtherCAT ports

Figure 6 describes the two status LEDs for each RJ-45 connector. The link/activity indicators show the state of the physical link and activity on this link.

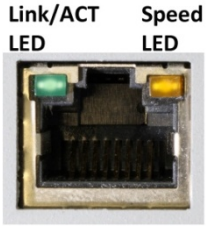


Figure 7: Link/Activity Indicators

Table 2 displays the LEDs' link and activity.

LED	State		Definition
Link and/or Activity	Off	No Connection	Green color, defines the state of the physical link/activity of the link.
	On	Connection Established (Link)	
	Flashing	Data transmission active (Act)	
Speed	On	100Mbps Connection (default) otherwise there is no EtherCAT connection	Orange color, define the speed of the EtherCAT line.
	Off	10Mbps connection	

Table 2: LED Functionality

2.1.3. EtherCAT Status Indicator

This indicator is one bi-colored led that combines the green RUN indicator (Table 3: RUN Indicator States) and the red ERROR indicator (Table 4: ERROR Indicator States) of the EtherCAT state machine. The LED indicator signals are based on ETG1300 of which Elmo drives are fully certified.

Indicator State	ESC State	Description
Off	Initialization	Elmo drive is in state INIT
Blinking	Pre-Operational	Elmo drive is in state Pre-OP
Single Flash	Safe-Operational	Elmo drive is in state Safe-OP
On	Operational	Elmo drive is in state OP
Flickering	Initialization or Bootstrap	Elmo device is booting and has not yet entered the INIT state, or Elmo device is in state Bootstrap. Firmware download operation in progress
Triple Flash	Device Identification	User sets this state from the master to locate the specific slave

Table 3: RUN Indicator States

Error State	Error Name	Description
On	Application controller failure	An critical communication or application controller error has occurred
Double Flash	Process data watchdog timeout / EtherCAT watchdog timeout	An application watchdog timeout occurred
Single Flash	Local error	Elmo device application has changed the EtherCAT state autonomously, due to local error.
Blinking	Invalid configuration	General configuration error
Flickering	Booting error	Booting error was detected
Off	No error	The EtherCAT communication of the device is in working condition.

Table 4: ERROR Indicator States

2.1.4. EtherCAT Address Switches (where present on servo drive)

The EtherCAT address of each axis is specified by two switches. The position of the EtherCAT switches, may either be found, inside the battery compartment as in the Gold Duet, or in other positions on the drive or drivers interface. Use a fine screwdriver to set the low and the high bytes of the EtherCAT address.

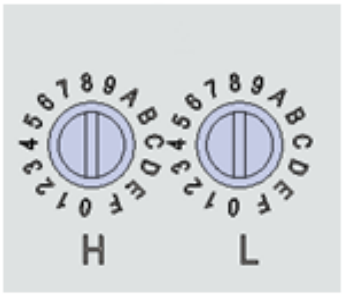


Figure 8: EtherCAT Address Switches

2.2. Maestro Indicators

This section describes the Platinum and Gold Maestro indicators and EtherCAT indicators used to support visual inspection and troubleshooting of the drive and networks.

2.2.1. Status LED Indicator

Figure 5 shows the position of the red/green dual LED in the Platinum and Gold Maestro, used for immediate indication of the Initiation and Working states, i.e. the drive status.

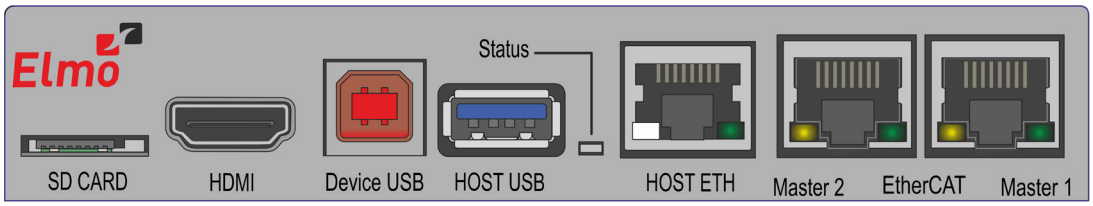


Figure 9: Platinum Maestro Status Indicator

The red/green dual LED is used for immediate indication of the following states:

Maestro	States	LED indication
Platinum	**Initiation**	The LED indicates whether the drive is in the boot state (blinking red) or in the operational state (steady green).
	Error	The LED indicates whether the motion controller is in error state (blinking green/red).
Gold	**Initiation**	The LED indicates whether the drive is in the boot state (blinking red) or in the operational state (steady green).
	Error	The LED indicates whether the motion controller is in error state (blinking green).

2.2.2. EtherCAT Link/Activity Indicators

The following table describes re the Gold and Platinum Maestro EtherCAT Link/Activity Indicators.

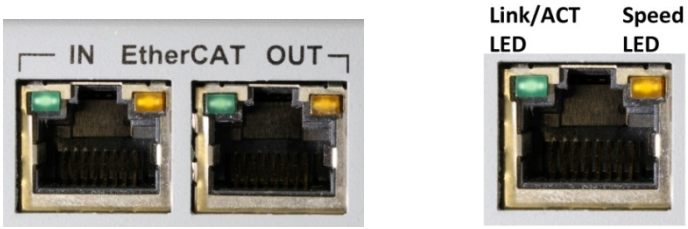
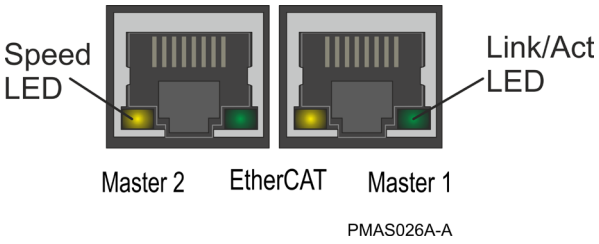
Hardware	EtherCAT Ports and EtherCAT Staus LEDs
Gold Maestro	
Platinum Maestro: **Master 1 and 2** **Slave IN and OUT**	

Table 5: Maestro EtherCAT Ports

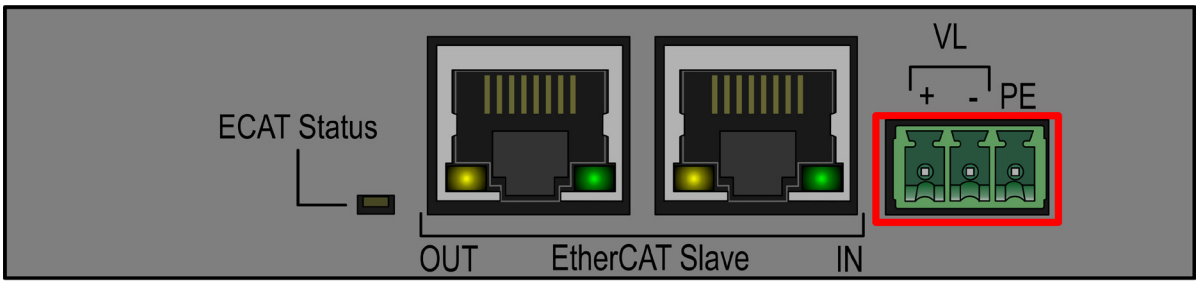
The green LED is the link/activity indicator (Table 5). It shows the state of the applicable physical link and the activity on that link.

The amber LED is the speed indicator (Table 5). It shows the speed of the connection on the Ethernet line. The possible states of these LEDs are summarized in Table 6.

LED	State	Meaning
Link /Activity	Off	No link is established
	On	A link is established
	Blinking	There is data transmission activity
Speed	On	The connection speed is 100 Mbps The speed of the EtherCAT line must be 100 Mbps. Otherwise, there is no EtherCAT data transmission
	Off	The connection speed is 10 Mbps

Table 6: LED States

2.2.3. EtherCAT Status Indicator



PMAS024A-C

Figure 10: EtherCAT Status LEDs

This indicator is one bi-colored led that combines the green RUN indicator (Table 3: RUN Indicator States) and the red ERROR indicator (Table 4: ERROR Indicator States) of the EtherCAT state machine. The LED indicator signals are based on ETG1300 of which Elmo drives are fully certified.

Indicator State	ESC State	Description
Off	Initialization	Elmo drive is in state INIT
Blinking	Pre-Operational	Elmo drive is in state Pre-OP
Single Flash	Safe-Operational	Elmo drive is in state Safe-OP
On	Operational	Elmo drive is in state OP
Flickering	Initialization or Bootstrap	Elmo device is booting and has not yet entered the INIT state, or Elmo device is in state Bootstrap. Firmware download operation in progress
Triple Flash	Device Identification	User sets this state from the master to locate the specific slave

Table 7: RUN Indicator States

Error State	Error Name	Description
On	Application controller failure	An critical communication or application controller error has occurred
Double Flash	Process data watchdog timeout / EtherCAT watchdog timeout	An application watchdog timeout occurred
Single Flash	Local error	Elmo device application has changed the EtherCAT state autonomously, due to local error.

Blinking	Invalid configuration	General configuration error
Flickering	Booting error	Booting error was detected
Off	No error	The EtherCAT communication of the device is in working condition.

Table 8: ERROR Indicator States

Chapter 3: Elmo EtherCAT Slave Devices

3.1. CoE – CANopen Over EtherCAT

Defines a standard way to access the CANopen protocol and includes an object dictionary, SDO, PDO Emergency and Abort messages.

3.1.1. PDO (Process Data Object)

The PDO protocol is used for communication with SYNC Manager 2 for RxPDO, and SYNC Manager 3 for TxPDO.

Each PDO consist of objects in the object dictionary, which is PDO map able. The PDO mapping objects describes how these objects are related to a PDO.

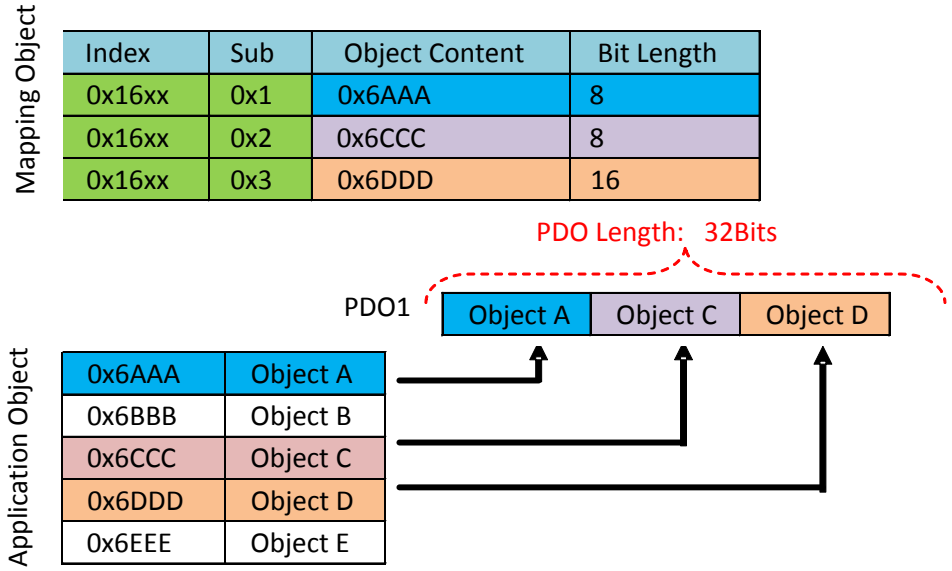


Figure 11: PDO Mapping

Each sync manager channel object describes a consistent area inside the EtherCAT process data and consists of several process data objects.

All PDO mappings are located in the object dictionary from index 0x1C10 to 0x1C2f.

3.1.1.1. Receive PDO Mapping (Outputs)

This defines a list of objects that include data from the master to the slave. All RPDOs are located in the object dictionary from index 0x1600 to 0x17FF.

Sub-index	Description	Data Type	PDO Mapping	Value
0	Number of objects in this PDO	Unsigned8	No	0-8
1	First output object to be mapped	Unsigned32	No	Bit 0-7: length of the mapped objects in bits. Bit 8-15: sub index of the mapped object. Bit 16-31: index of the mapped object
……				
N	Last output object to be mapped	Unsigned32	No	Bit 0-7: length of the mapped objects in bits. Bit 8-15: sub index of the mapped object. Bit 16-31: index of the mapped object

Table 9 : Receive PDO Mapping Configuration

The following table describes Elmo RxPDO's objects (PDO Index, Default Value, Bit Length, Description, Access (R=Read, RW=Read/Write), Exclude):

PDO Index	Default Value	Bit Len	Description	Access	Exclude
0x1600	0x607A 0x60FE:1 0x6040	32 32 16	Target Position Digital Outputs Control Word	R	0x1601-0x1606
0x1601	0x60FF 0x6040	32 16	Target Velocity Control Word	R	0x1600, 0x1602-0x1606
0x1602	0x6071 0x6040	16 16	Target Torque Control Word	R	0x1600-0x1601 0x1603-0x1606
0x1603	0x607A 0x60FE:1 0x60B1 0x6040	32 32 32 16	Target Position Digital Outputs Velocity Offset Control Word	R	0x1600-0x1602 0x1604-0x1606
0x1604	0x607A 0x60FF 0x6072 0x6040	32 32 16 16	Target Position Target Velocity Max. Torque Control Word	R	0x1600-0x1603 0x1605-0x1606
0x1605	0x607A 0x60FF 0x6071	32 32 16	Target Position Target Velocity Target Torque	R	0x1600-0x1604 0x1606

PDO Index	Default Value	Bit Len	Description	Access	Exclude
	0x6072 0x6040 0x6060	16 16 8	Max. Torque Control Word Mode Of Operation		
0x1606	0x607A 0x60FE:1 0x60FF 0x60B1 0x60B2 0x6040	32 32 32 32 16 16	Target Position Digital Outputs Target Velocity Velocity Offset Torque Offset Control Word	R	0x1600-0x1605
0x1607	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000	8 8 8 8 8 8 8 8	Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte	RW	
0x1608	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000	8 8 8 8 8 8 8 8	Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte	RW	
0x160A	0x6040	16	Control Word	R	
0x160B	0x6060	8	Mode Of Operation	R	
0x160C	0x6071	16	Target Torque	R	
0x160D	0x6072	16	Max. Torque	R	
0x160E	0x6073	16	Max. Current	R	
0x160F	0x607A	32	Target Position	R	
0x1610	0x607F	32	Max. Profile Velocity	R	
0x1611	0x6081	32	Profile Velocity	R	
0x1612	0x6082	32	End velocity	R	
0x1613	0x6083	32	Profile Acceleration	R	
0x1614	0x6084	32	Profile Deceleration	R	
0x1615	0x6087	16	Torque Slope	R	

PDO Index	Default Value	Bit Len	Description	Access	Exclude
0x1616	0x60B0	32	Position Offset	R	
0x1617	0x60B1	32	Velocity Offset	R	
0x1618	0x60B2	16	Torque Offset	R	
0x1619	0x60B8	16	Touch Probe Function	R	
0x161A	0x2E00	32	Gain scheduling manual index	R	
0x161C	0x60FF	32	Target Velocity	R	
0x161D	0x60FE:1	32	Digital Output	R	
0x161E	0x607F	8	Polarity	R	
0x161F	0x6085	32	Quick stop deceleration	R	
0x1620	0x22A1:1	32	Extended outputs	R	
0x1621	0x2005	32	Fast reference	R	

Table 10: RxPDO Default Values

3.1.1.2. Transmit PDO Mapping (Inputs)

This defines a list of objects that include data from the slave to the master. All TPDOs are located in the object dictionary from index 0x1A00 to 0x1BFF.

Sub-Index	Description	Data Type	PDO Mapping	Value
0	Number of objects in this PDO	Unsigned8	No	0-254 Writable if variable mapping is supported
1	First output object to be mapped	Ubsgined32	No	Bit 0-7: length of the mapped objects in bits. Bit 8-15: sub index of the mapped object. Bit 16-31: index of the mapped object
…				
N	Last output object to be mapped	Unsigned32	No	

Table 11 : Transmit PDO Mapping Configuration

The following table describes Elmo TxPDO's objects (PDO Index, Default Value, Bit Length, Description, Access (R=Read, RW=Read/Write), Exclude):

PDO Index	Default Value	Bit Len	Description	Access	Exclude
0x1A00	0x6064 0x60FD 0x6041	32 32 16	Position Actual Value Digital Inputs Status Word	R	0x1A01-0x1A04
0x1A01	0x6064 0x606B 0x6074 0x6041	32 32 16 16	Position Actual Value Velocity Demand Value Torque Demand Value Status Word	R	0x1A00, 0x1A02-0x1A04
0x1A02	0x6064 0x6077 0x6041 0x6061	32 16 16 8	Position Actual Value Torque actual Value Status Word Mode Of Operation Display	R	0x1A00-0x1A01, 0x01A03-0x1A04
0x1A03	0x6064 0x60FD 0x606C 0x6041	32 32 32 16	Position Actual Value Digital Inputs Velocity Actual Value Status Word	R	0x1A00-0x1A02, 0x1A04
0x1A04	0x6064 0x60F4 0x6077 0x6041 0x6061	32 32 16 16 8	Position Actual Value Position Following error Value Torque Actual Value Status Word Mode Of Operation Display	R	0x1A00-0x1A03
0xc1A07	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000	8 8 8 8 8 8 8 8	Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte	RW	
0xc1A08	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000	8 8 8 8 8 8 8 8	Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte Dummy Byte	RW	
0x1A0A	0x6041	16	Status word	R	

PDO Index	Default Value	Bit Len	Description	Access	Exclude
0x1A0B	0x6061	8	Mode of operation display	R	
0x1A0C	0x6062	32	Position Demand [UU]	R	
0x1A0D	0x6063	32	Actual position [counts]	R	
0x1A0E	0x6064	32	Position actual value	R	
0x1A0F	0x6069	32	Velocity sensor actual value [counts/sec]	R	
0x1A10	0x606B	32	Velocity demand [cnt/sec]	R	
0x1A11	0x606C	32	Velocity actual value	R	
0x1A12	0x6074	16	Torque demand value	R	
0x1A13	0x6077	16	Torque actual value	R	
0x1A14	0x60B9	16	Touch Probe status	R	
0x1A15	0x60BA	32	Touch Probe Pos1 Positive	R	
0x1A16	0x60BB	32	Touch Probe Pos1 Negative	R	
0x1A17	0x60BC	32	Touch Probe Pos 2 Positive	R	
0x1A18	0x6079	32	DC link circuit voltage	R	
0x1A19	0x60F4	32	Position Following error	R	
0x1A1A	0x60FA	32	Control Effort [cnt/sec]	R	
0x1A1B	0x60FC	32	Position Demand Value [cnt]	R	
0x1A1C	0x60FD	32	Digital Inputs	R	
0x1A1D	0x2205	16	Analog input	R	
0x1A1E	0x20A0	32	Auxiliary position actual value	R	
0x1A1F	0x6078	16	Current actual value	R	
0x1A20	0x60BD	32	Touch Probe Pos 2 Negative	R	
0x1A21	0x2085	16	Extra Status Register	R	
0x1A22	0x1002	32	ELMO Status Register	R	
0x1A23	0x2202	32	Extended Inputs	R	
0x1A24	0x2203	32	App Object	R	

Table 12: TxPDO Default Values

3.1.2. Emergency Requests

Emergency messages are trigger by the occurrence of a device internal error situation. The transmission execute via mailbox interface. An emergency object is transmitted only once per 'error event'.

CoE Header			Emergency			
Number	RSVD	Service	Error Code	Error Register	Data	RSVD

Frame Part	Data Field	Value / Description
CoE Header	Number	0x00
	RSVD	0x00
	Service	0x01 – Emergency
Emergency	Error Code	Error Code
	Error Register	Error Register
	Data	0000-9FFF: Manufacture Specific A000-EFFF: Diagnostic Data F000-FFFF: Manufacture Specific
	RSVD	

Table 13: Emergency Data Fields

3.2. Synchronization Modes

Elmo devices support two different modes; Master and Slaves for synchronization:

Free run	The Master cycle time and Slave cycle time are independent.
Synchronization with distributed clocks	The Master cycle time and Slave cycle time are synchronized.

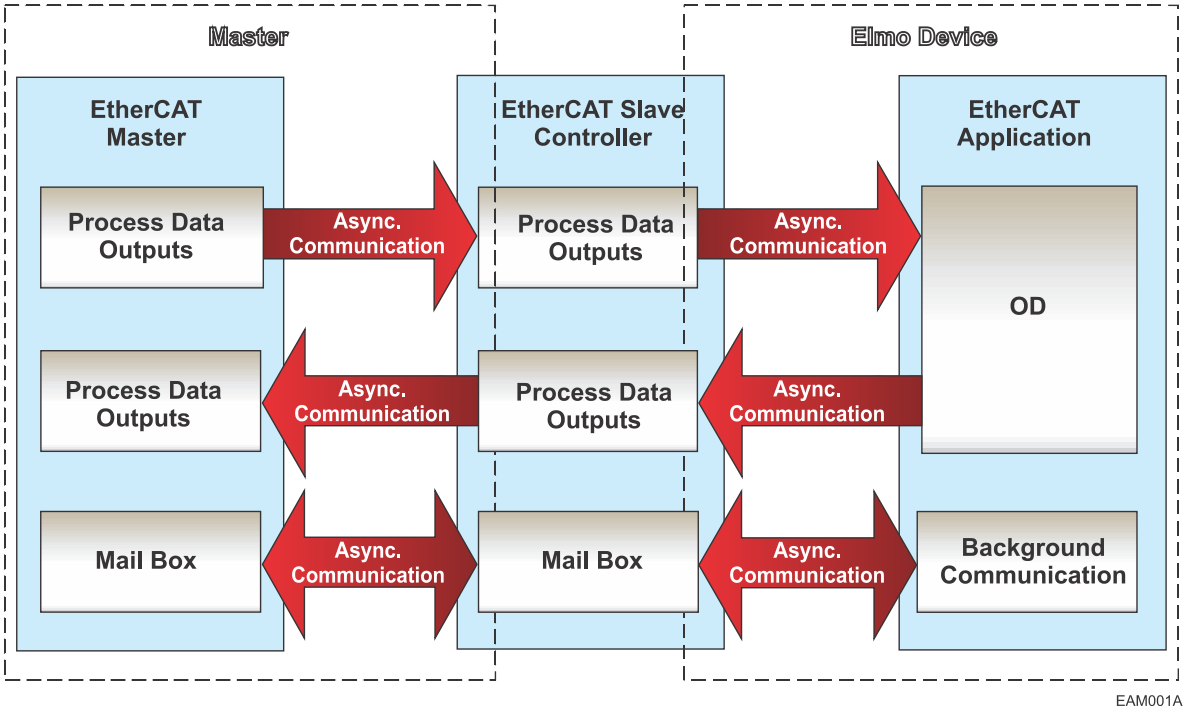


Figure 12: Frame Processing

3.2.1. Free Run

Configuration of the Master cycle time and the Slave cycle time are fully independent, and the slave's application is not synchronized to EtherCAT.

Each slave device reads/writes its own process data according to its local time, independent of the Master's cycle time.

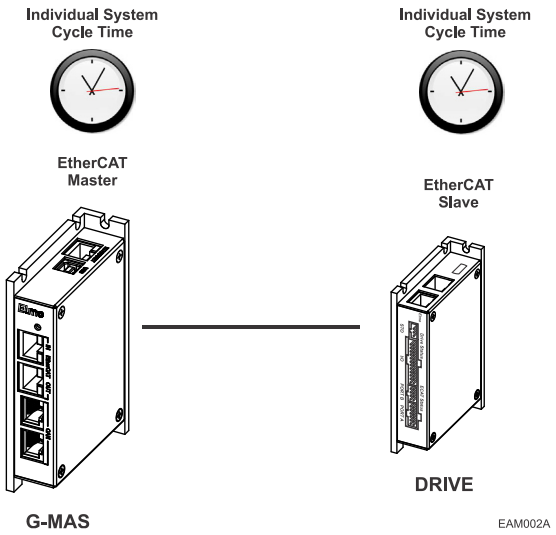


Figure 13: Master - Slave Cycle Times

The following diagram details the process timing in Free run mode:

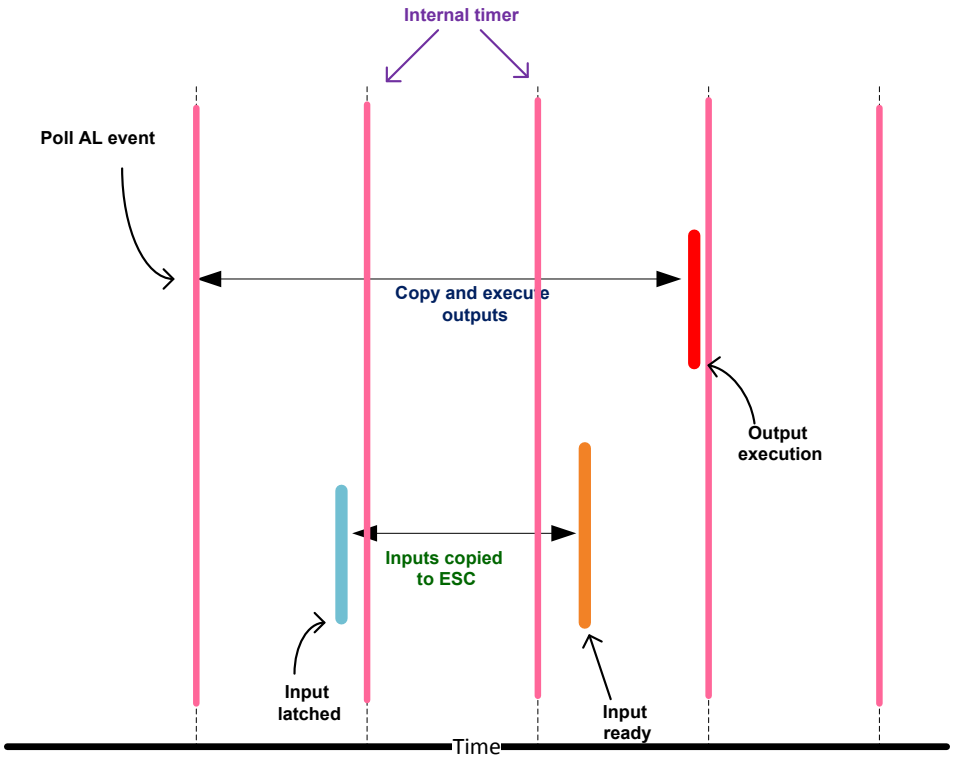


Figure 14: Frame Processing Free Run Mode

The above diagram is explained by the following definitions:

The **Poll AL event** is when the drive pools the AL event in its internal clock every 250µSec. In accordance with the AL indication, the outputs are **copied** to the relevant objects and **executed**. The **Input latched** are inputs which are latched every internal cycle and copied to the ESC. The **Outputs duration** from AL to execution is 750µSec, whereas the **Inputs duration** from latch time to ESC is 500µSec.

3.2.2. Distributed Clocks (DC Mode) - Synchronous with SYNC0

The Slave's application is synchronized to the SYNC0 event, which is based on the distributed clock unit to ensure that the time on all field bus devices is synchronized. Configuration of the Master and Slave cycle times are fully synchronized to the first Slave in the network that is used as a reference clock with the SYNC0 signal. This is then used to synchronize the slave clocks of the other devices and the controller.

The following diagrams define the internal synchronization in cases where the Distributed Clock (DC) is used. DC is set by the master and produce hardware interrupt (SYNC0) to the drive CPU. The drive synchronizes all motion and ECAT communication to this HW signal.

Inputs are latched relative the SYNC0 event where all signals beside the main position might jitter in few tens of uSec (<25uSec). The Actual Position (0x6064) is sampled on the SYNC0 with no jitter.

Outputs are been executed relative to the SYNC0 signal. The motion profiler is synchronized to the SYNC0. All other outputs might jitter in <100uSec.

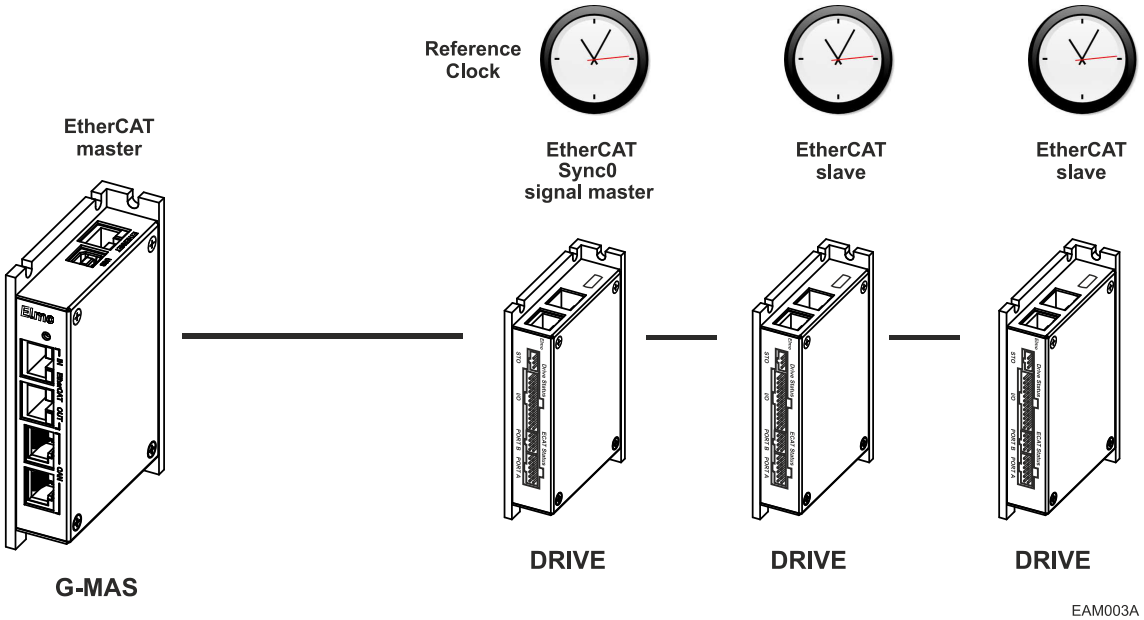


Figure 15: Internal Synchronization with Distributed Clock (DC)

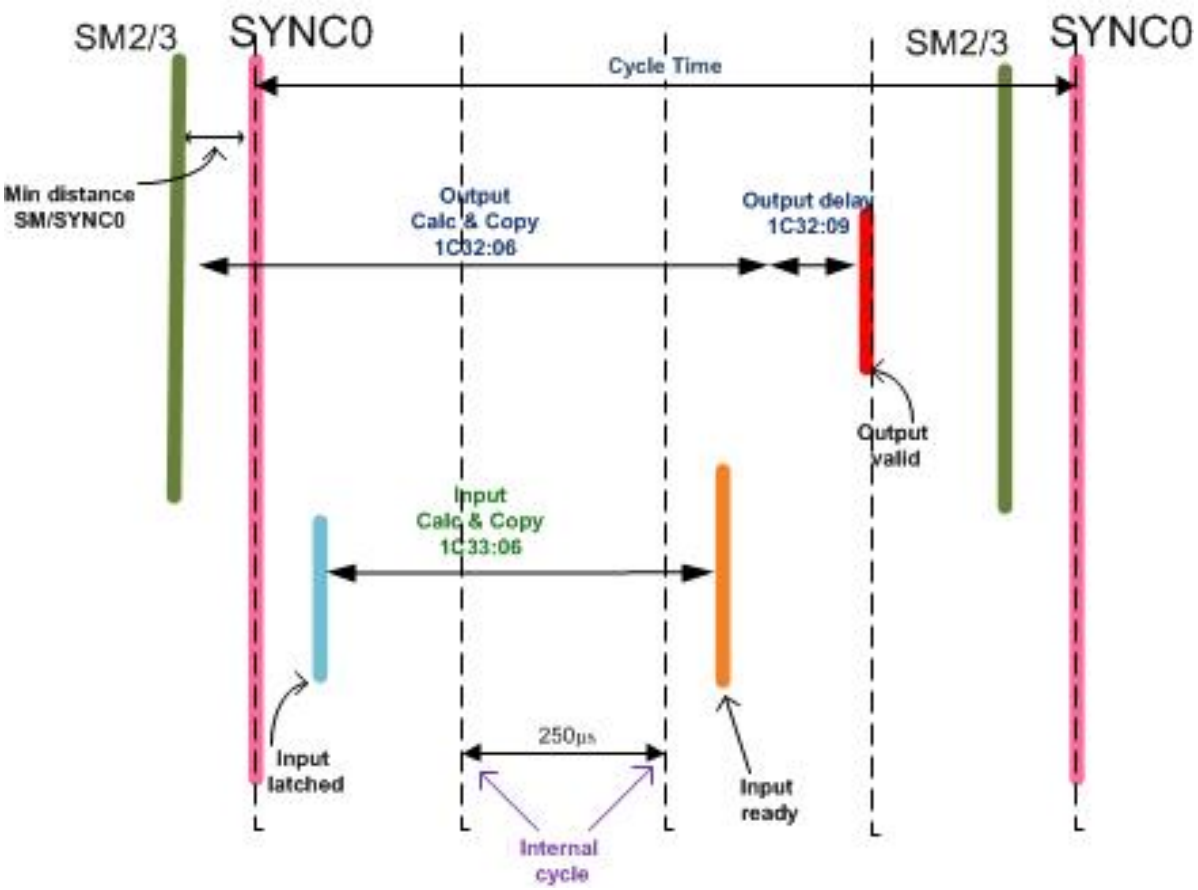


Figure 16: Frame Processing DC Mode Timing

The above Figure 13 details the DC mode timing, where the **Min distance SM/SYNC0** is the minimum time between SM2 arrival and the DC clock. Failing to attain this time may cause a delay in the output execution; typically 50uSec. The **Output Calc&Copy** is the time required to detect that AL arriving and obtain the relevant data from the ESC; up to 250uSec.

Output delay is defined as the time required for the application to perform the output, e.g. in cases of position set-point in cyclic synchronous position mode, this is the time from the set-point time interpolation until the actual torque as a result of this set point; up to 500uSec. The **Input delay** is the time from the SYNC0 signal until the actual inputs capture; 250uSec, and the **Inputs Calc&Copy** is the time taken to copy the inputs to the ESC.

All Slaves can be configured to a maximum of 250 µs (up to 4 KHz) and the Master cycle time should be a multiple of that value (e.g. 1x250, 2x250, 3x250… nx250).

Synchronization that is carried out due to ESC (EtherCAT Slave Controller) registers configuration. For more information, see Beckhoff's ET1100 Hardware Data Sheet (http://www.beckhoff.com/english.asp?download/ethercat_development_products.htm) Section II, 3.8 ESC Features (Version 1.8 obtainable from http://download.beckhoff.com/download/Document/EtherCAT/Development_products/EtherCAT_ET1100_Datasheet_all_v1i8.pdf).

3.2.3. No Distributed Clocks (Non-DC Mode) - Synchronous with SM2 event

In this mode, synchronization is performed on an SM2 event. Typically, SM2 and SM3 are transmitted by the master at the same instance. The jitter between SM2 transmissions should not exceed a few µSec.

The None-DC mode is been activated if SM2 is used and the drive is in EtherCAT OP state and no DC clock was initiated by the master.

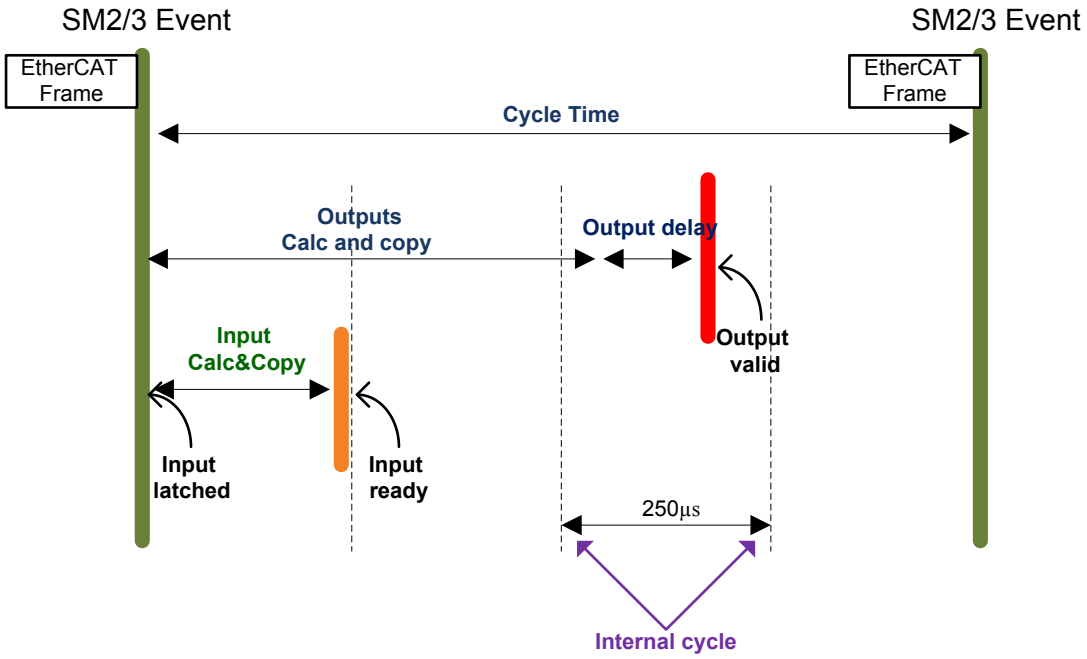


Figure 17: Frame Processing Non DC Mode Timing

Notes and limitations:

- In this mode the minimum cycle time is 1[msec].
- This mode depends on SM2/3 jitter of few µsec (up to 5 µsec).
- In case cycle time is smaller or equal to 1msec, there can be a delay in output processing up to 250 µsec.

3.3. EoE – Ethernet Over EtherCAT

The Ethernet Over EtherCAT (EoE) protocol enables the standard IP-based protocols TCP/IP and UDP/IP which are based on the higher level protocols HTTP and FTP, etc.

EoE can be used as layer 2 protocol (like an Ethernet switch) for transparently tunneling Ethernet frames over the EtherCAT communication.

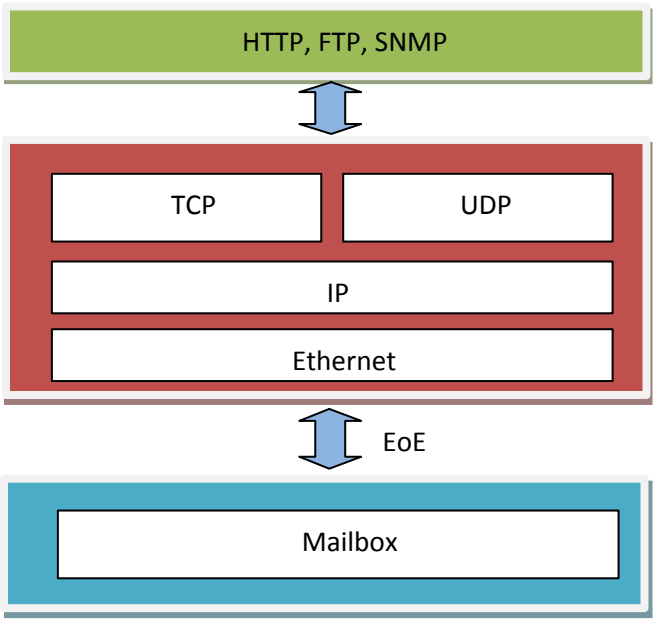


Figure 18: EoE Model

EoE has these services:

Initiate EoE	Used to transmit the first fragment of an Ethernet frame. It conveys the service specific parameters (such as MAC, IP, DNS, etc.) of the service request.
EoE fragment	Used to transmit the fragments of an Ethernet frame. This follows the Initiate EoE service, in case the Ethernet frame that is being transmitted is larger than the data parameter of the Initiate EoE service.
Set IP parameters	Used to transfer IP parameters from the client to the server. The server responds with the result of the Set operation. (The parameters are listed in section 5.3.2 – IP Address Info).
MAC Filter	Used if the EtherCAT device acts as bridge. Not currently supported.

3.3.1. EoE Extensions

3.3.1.1. MAC Address Info

Specify a virtual MAC address for the device according to ISO/IEC 8802-3. With this object the user can set or get the device's virtual MAC address.

Index	Sub-Index	Description	Data Type
0x1110	0	Virtual Ethernet MAC address	Unsigned 48

3.3.1.2. IP Address Info

With this object the user can set or get the device's virtual IP address.

Index	Sub-Index	Description	Data Type
0x1111	0	Number of entries	Unsigned 8
	1	IP Address	Unsigned 32
	2	Subnet Mask	Unsigned 32
	3	Default Gateway	Unsigned 32
	4	DNS Server	Unsigned 32
	5	DNS Name	String

3.4. FoE – File Access over EtherCAT

FoE specifies the standard method for downloading a firmware file from a Client machine (Master EtherCAT) to the Server (Slave) machine. This method uses the slave machine's mailbox interface to perform the firmware update. The FoE can also be used to update the PAL code in the drive.

The following services are defined within a FoE operation:

WRQ	Writing with an indication of the file name
PRQ	Reading with an indication of the file name
DATA	Data block
ACK	Acknowledgment of DATA and WRQ requests
ERR	Error notification
BUSY	Notification that the slave machine is currently busy

To transfer a new firmware version, the Slave initially must be set to "Boot" mode.

Notes

- Currently, only the write service is supported by the FoE operation. The service is mapped to mailbox transmission.
- The latest firmware files are located on Elmo's website.
- The Maestro allows downloading of both firmware and PAL to the drive or several drives, via the EAS configurator, using the FoE.

3.5. EEPROM

Each node in the EtherCAT network contains an EEPROM device. The EEPROM stores device configuration and device description information.

All information, such as device identity, application data, PDI control and configuration settings are stored at a specific address. The addresses range from 0x00 to 0x3F.

The startup process includes a step where the EtherCAT Slave Controller must acquire and load configuration data from the EEPROMs.

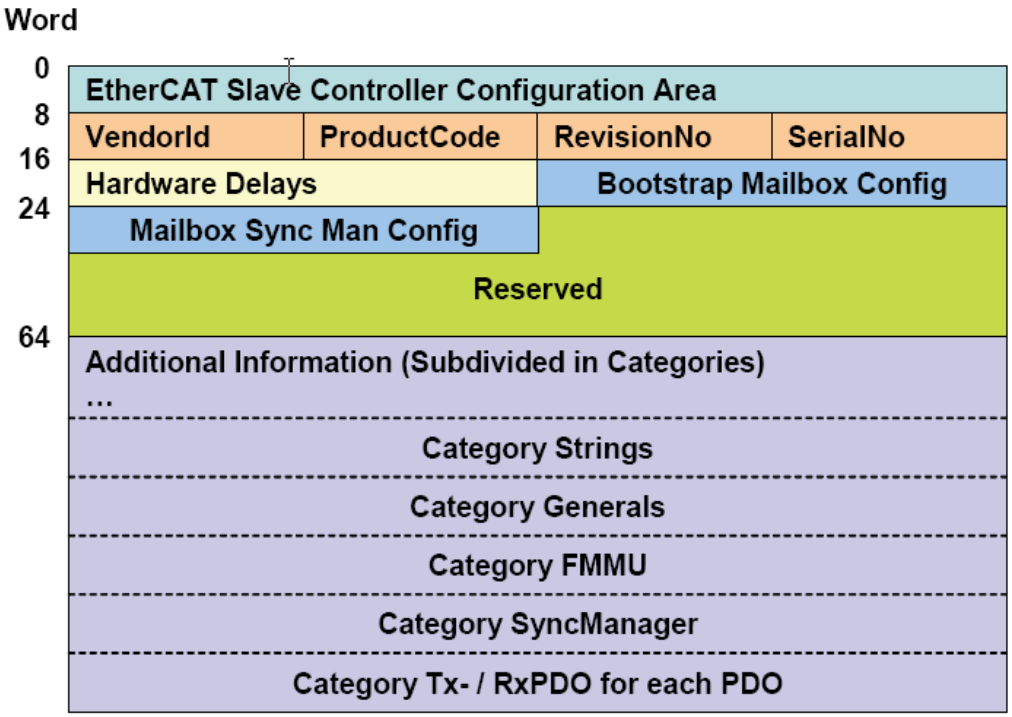


Figure 19: EEPROM Structure

Note: **Write Access is not supported.**

The information stored in the address range from word 0 to 63 (0x00 to 0x3F) is mandatory.

Word Address	Parameter	Word Address	Parameter
0x0	PDI Control	0x13	Reserved
0x1	PDI Configuration	0x14	Bootstrap Receive Mailbox Offset
0x2	Pulse Length of SYNC Signal	0x15	Bootstrap Receive Mailbox Size
0x3	Extended PDI Configuration	0x16	Bootstrap Send Mailbox Offset
0x4	Configure Station Alias	0x17	Bootstrap Send Mailbox Size
0x5 : 0x6	Reserved	0x18	Standard Receive Mailbox Offset
0x7	Checksum	0x19	Standard Receive Mailbox Size
0x8 : 0x9	Vendor ID	0x1A	Standard Send Mailbox Offset
0xA : 0xB	Product Code	0x1B	Standard Send Mailbox Size
0xC : 0xD	Revision Number	0x1C	Mailbox Protocol
0xE : 0xF	Serial Number	0x1D : 0x3D	Reserved
0x10	Execution Delay	0x3E	Size
0x11	Port0 Delay	0x3F	Version
0x12	Port1 Delay	0x40	…..

Table 14 : EEPROM Mandatory range memory

For a more detailed description of the EEPROM, refer to the “EtherCAT Slave Device Description” specification, which is available from the EtherCAT Technology Group at http://www.ethercat.org.

3.6. ESI (xml format)

ESI contains information about Elmo devices and their functionality. This information is as follows:

- Vendor: ID, Name, Logo
- Device Identity, Name, PDI type
- PDO mapping
- FMMU/SyncManger
- Group
- Types
- Etc.

Based on:

- EtherCATInfo.xsd
- EtherCATBase.xsd
- EtherCATDict.xsd

Notes

- The latest ESI file can be downloaded from Elmo's website at www.elmomc.com.
- Modification of the ESI file is prohibited.

Chapter 4: Maestro Communication

This chapter describes how to set up the Gold and Platinum Maestro and drives using EtherCAT communication. The setup applies to a configuration where the Maestro is not connected, but can equally apply to a Maestro which is connected.

4.1. EASII EtherCAT Quick Configuration

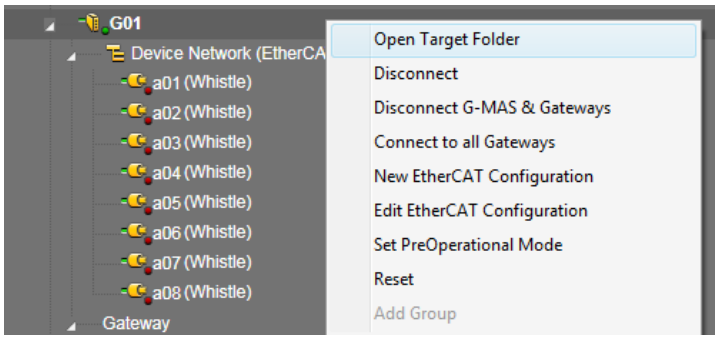
This EtherCAT configuration allows the user to quickly setup the system for EtherCAT communication without have to configure individual drives. This setup assumes the Maestro and /or Drives and connected and are visually seen in the Workspace by the EAS application.

To perform a quick EtherCAT configuration:

Important: If you have upgraded the Gold Maestro (not relevant to Platinum Maestro) to version 1.1.5.0, then you must use the EASII application version 2.1.1.x, and in addition, reconfigure the EtherCAT network.

1. Right-click the EtherCAT Platinum Maestro/Gold Maestro device in the workspace tree and select **New EtherCAT Configuration** from the drop-down menu.

   If you only want to edit the EtherCAT configuration that was already created, select **Edit EtherCAT Configuration** from the drop-down menu.



   Or, alternatively,

   In the **System Configuration** activity, go to the **System Configuration** ribbon and click the **New EtherCAT Configuration** button to create a new EtherCAT configuration or click the **Edit EtherCAT Configuration** button to edit an EtherCAT configuration that was already created.

   If you go into the EtherCAT Configuration tool using the **Edit** button, you must click the **Start and Download** button in order to create a new configuration for this Platinum Maestro/Gold Maestro.

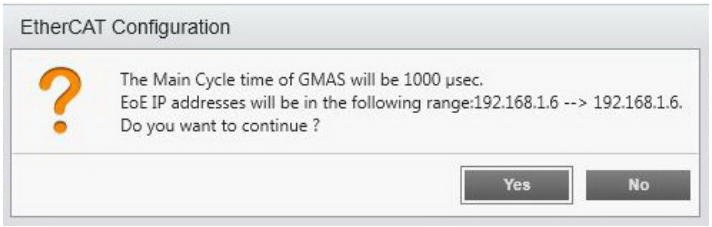
Note: When editing an EtherCAT configuration, and EASII detects an inconsistency between the Workspace Resource file and the Resource file located on the Platinum Maestro/Gold Maestro, the G-MAS Resource window is displayed.

EASII compares the number, type, and revision of devices in your EtherCAT network to those in the current Platinum Maestro/Gold Maestro configuration. If a mismatch is found, the following dialog box appears displaying the mismatched revision details:

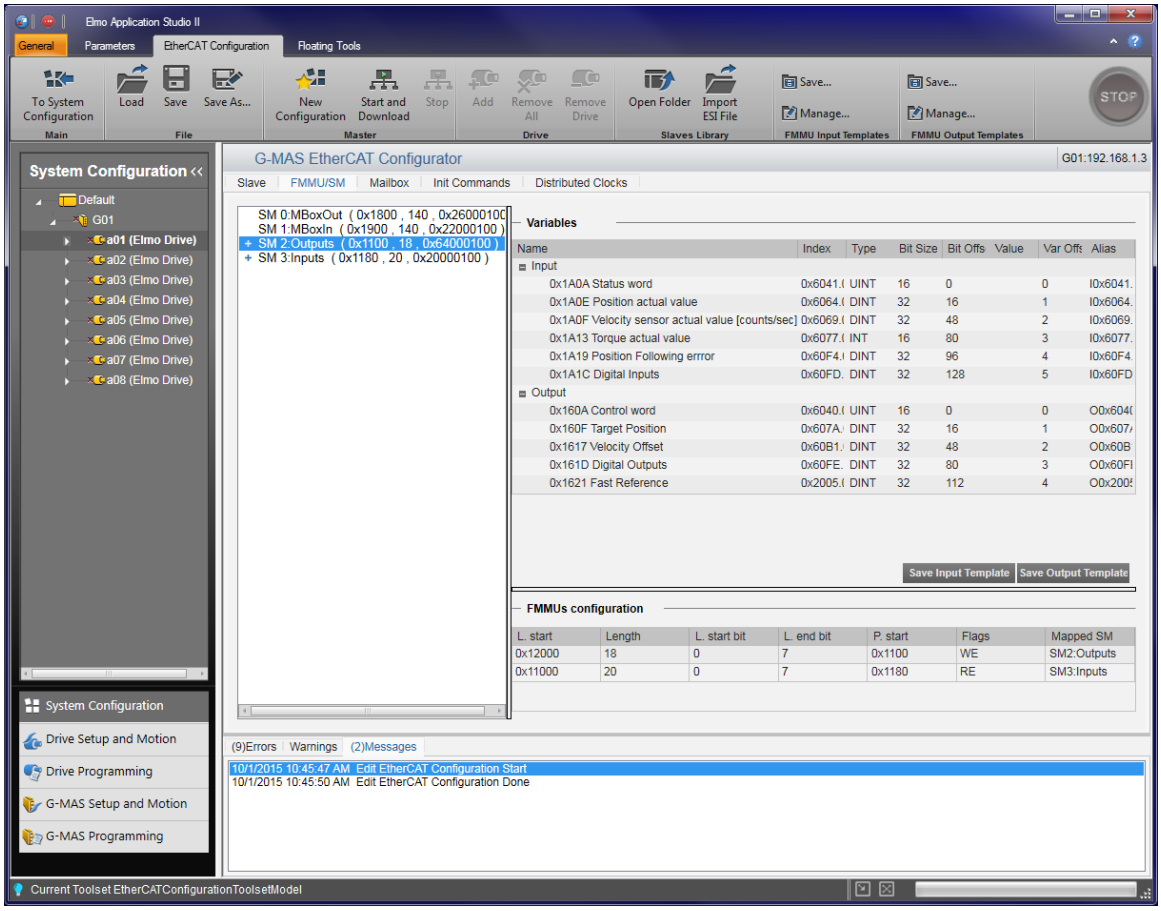
- Click **Keep Current Configuration** to keep the current configuration
- Click **Accept Scan Results** to change the configuration.

Note: After clicking the **New EtherCAT Configuration** button EASII creates a new EtherCAT configuration for this Platinum Maestro/Gold Maestro without asking the user. Hence, the Compare Configuration dialog does not appear and the current configuration is overridden.

The EtherCAT Configuration window appears prompting you if you want to continue using the displayed EoE IP address range.



2. Click **Yes**. The EtherCAT Configuration window is displayed.
3. Select a Platinum Maestro/Gold Maestro EtherCAT slave.
4. Click on the **FMMU/SM** tab.



5. Click the output name's corresponding checkbox in the left pane of the **FMMU/SM** tab to add it to the **Variables Output** list on the right pane of the **FMMU/SM** tab.
6. Click the **Save Output Template** button on the lower part of the **Variables** pane.

   The Save Output Template dialog box is displayed.
7. Save the file as an output template file (***.tmo** format).

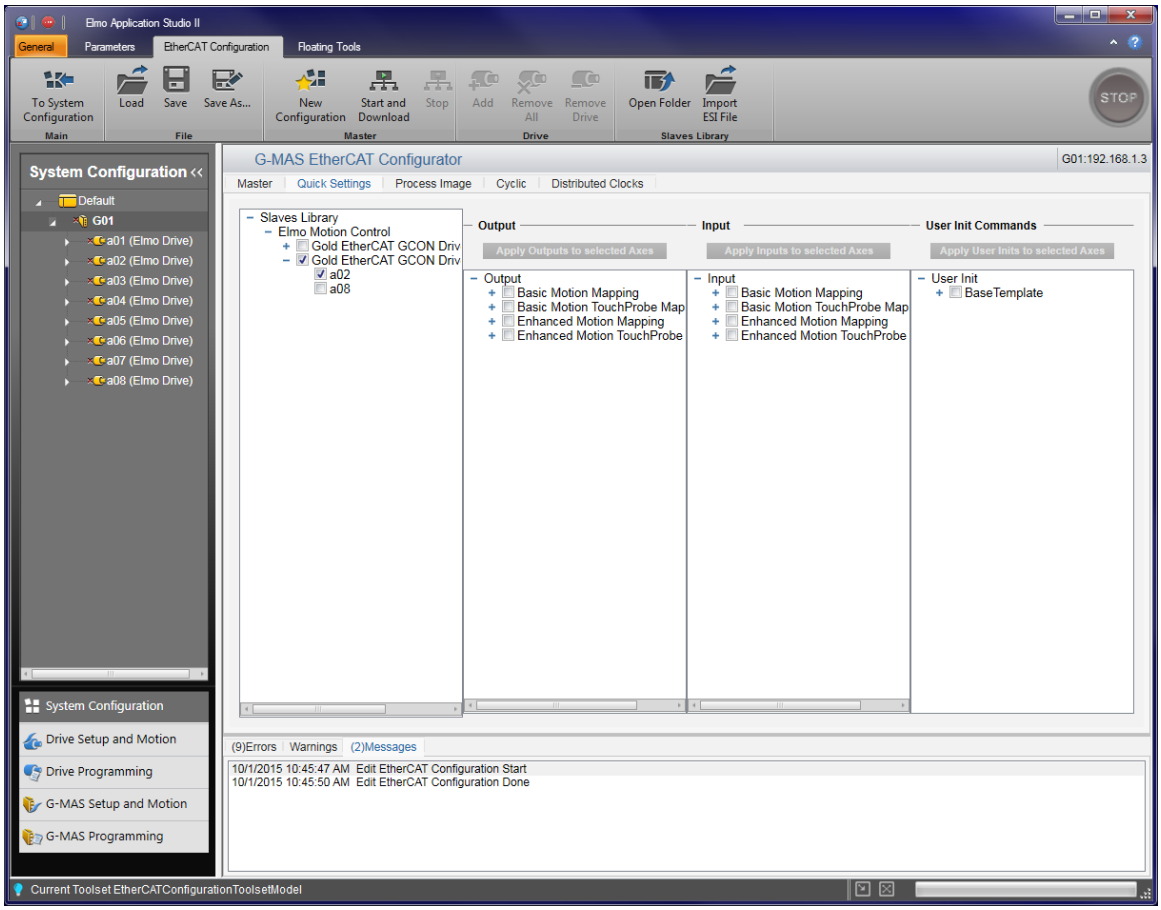
   The selected outputs are now saved as a template.
8. Scroll to the lower part of the left pane then click the input name's corresponding checkbox in the left pane of the **FMMU/SM** tab to add it to the **Variables Input** list on the right of the **FMMU/SM** tab.
9. Click the **Save Input Template** button on the lower part of the **Variables** pane.

   The Save Input Template dialog box is displayed.
10. Save the file as an input template file (***.tmi** format).

    The selected inputs are now saved as a template.

11. Click the Master (EtherCAT Platinum Maestro/Gold Maestro) device in the workspace tree.

12. Click the **Quick Settings** tab.



13. In the **Quick Settings** tab, select the slave(s) to implement the new configuration by clicking on the corresponding checkbox(es).

14. Select the output you want to add to the selected slave(s) by clicking on the output name's corresponding checkbox in the **Output** list.

    The **Apply Outputs to selected Axes** button becomes active.

15. Click the Apply Outputs to selected Axes button.

16. Select the input you want to add to the selected slave(s) by clicking on the input name's corresponding checkbox in the **Input** list.

    The **Apply Inputs to selected Axes** button becomes active.

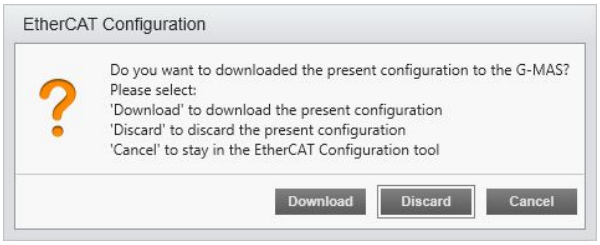
17. Click the Apply Inputs to selected Axes button.

18. Click the  **Start and Download** button from the **Master** group in the **EtherCAT Configuration** ribbon.

    The new ETherCAT configuration is downloaded to the device(s).

19. In the **ETherCAT Configuration** ribbon, click the **To System Configuration** button from the **Main** group to go back to the system configuration.

Note: If you choose to go back to the configuration system without downloading the new EtherCAT configuration, the following message appears:

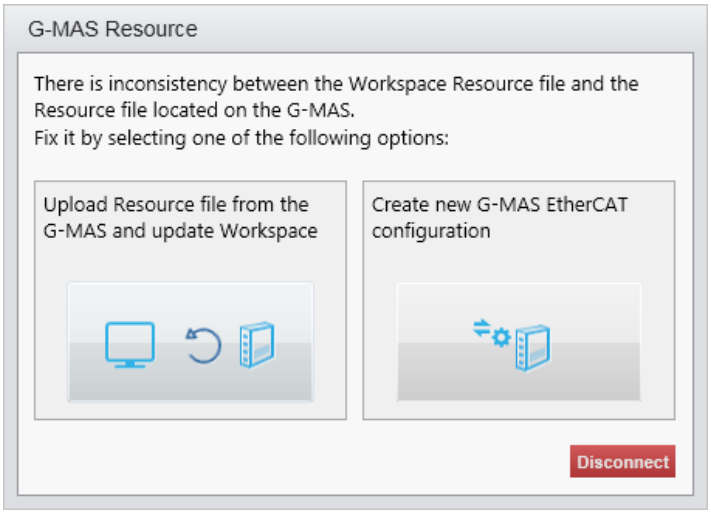


- Click **Download** to download the configuration and exit the EtherCAT Configuration tool.
- Click **Discard** to exit the EtherCAT Configuration tool without downloading the new configuration.
- Click **Cancel** to go back to the EtherCAT configuration tool.

4.1.1. Connecting to the Platinum Maestro/Gold Maestro

If there is an inconsistency between the Workspace Resource file and the Resource file located on the Platinum Maestro/Gold Maestro when connecting to the Platinum Maestro/Gold Maestro, EASII prompts you to resolve the inconsistency using the G-MAS Resource window.

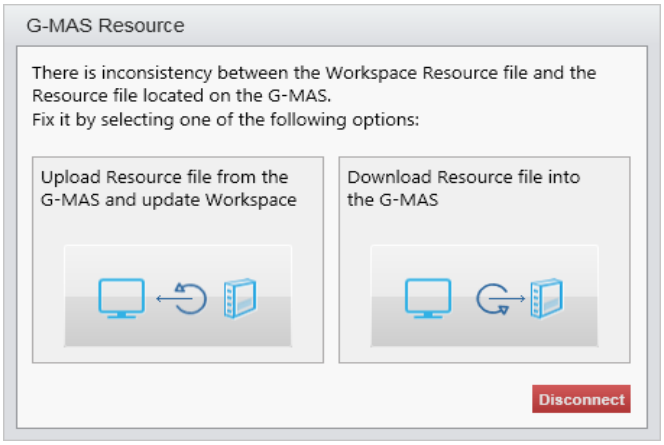
- If you are connecting to an EtherCAT Platinum Maestro/Gold Maestro for the first time, the following G-MAS Resource window appears:



Select one of the following options:

- Upload the resource file from the Platinum Maestro/Gold Maestro and update the workspace.
- Create a new Platinum Maestro/Gold Maestro configuration file.
- Click the **Disconnect** Disconnect button to close the window without establishing a connection.

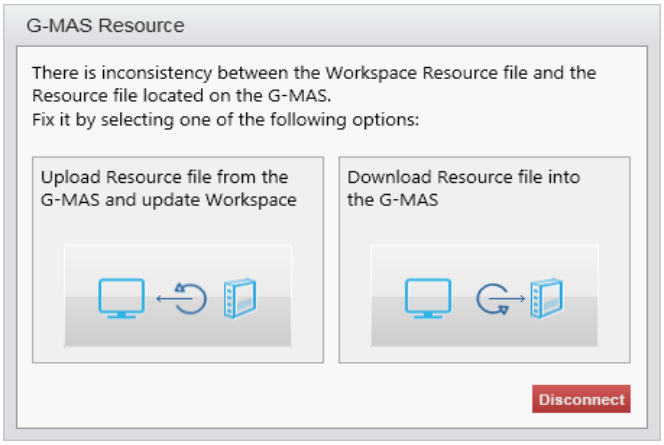
If EASII detects a change in the Resource file, it displays the following G-MAS Resource window:



Select one of the following options:

- Upload the resource file from the Platinum Maestro/Gold Maestro and update the workspace.
- Download the Resource file into the Platinum Maestro/Gold Maestro.
- Click the **Disconnect** Disconnect button to close the window without establishing a connection.

- If you are connecting to a CAN Platinum Maestro/Gold Maestro either for the first time or if EASII detects a change in the configuration file, only the following G-MAS Resource window appears:



Select one of the following options:

- Upload the resource file from the Platinum Maestro/Gold Maestro and update the workspace.
- Download the resource file into the Platinum Maestro/Gold Maestro.
- Click the **Disconnect** Disconnect button to close the window without establishing a connection.

4.2. Maestro EoE Configuration

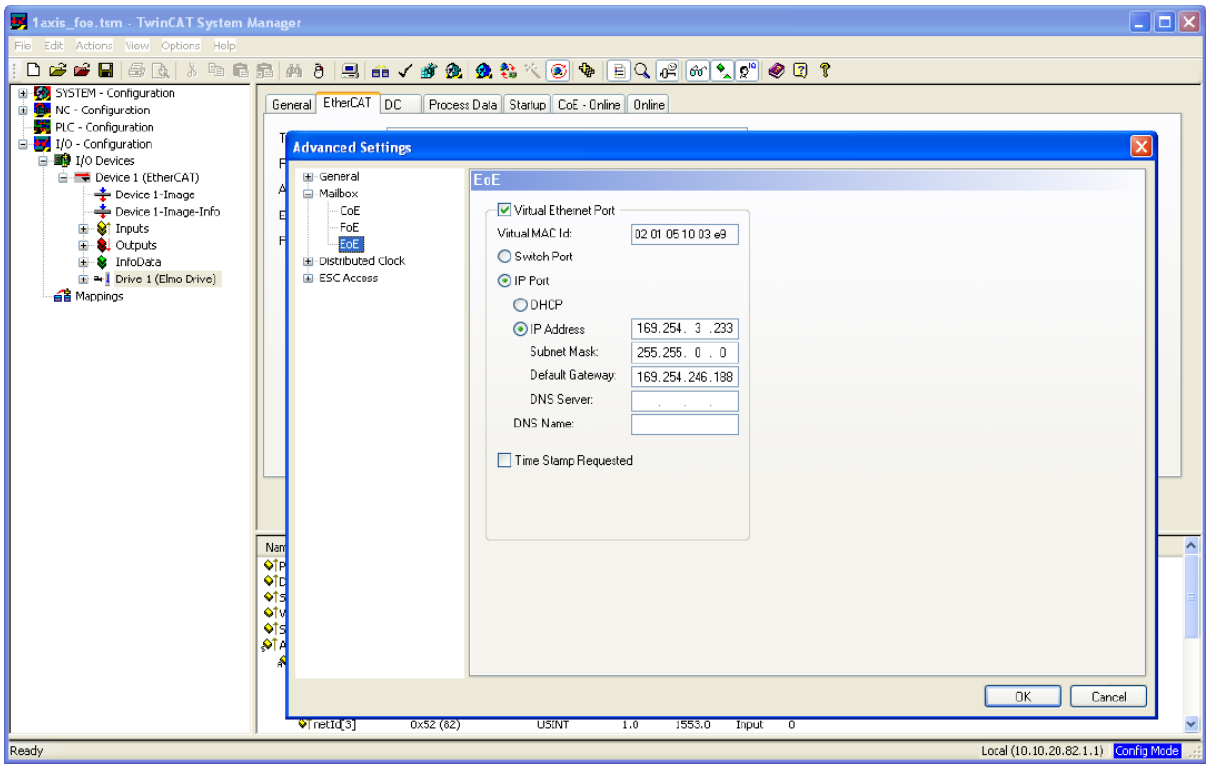
The following instructions demonstrate how to configure EAS application to Elmo drives using EoE protocol. Before starting make sure, the drive connected to a network card (section 4.2) or to a switch port (section 4.3).

Note: **This example assuming there is no switch port connected.**

1. Setup the Drive IP address:

   a. Using the TwinCAT application, open drive advanced setting:

   Device1 -> Drive1->EtherCAT ->Advanced Setting->Mailbox->EoE



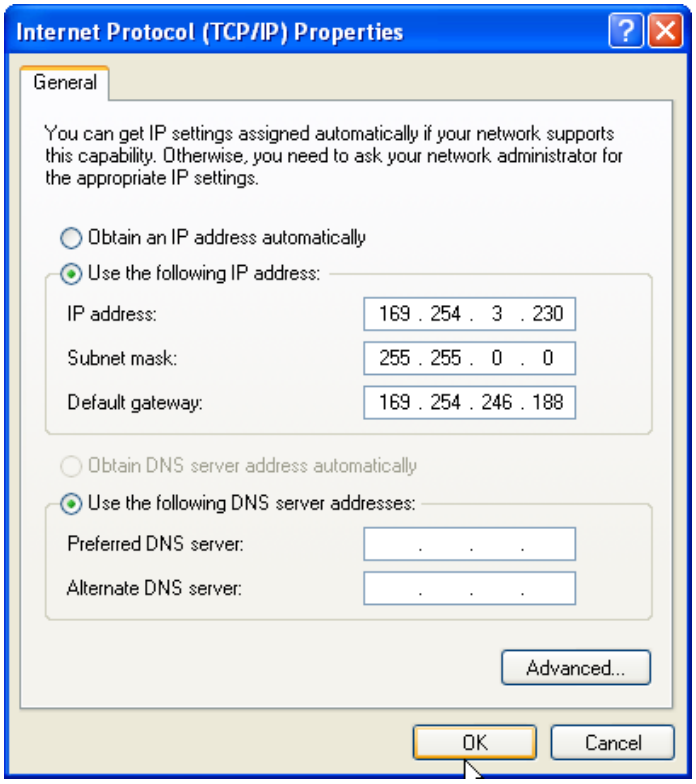
   b. Select IP Port and IP Address (radio buttons), TSM automatically update the IP address, Subnet Mask and Default Gateway. Click **OK**.

2. Configure the PC Network Card:

   a. To enable the network card using EoE protocol, the network card must configured as follows:

      i. From the Windows Control Panel, select **Network Connections**.
      ii. Right click the relevant network and select **Properties**.
      iii. In the list of items, select **Internet Protocol (TCP/IP).** You may need to scroll down.
      iv. Click **Properties**.

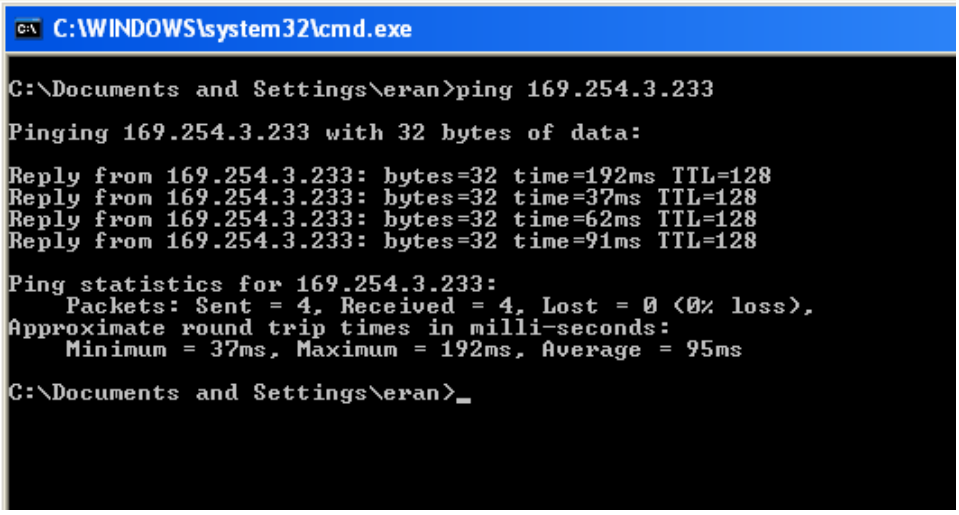
v. Select **Use the following IP address**, and enter the new IP address, Subnet mask and Default gateway for example:



vi. Click **OK**.

b. Return to TSM and reload the devices (activate configuration).

3. Before setting the EAS application, open a command prompt window and using the Ping command check for a reply from the drive.

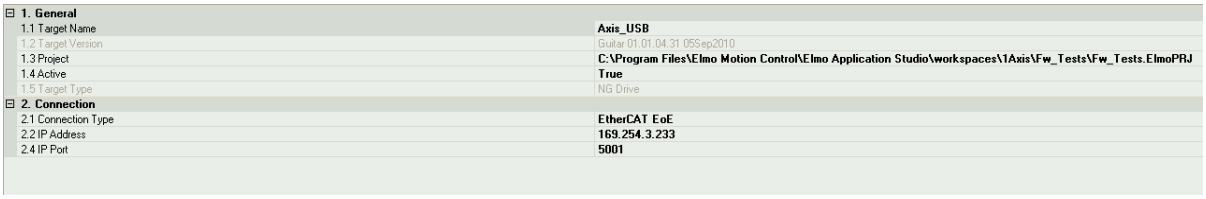


4. Now open the EAS application and configure the Drive communication as follow:

a. Connection Type : EtherCAT EoE

b. IP Address: Same IP address as configure in TwinCAT application (see Section 4.4.1).

c. IP Port: 5001 (Default)



5. Press **Connect**.

Chapter 5: TwinCAT Communication

5.1. Architecture

Due to flexibility of EoE there is more than one way to configure an EtherCAT network that includes EoE devices. Each configuration uses the EAS for downloading parameters, axis tuning, recorder, Ethernet communication, etc.

5.2. Using TwinCAT Master

In this configuration, the Master EtherCAT device and EAS run on IPC using the same Ethernet port. EoE communication goes directly from the Master EtherCAT device to the network device (i.e., a switch port is not required).

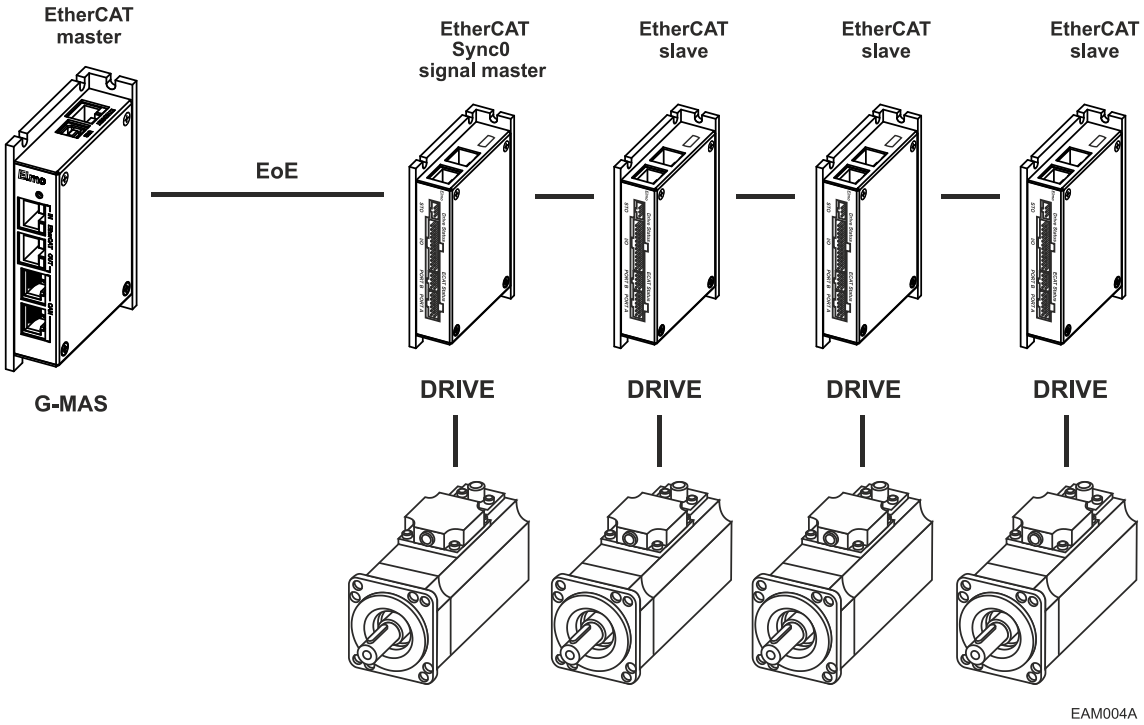


Figure 20: TwinCAT and EAS EoE Configuration

5.3. Using a Switch Port

In this configuration, the EoE communication goes via a switch port to and from the device using the Ethernet protocol.

EAS (Elmo Application Studio) runs on a remote PC using the switch port terminal. It can run all the EoE services (download parameters, recording tuning, etc.).

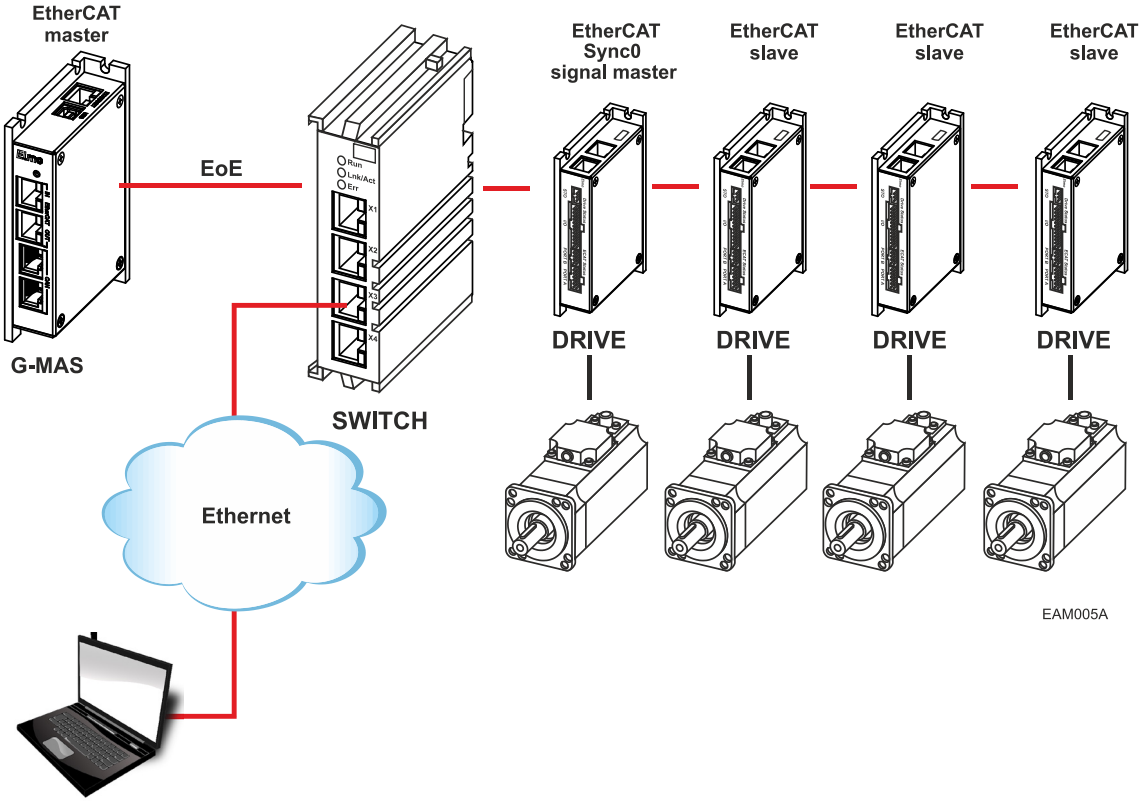


Figure 21: TwinCAT and EAS EoE Configuration Using a Switch Port

5.4. Download Firmware using FoE (via TwinCAT)

5.4.1. Setup Using the TwinCAT NC/PTP System Manager

Before starting the drive setup procedures, the following must be complete:

- The drive has been configured by the EAS and the servo motor can move.
- TwinCAT software from Beckhoff (NC/PTP mode setup) is installed.
- The latest ESI (xml format) file is available and has been copied to the TwinCAT slave library (usually found in C:\TwinCAT\IO\EtherCAT). It is then possible to scan the network for new devices.
- The drive is connected to an EtherCAT Master PC/IPC.
- The TwinCAT system manager resides in configuration mode.
- The TwinCAT system manager must be configured with a network card.

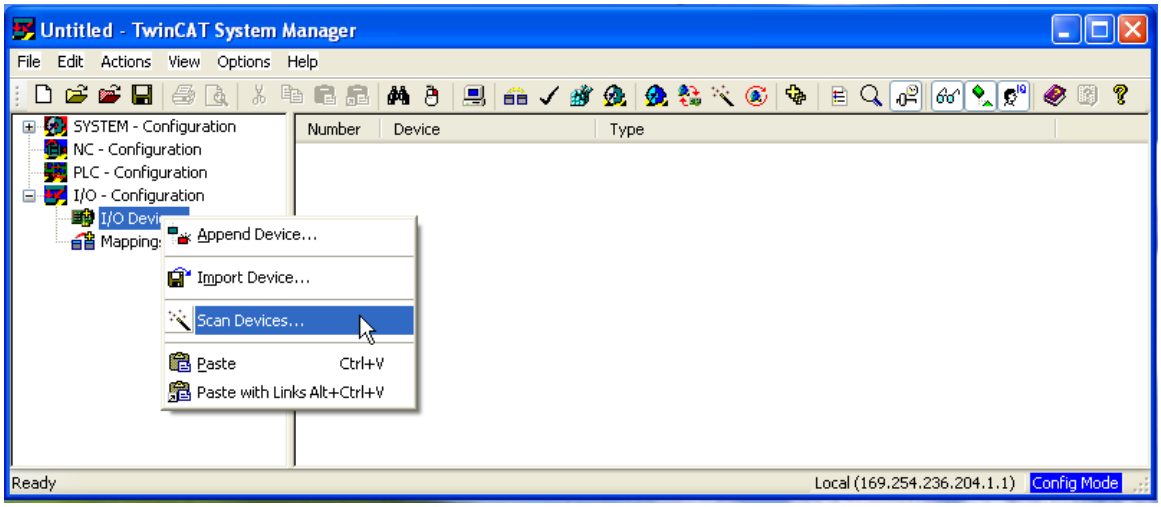
5.4.2. Setup Procedure

The following example demonstrates the automatic EtherCAT network setup procedure. Network setup may also be performed manually – refer to the TwinCAT manual for more information.

1. Scan Devices:

   a. Ensure the EtherCAT network is fully connected, and the power is switched ON.

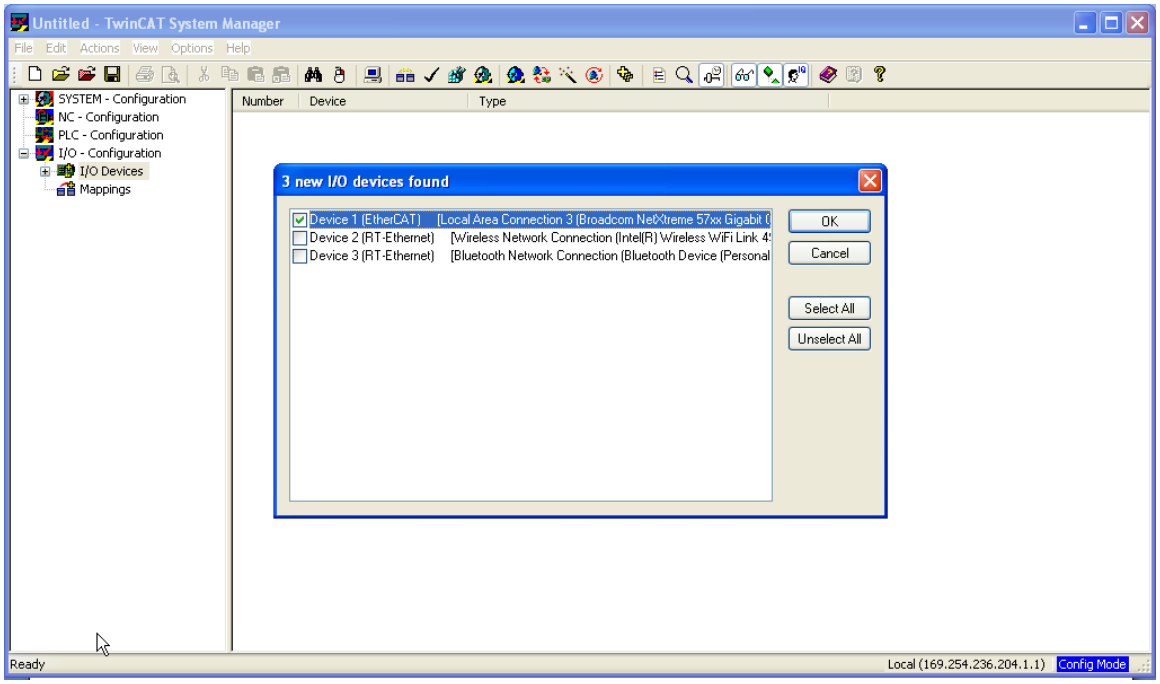
   b. Create a new project, then right-click and select I/O Devices. Choose **Scan Devices** from the menu. The scan is performed.



   A pop-up window informs you that not all devices can be detected by the TwinCAT software. Click **OK** to continue.

2. Select New Devices

   An EtherCAT card is found during the scan. Select the EtherCAT checkbox, and then click **OK** to continue.

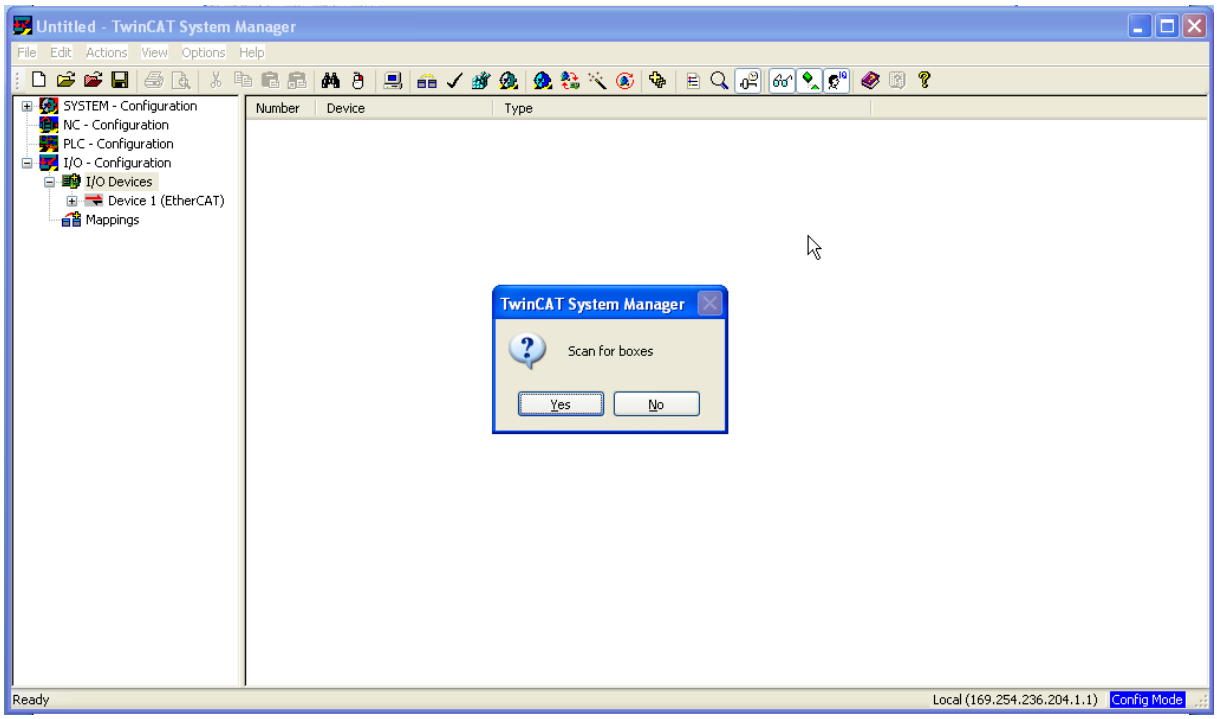


If more than one Ethernet devices is found, select the EtherCAT device only and click **OK** to continue.

If no EtherCAT card is found during the scan, verify that the EtherCAT slave is connected to the correct EtherCAT network that was already installed. Otherwise, TwinCAT will find a real time Ethernet card.

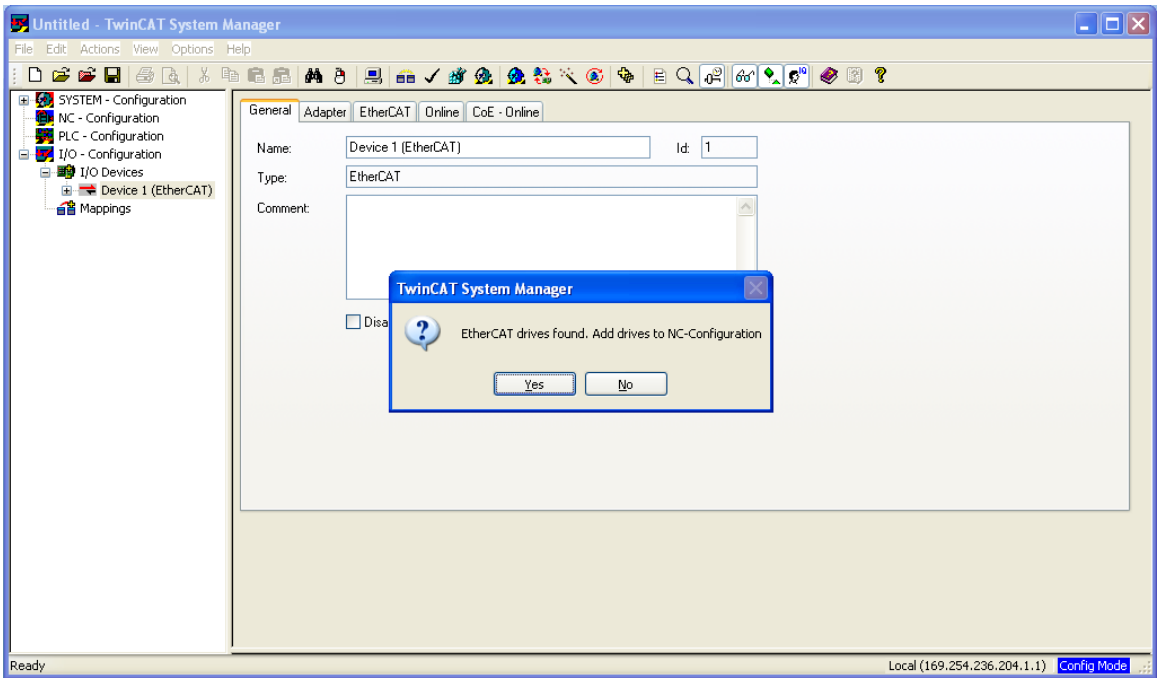
3. Scan for Boxes

To allow TwinCAT to scan for boxes, click **Yes** in the pop-up window that appears.



4. Adding to the NC Task

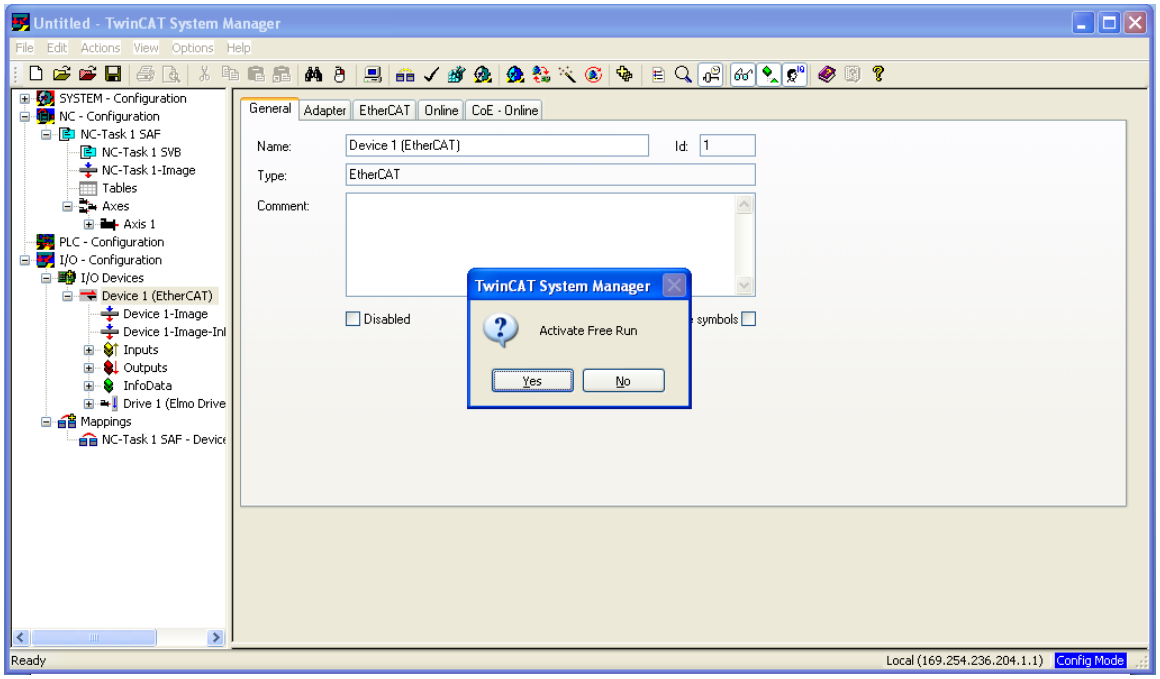
After the Box scan is complete, a pop-up window appears. The pop-up window requests that the device in question be added to the NC – Configuration.
Click **Yes**, TwinCAT adds the device to the NC – Configuration.



5. Activate the TwinCAT Configuration

   After TwinCAT adds the device to the NC – Configuration, a pop-up window appears. Click **Yes** to stay in the EtherCAT system manager's configuration mode. The TwinCAT system manager automatically changes to an OP state.

   Otherwise, click **No** to automatically change the TwinCAT system manager to a PO state and continue to the next step.



6. Enable NC Configuration
7. Enable Axis

To define the download process for new firmware files to Elmo's drive, use the FoE communication link. Each process starts with a user request while the drive is in a BOOT state Each process finishes after all files are sent to the drive and the Master machine receives an ACK for the last packet.

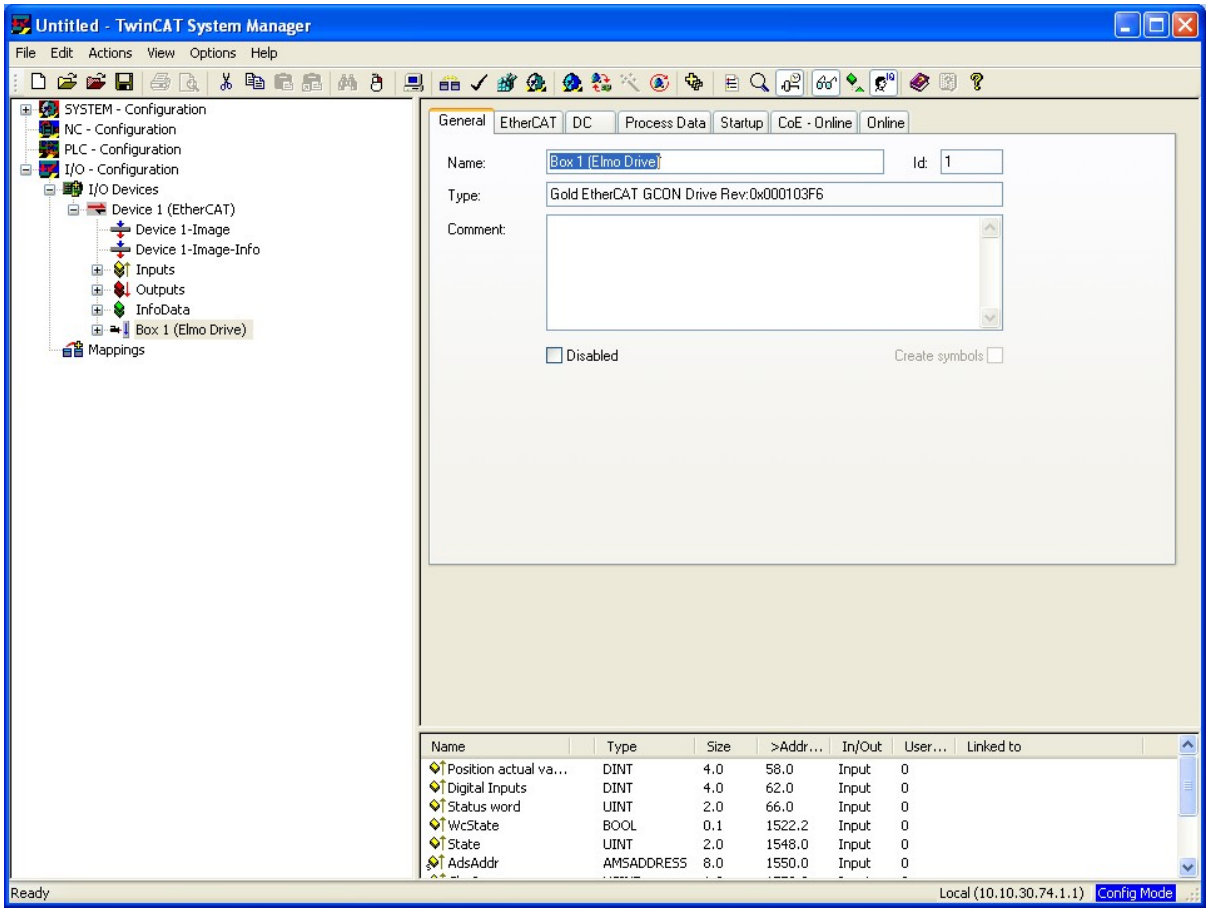
Only FoE communications are handled by the Master machine during a firmware download.

The following steps must be executed in order to update the Elmo devices' firmware.

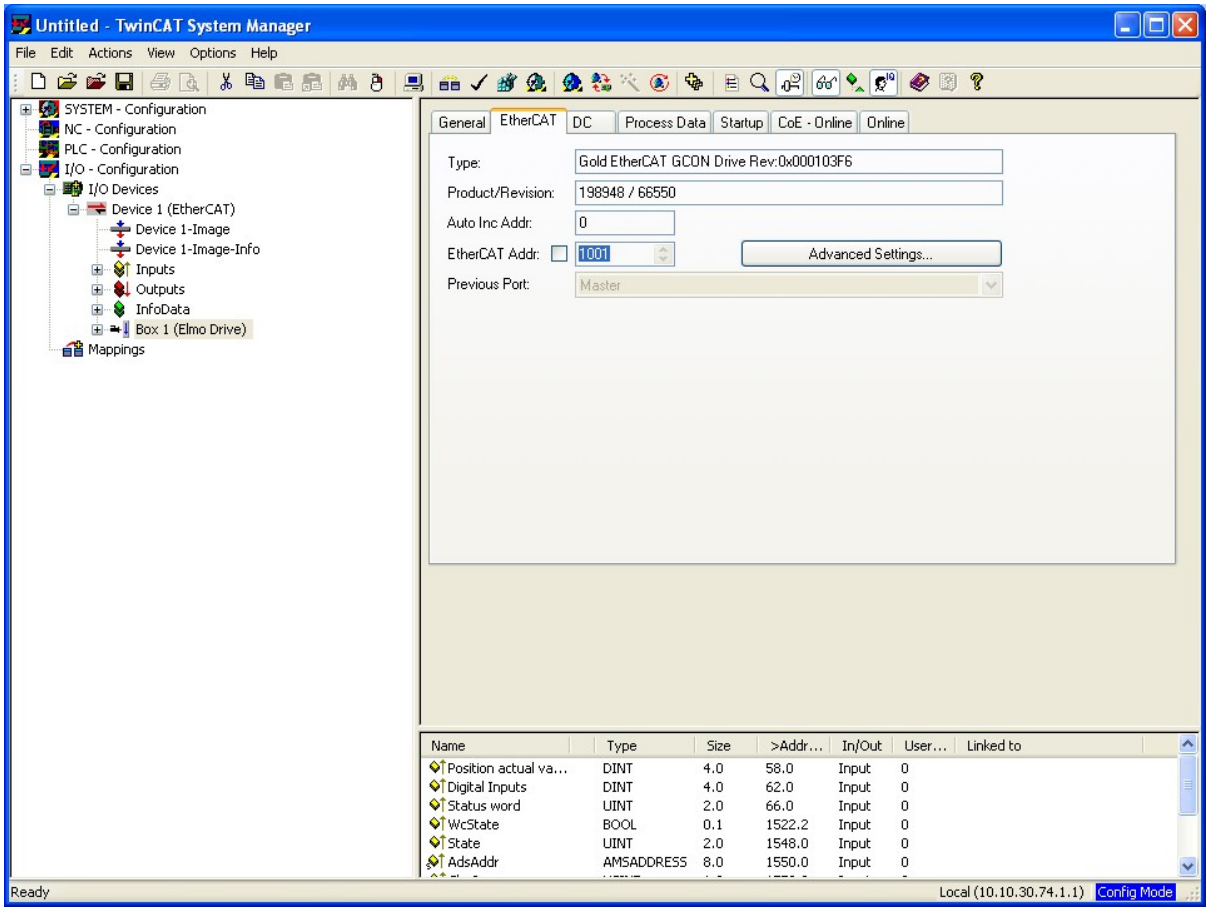
- The USB cable is disconnected.
- All user programs must be closed.
- The mailbox cycle time is configured to 50 msec by default.

To update the firmware

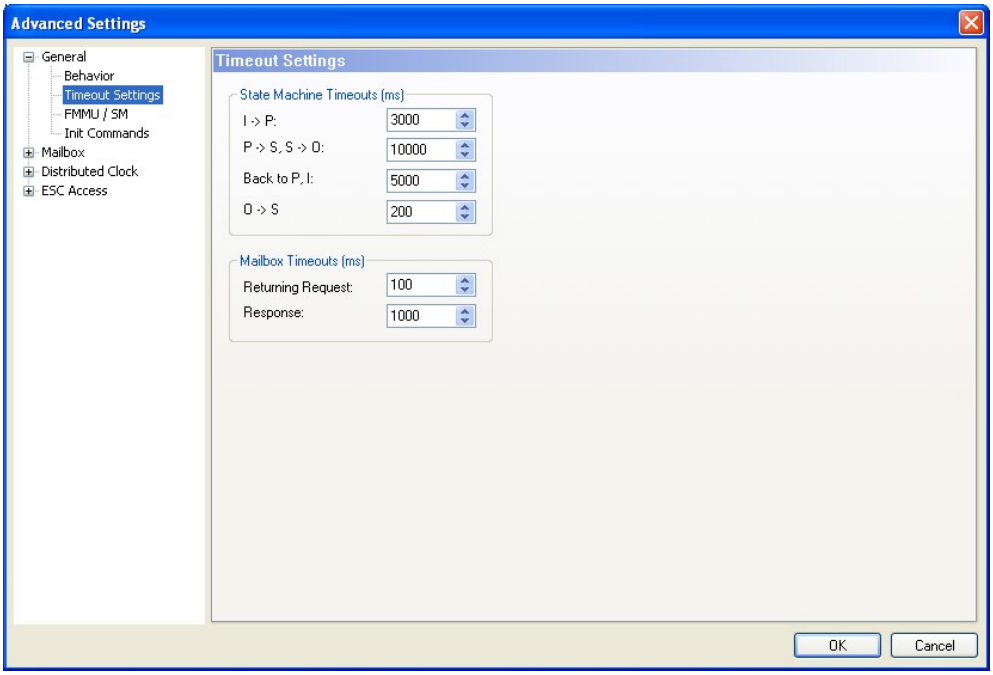
1. Perform the procedural steps described in section 4.4.1 Setup Using the TwinCAT NC/PTP System Manager on page 53 to configure the devices and Boxes attached to the I/O Configuration.


Untitled - TwinCAT System Manager
File Edit Actions View Options Help
SYSTEM - Configuration
NC - Configuration
PLC - Configuration
I/O - Configuration
I/O Devices
Device 1 (EtherCAT)
Device 1-Image
Device 1-Image-Info
Inputs
Outputs
InfoData
Box 1 (Elmo Drive)
Mappings
General EtherCAT DC Process Data Startup CoE - Online Online
Name: Box 1 (Elmo Drive) Id: 1
Type: Gold EtherCAT GCON Drive Rev:0x000103F6
Comment:
Disabled
Create symbols
Name Type Size >Addr... In/Out User... Linked to
Position actual va... DINT 4.0 58.0 Input 0
Digital Inputs DINT 4.0 62.0 Input 0
Status word UINT 2.0 66.0 Input 0
WcState BOOL 0.1 1522.2 Input 0
State UINT 2.0 1548.0 Input 0
AdsAddr AMSADDRESS 8.0 1550.0 Input 0
Ready
Local (10.10.30.74.1.1)
Config Mode

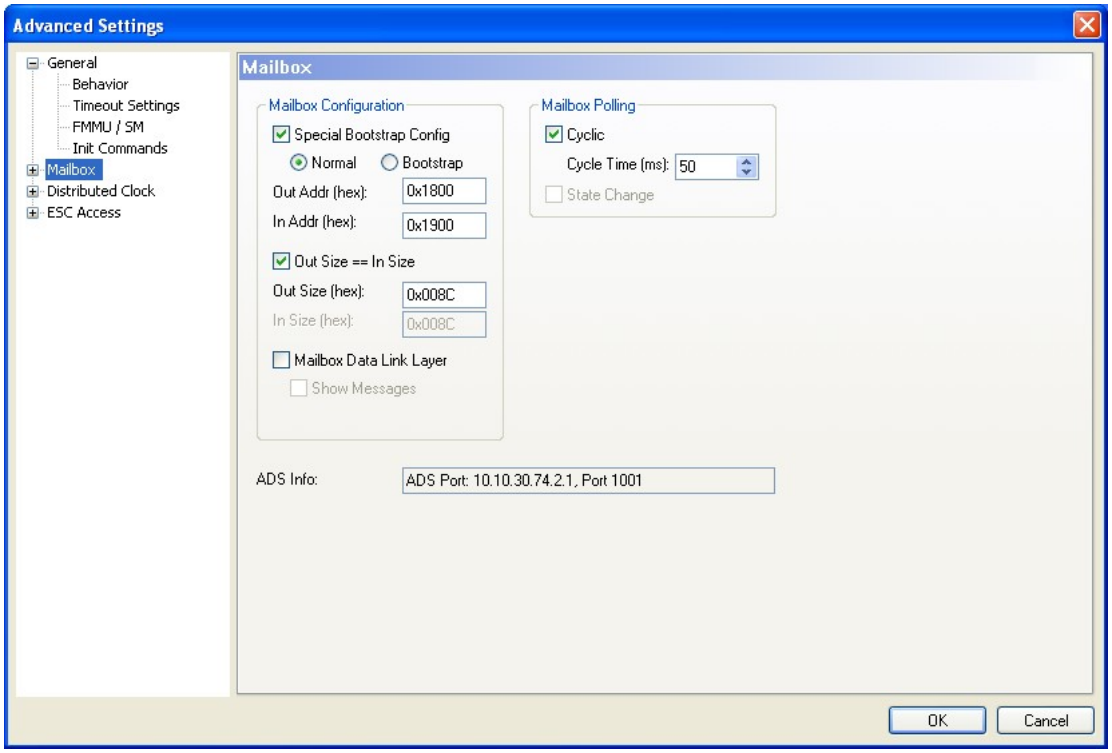
2. Select the EtherCAT tab, and click **Advanced Settings**.



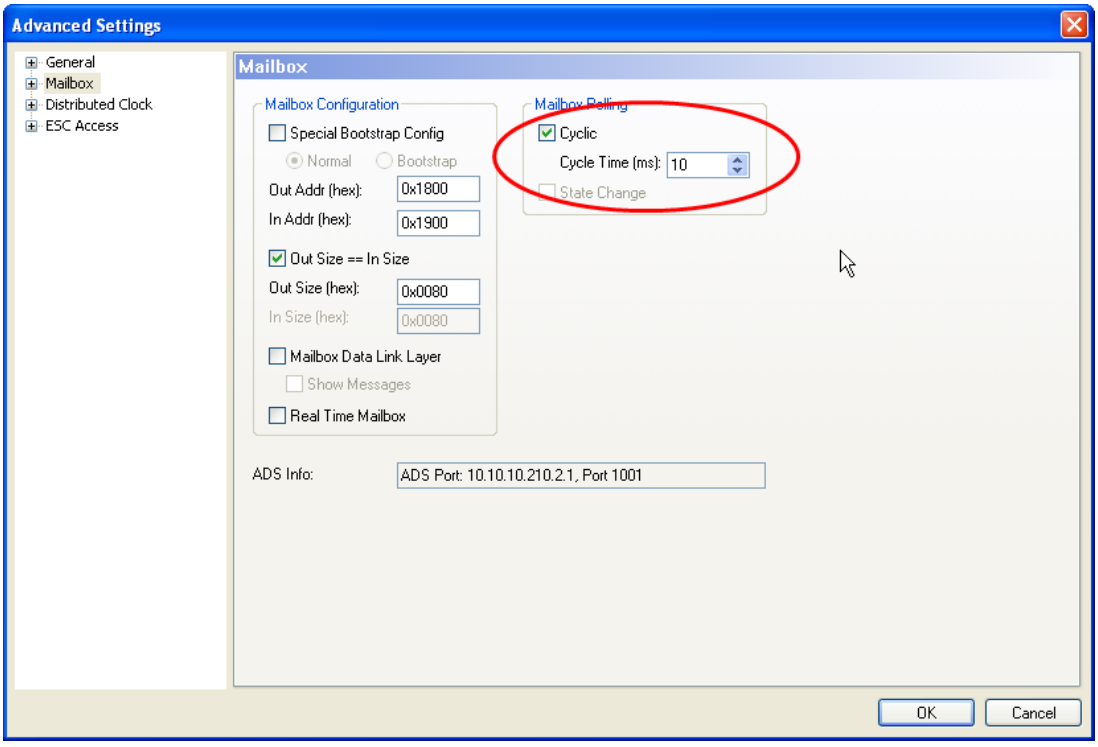
The Advance Settings window opens.



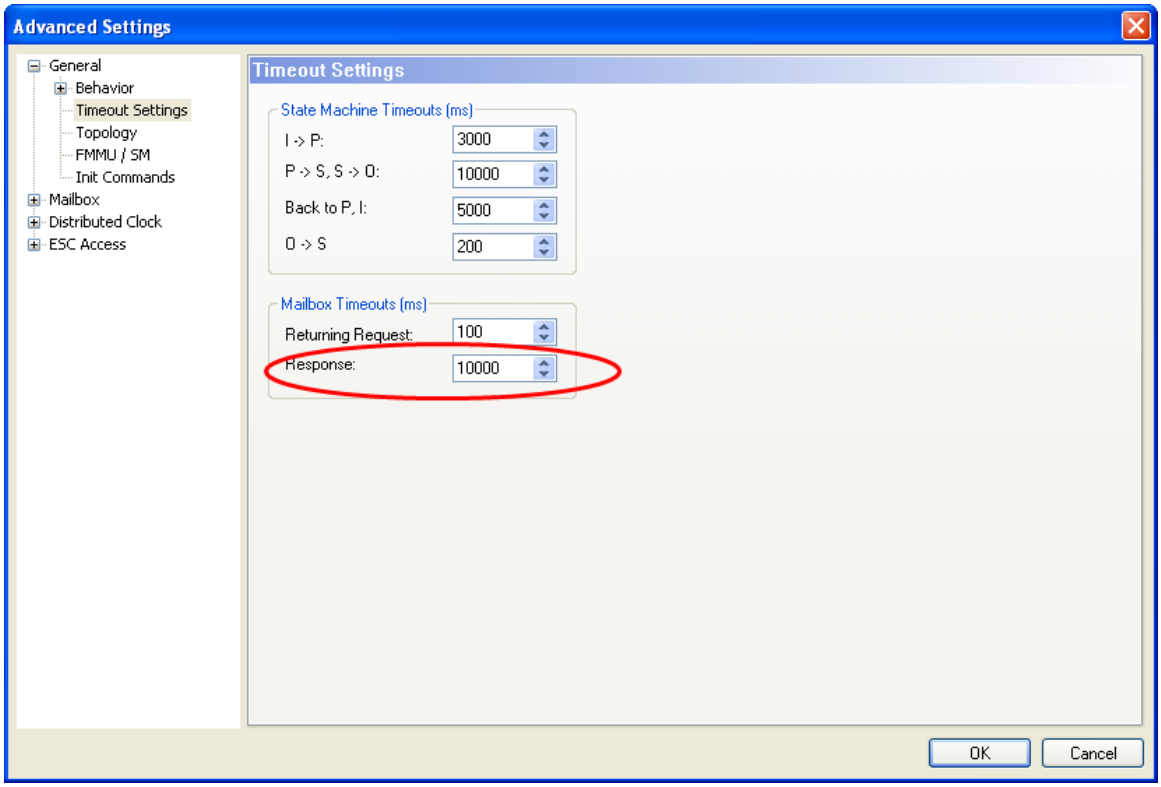
3. At the left hand explorer, click **Mailbox**.



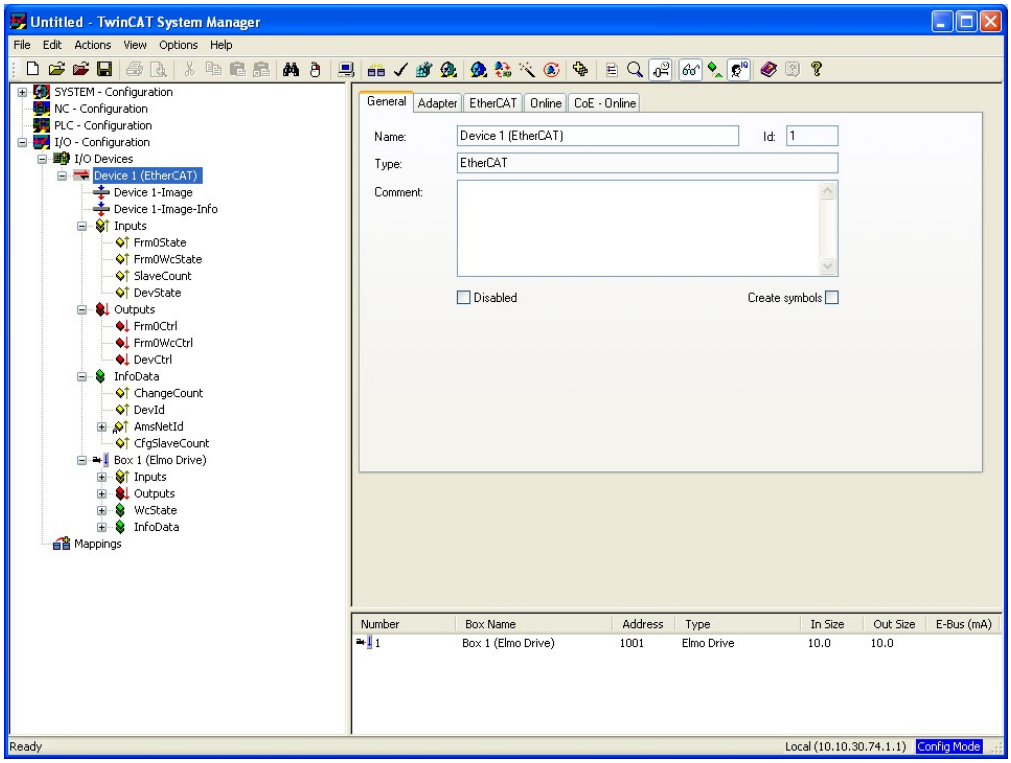
4. At the Mailbox Polling option, click the **Cyclic** checkbox and set the mailbox cycle time to **50 msec** by default.



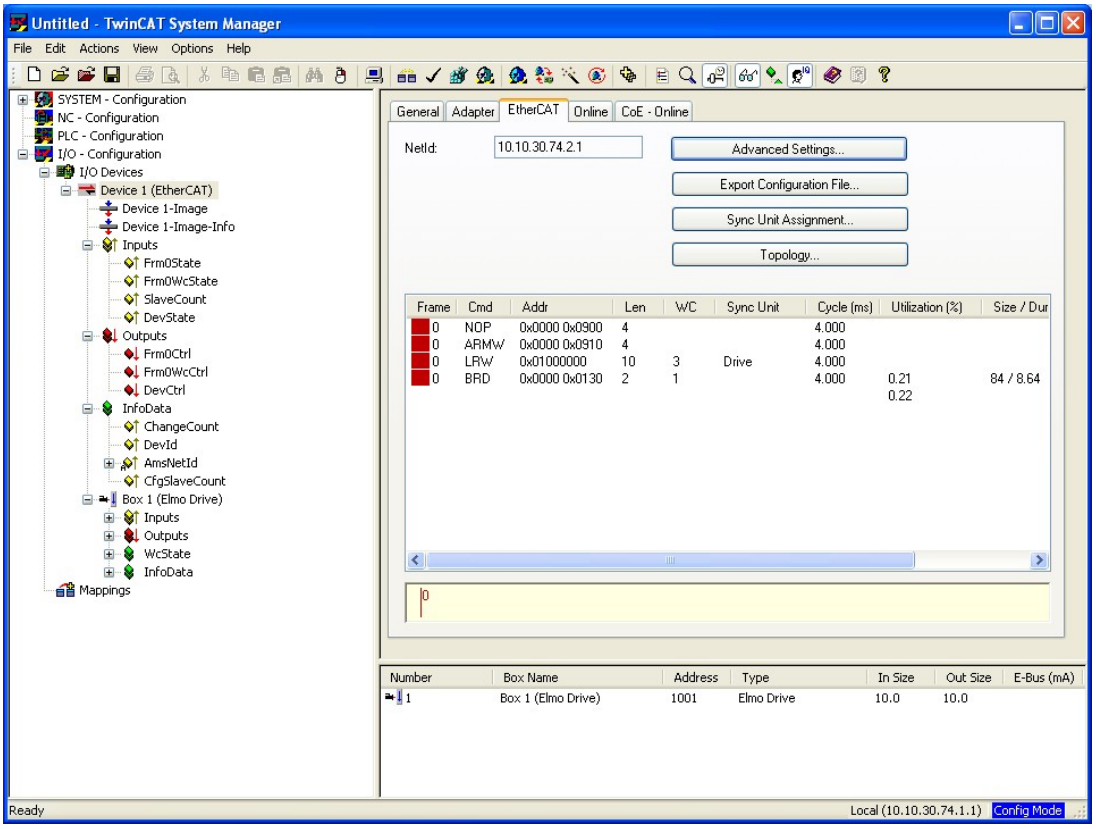
5. In the Timeout Settings of each mailbox, under General, set the Mailbox Timeout Response time to a minimum of **20,000 msec**.



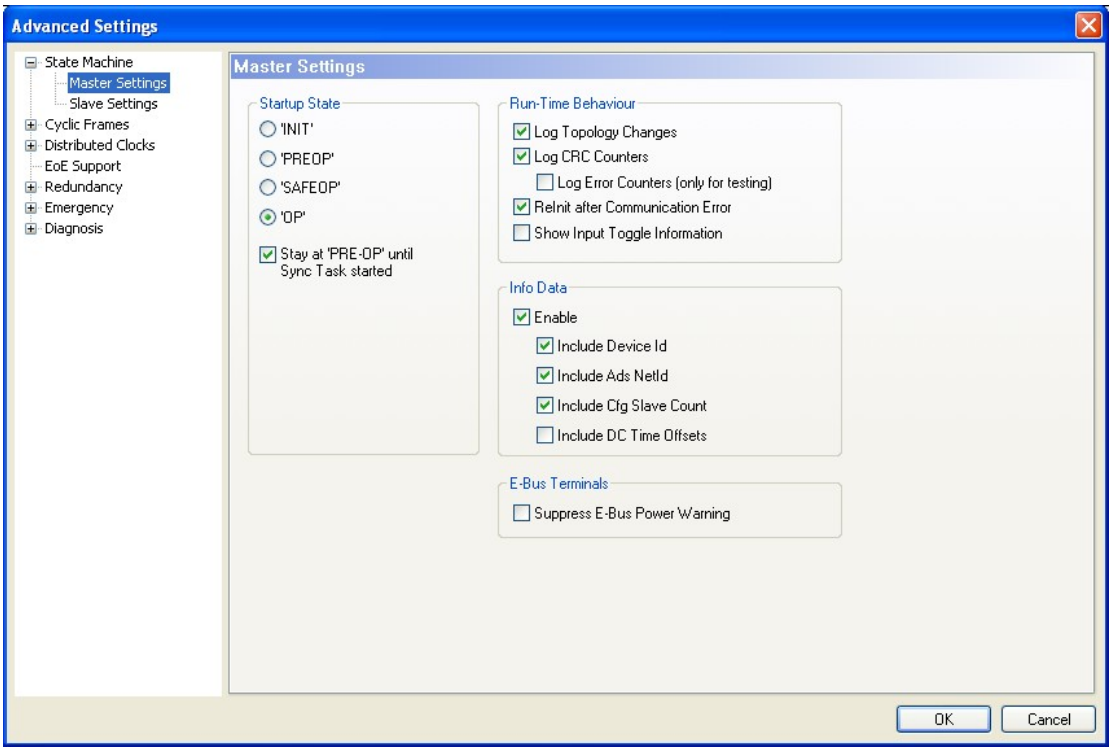
6. Click the Master **Device 1 (EtherCAT)** in the main explorer window at the left side.



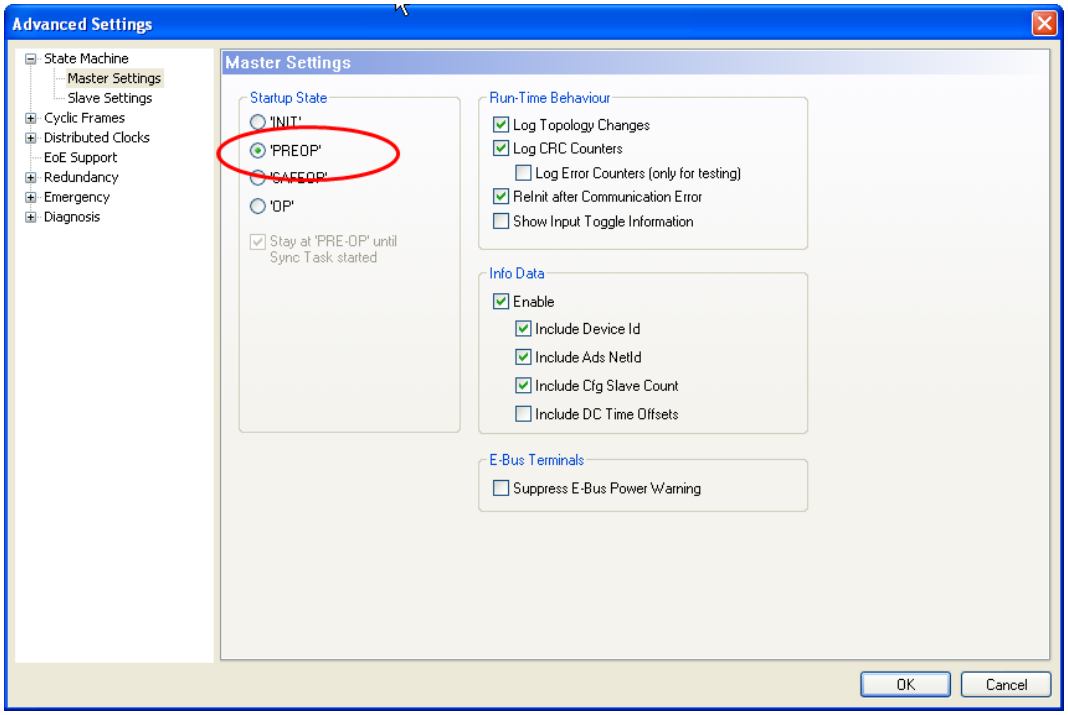
7. Click the EtherCAT tab, and select **Advanced Settings**.



8. From the explorer at the left side, select **Master Settings**.



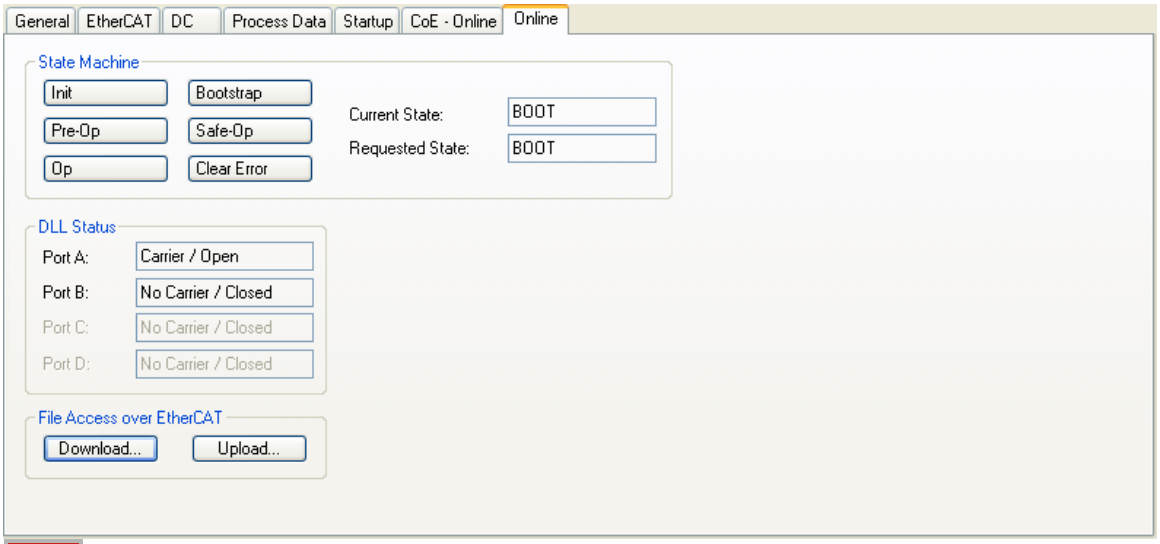
9. Set the EtherCAT Master to the **PreOP** startup state.



5.4.3. The Firmware Download Procedure

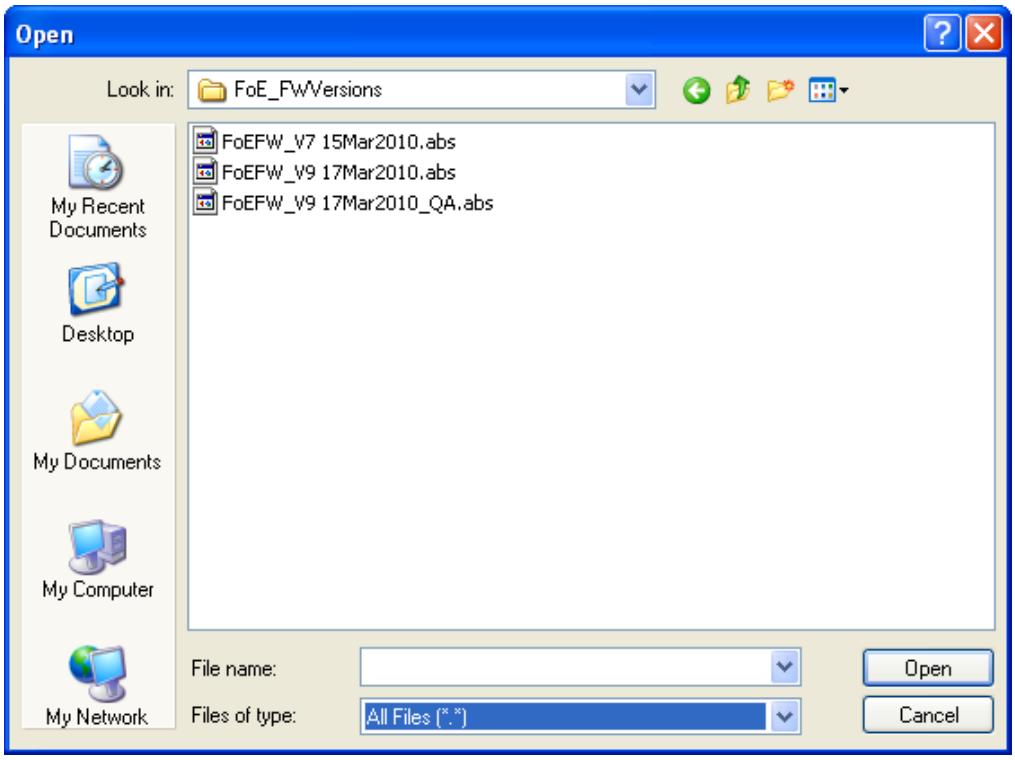
To perform firmware download

1. Restart all the drives in the network (including the target drive for download). Select the **Online** tab, set the target drive state to BOOT, and then click **Download**.



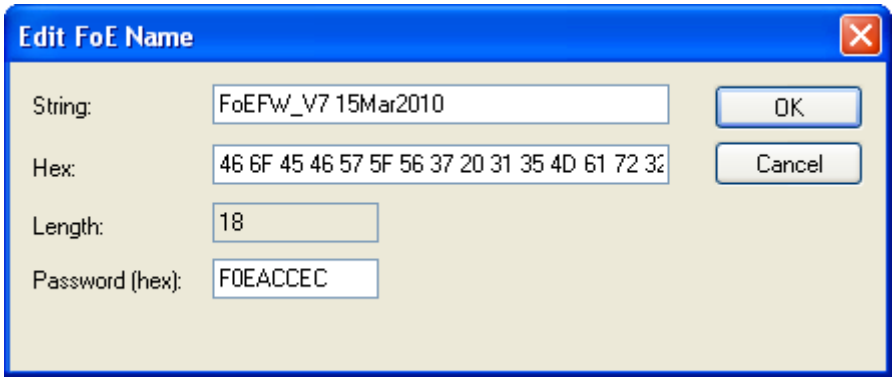
Note: If the state has not been changed to BOOT, wait 10 seconds and try again. If the state has still not changed, run: “set/reset TwinCAT to config mode”, wait 10 seconds and try again.

2. Click **Download** and an Open dialog box will appear. Change “Files of type” to “All Files”.



3. Select the file that will be downloaded (the filename prefix must start with ‘FoEFW_V’).

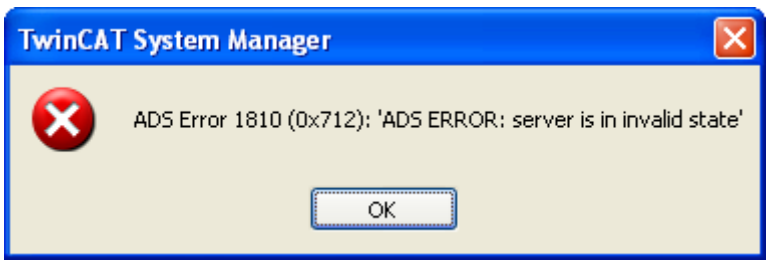
4. Click **Open**, and an Edit FoE Name dialog window appears.



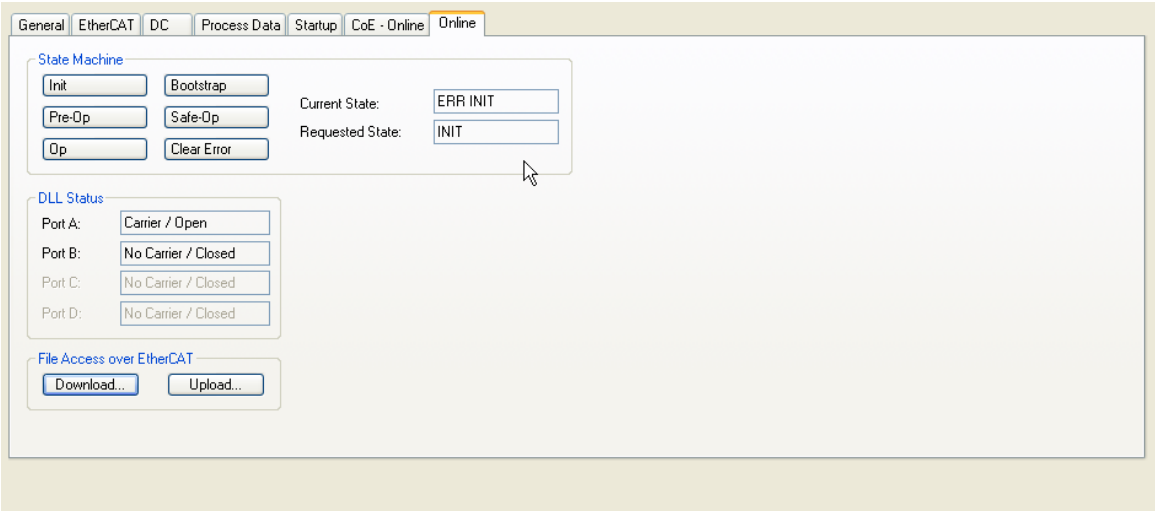
5. Enter the password: **F0EACCEC** and click **OK**. The download starts immediately.

6. After the download procedure is complete, do the following:

   a. When the TwinCAT progress bar finishes, an error “init state” message appears. Click **OK**.



b. Verify that the EtherCAT state has changed to **Err INIT**.



c. Reset the drive's hardware. When a PLC program is running, the whole network of drives require a hardware reset.

d. Reload the TwinCAT I/O devices (press F4).

Chapter 6: Gold Drive Object list

This chapter describes the Gold drive supported objects in a concise table. For detailed information on the objects, refer to the relevant Elmo standard document. Both EtherCAT CoE protocol and CANopen use a similar object list, but some objects used in one, are not in the other. The EtherCAT CoE specific object list is listed in the section.

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 sub-section details the Complete Object List (Section 5.1), which includes CAN, EtherCAT, and the Common Objects.

6.1. Complete Object Dictionary

The following table (Table 11) 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 in the MAN-G-DS301 manual, and objects 0x6000 – 0x6FFF are described in the MAN-G-DS402 manual.

Refer to the Table 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.
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.
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;
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)
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]**
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	Sub-index 1: TxMap	CAN, ECAT. In mV

2206/0	5V DC supply	UINT16	RO	TxMap	CAN, ECAT. In mV
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]**
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
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
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 1,2: RW	No	CAN, ECAT
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 Methods	UINT8	RO	No	CAN, ECAT
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 15: Object Dictionary

6.2. EtherCAT CoE - PDO Objects list

The CoE includes a predefined static list of PDO mappings. Table 6: RxPDO Default Values and Table 8: TxPDO Default Values describe the process data objects.

Notes: The total bytes which can be mapped by the Gold line EtherCAT is 32 bytes.
Objects between 0x1A00 (PDO1) to 0x1A09 are excluded from each other. Only one can be selected at a time.
The drive performs a single process in a cycle. In a situation where the same object is mapped twice, the last received object will be processed.

Padding in ESI (XML) file:
The padding object 0x0000 can be used to align address when required by the host.
When the host adds a padding object to a PDO, the drive ignores it.
Padding an object is described in ETG 1000-6.

Chapter 7: Elmo Emergency Error and Abort List

This chapter describes the Elmo Emergency Error and Abort codes. An Emergency Error at the drive sends a callback event, which can be interpreted and solved. These error will vary in their definition and description. In addition, further Errors may be added to the list.

An Abort Error is different. This error can occur due to a misreading of a command, causing the drive to abort and send an Abort Code via a handshake protocol. They are fixed by the CANopen standardization, and cannot be edited or added.

7.1. Emergency Error Description

The following list the Emergency Error codes and their description with the relevant Elmo code where applicable.

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
Reserved	**1000**	81	0	0
Over current	**2311**	03	0	0
Short circuit: motor or its wiring may be defective, or drive is faulty.	**2340**	03	0	0
Reserved	**3100**	05	0	0
Under-voltage: power supply is shut down or it has too high an output impedance.	**3120**	05	0	0
Loss of phase: Activated in specific HW version of the drive. Refer to the drive User manual.	**3130**	11	0	0
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.	**3310**	05	0	0
Temperature: 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.	**4310**	09	0	0
A difference is too large in ECAM table.	**5280**	81	0	0
Timing Error	**5281**	81	0	0
Reserved	**5282**	81	0	0
Reserved	**5400**	21	0	0
Motor disabled by switch FLS, RLS, STOP	**5441**	21	0	0

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
Motor disabled by additional abort motion. Function can be used to shut off the motion, produces a different fault code. (See **MF** command)	**5442**	81	0	0
Fatal CPU error: stack overflow	**6180**	81	0	0
CPU exception - fatal exception. Something such as an attempt to divide in zero or another fatal firmware error has occurred. Use the CD command to get the CPU dump and report to your service center.	**6181**	81	0	0
User program aborted by an error	**6200**	81	0	0
Object mapped to an RPDO returned an error during interpretation or a referenced motion failed.	**6300**	01	Elmo Error Code see section	0
Cannot start because of inconsistent database. The type of database inconsistency is reflected in the status SR report, and in the CD CPU dump report.	**6320**	21	0	0
Motor stuck - the motor is powered but is not moving according to the definition of **CL[2]** and **CL[3].**	**7121**	21	0	0
Feedback error: Resolver feedback is not ready – Resolver angle was not found yet. Analog encoder or Resolver feedback is either lost or with too low amplitude. Battery Alarm: Absolute Position may be incorrect due battery power loss.	**7300**	81	0	0
Commutation failed	**7306**	21	0	0
Feedback loss: no match between encoder and Hall location. Available in encoder + Hall feedback systems.	**7380**	81	0	0
Two digital Hall sensors were changed at the same time. Error occurs because digital Hall sensors must be changed one at a time.	**7381**	81	0	0
Commutation process fail during motor on for the reasons: For locking the phase	**7382**	81	0	0

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
Planar motor when on alignment process Fault reaction can be selected via 0x605E (**OF[6]**)				
CAN message lost (corrupted or overrun)	**8110**	11	Elmo Error Code see section 6.2	0x4000 (Sync lost) 0x2000 (RPDO lost) 0x200 (NMT lost) 0x100 (SDO lost)
Heartbeat event	**8130**	11	Elmo Error Code see section 6.2	Node ID
Recovered from bus off	**8140**	11	Elmo Error Code see section 6.2	0
Protocol error (unrecognized NMT request)	**8200**	11	0	NMT command received
Attempt to access a non-configured RPDO	**8210**	21	0	0
The peak current has been exceeded. Possible reasons are drive malfunction or bad tuning of the current controller.	**8311**	21	0	0
Failed to find the electrical zero of the motor in an attempt to start it with an incremental encoder and no digital Hall sensors. The reason may be that the applied motor current did not suffice for moving the motor from its position.	**8380**	81	0	0
Cant tune current offsets	**8381**	81	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 not a powerful enough motor	**8480**	81	0	0
Speed limit exceeded: **VX<LL[2]** or **VX>HL[2]**. (Compatibility only)	**8481**	81	0	0

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
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
Queue is low. Number of yet unexecuted PVT table rows has dropped below the value stated in **MP[4]**. (reserved)	**FF00**	81	0	Field 1: Write pointer, Field 2: Read pointer
Write pointer is out of physical range ([1…64]) of PVT table. Reason may be an improper setting of **MP[6].** (reserved)	**FF00**	81	0	Value of MP[6]
(Reserved for Compatibility reason)	**FF00**	81	0	0
An attempt has been made to program more PVT points than are available in queue. (reserved)	**FF00**	81	0	Field 1: Index of PVT table entry that could not be programmed
Cannot initialize motion due to bad setup data. The write pointer is outside the range specified by the start and end pointers. (reserved)	**FF00**	81	0	0
Mode terminated and motor has been automatically stopped (in **MO=1**). (reserved)	**FF00**	81	0	Data field 1: Write pointer; Data field 2: End of trajectory in non-cyclic mode A zero or negative time specified for a motion interval Read pointer reached write pointer
A CAN message has been lost. (reserved)	**FF00**	81	0	0

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
Request by user program “emit” function	**FF01**	81	0	0
IP mode underflow	**FF02**	81	Elmo Error Code see section 6.2	0
INTERPOLATION QUEUE FULL In Interpolated position motion mode the set-point buffer is full. The last received set-point is not interpolated.	**FF02**	81	Elmo Error Code see section 6.2	0
Index is not supported – Bad Index	**FF02**	81	Elmo Error Code see section 6.2	0
One or more PVT parameter is not set correctly - BAD PVT SEND ORDER	**FF02**	81	Elmo Error Code see section 6.2	0
Cannot start motor.	**FF10**	81	Elmo Error Code see section 6.2	0
Safety switch is sensed – drive in safety state	**FF20**	05	0	0
Modulo overflow limit	**FF30**	81	0	0
Numeric overflow limit	**FF34**	81	0	0
Gantry Slave Disabled	**FF40**	81	0	0

7.2. ELMO Error Codes

The following list the ELMO Error codes and their description with the relevant Elmo code where applicable.

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
Reserved	**1000**	81	0	0
Over current	**2311**	03	0	0
Short circuit: motor or its wiring may be defective, or drive is faulty.	**2340**	03	0	0
Reserved	**3100**	05	0	0
Under-voltage: power supply is shut down or it has too high an output impedance.	**3120**	05	0	0
Loss of phase: Activated in specific HW version of the drive. Refer to the drive User manual.	**3130**	11	0	0
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.	**3310**	05	0	0
Temperature: 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.	**4310**	09	0	0
A difference is too large in ECAM table	**5280**	81	0	0
Timing Error	**5281**	81	0	0
Reserved	**5282**	81	0	0
Reserved	**5400**	21	0	0
Motor disabled by switch FLS, RLS, STOP	**5441**	21	0	0
Motor disabled by additional abort motion. Function can be used to shut	**5442**	81	0	0

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
off the motion, produces a different fault code. (See **MF** command)				
Fatal CPU error: stack overflow	**6180**	81	0	0
CPU exception - fatal exception. Either an attempt to divide in zero or another fatal firmware error has occurred. Use the **CD** command to get the CPU dump and report to your service center.	**6181**	81	0	0
User program aborted by an error	**6200**	81	0	0
Object mapped to an RPDO returned an error during interpretation or a referenced motion failed.	**6300**	01	Elmo Error Code see section Chapter 1:	0
Cannot start because of inconsistent database. The type of database inconsistency is reflected in the status SR report, and in the CD CPU dump report.	**6320**	21	0	0
Motor stuck - the motor is powered but is not moving according to the definition of **CL[2]** and **CL[3].**	**7121**	21	0	0
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.	**7300**	81	0	0
Commutation failed	**7306**	21	0	0

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
Feedback loss: no match between encoder and Hall location. Available in encoder + Hall feedback systems.	**7380**	81	0	0
Two digital Hall sensors were changed at the same time. Error occurs because digital Hall sensors must be changed one at a time.	**7381**	81	0	0
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])	**7382**	81	0	0
CAN message lost (corrupted or overrun)	**8110**	11	Elmo Error Code see sec. 6.2	0x4000 (Sync lost) 0x2000 (rPDO lost) 0x200 (NMT lost) 0x100 (SDO lost)
Heartbeat event	**8130**	11	Elmo Error Code see sec. 6.2	Node ID
Recovered from bus off	**8140**	11	Elmo Error Code see sec. 6.2	0
Protocol error (unrecognized NMT request) (Reserved)	**8200**	11	0	**NMT** command received
Attempt to access a non-configured RPDO	**8210**	21	0	0
The peak current has been exceeded. Possible reasons are drive malfunction or bad tuning of the current controller.	**8311**	21	0	0
Failed to find the electrical zero of the motor in an attempt to start it with an incremental encoder and no digital	**8380**	81	0	0

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
Hall sensors. The reason may be that the applied motor current was insufficient to move the motor from its position.				
Cannot tune current offsets	**8381**	81	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 not a powerful enough motor	**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

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
Queue is low. Number of yet unexecuted PVT table rows has dropped below the value stated in **MP[4]**. (reserved)	**FF00**	81	0	Field 1: Write pointer, Field 2: Read pointer
Write pointer is out of physical range ([1…64]) of PVT table. Reason may be an improper setting of **MP[6].** (reserved)	**FF00**	81	0	Value of **MP[6]**
(Reserved for Compatibility reason)	**FF00**	81	0	0
An attempt has been made to program more PVT points than are available in queue. (reserved)	**FF00**	81	0	Field 1: Index of PVT table entry that could not be programmed
Cannot initialize motion due to bad setup data. The write pointer is outside the range specified by the start and end pointers. (reserved)	**FF00**	81	0	0
Mode terminated and motor has been automatically stopped (in **MO=1**). (reserved)	**FF00**	81	0	Data field 1: Write pointer; Data field 2: End of trajectory in non-cyclic mode A zero or negative time specified for a motion interval Read pointer reached write pointer
A CAN message has been lost. (reserved)	**FF00**	81	0	0
Request by user program *emit* function	**FF01**	81	0	0
IP mode underflow	**FF02**	81	Elmo Error Code see sec.	0

Description	Error Code Hex	Error Register Hex	Elmo Error Code Dec	Data Field
			6.2	
Interpolation Queue Full In Interpolated Position motion mode the set-point buffer is full. The last received set-point will not interpolate.	**FF02**	81	Elmo Error Code see sec. 6.2	0
Index is not supported – Bad Index	**FF02**	81	Elmo Error Code see sec. 6.2	0
One or more PVT parameter is not set correctly.	**FF02**	81	Elmo Error Code see sec. 6.2	0
Cannot start motor.	**FF10**	81	Elmo Error Code see sec. 6.2	0
Safety switch is sensed – drive in safety state	**FF20**	05	0	0
Modulo overflow limit	**FF30**	81	0	0
Numeric overflow limit	**FF34**	81	0	0
Gantry Slave Disabled	**FF40**	81	0	0

Table 16 Emergency error codes

7.3. Abort SDO Transfer Protocol

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

Client to server *or* server to client

0	1	4	8
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/sub-index of SDO.

d Four-byte abort code giving reason for abort, encoded as Unsigned32 value.

Abort Code	Description
0503 0000h	Toggle bit not alternated.
0504 0001h	Invalid or unknown client/server command specifier.
0504 0002h	Invalid block size
0504 0003h	Invalid sequence number in SDO block upload
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read a write-only object.
0601 0002h	Attempt to write a read-only object.
0602 0000h	Object does not exist in object dictionary.
0604 0041h	Object cannot be mapped to PDO.
0604 0042h	Number and length of objects to be mapped exceeds PDO length.
0604 0043h	General parameter incompatibility.
0606 0000h	Access failed due to hardware error.
0607 0012h	Data type does not match, service parameter too long.
0609 0011h	Sub-index does not exist.
0609 0030h	Value range of parameter exceeded (only for write access).
0609 0031h	Value of parameter written too high.
0609 0032h	Value of parameter written too low.

Abort Code	Description
0609 0036h	Maximum value is less than minimum value.
0800 0000h	General error. When the abort code is 0800 0000h, the actual error can be retrieved using the **EC** command.
0800 0020h	Data cannot be transferred to or stored in application.
0800 0022h	Data cannot be transferred to or stored in application due to present device state.


Inspiring Motion
Since 1988
For a list of Elmo's branches, and your local area office, refer to the Elmo site www.elmomc.com
Elmo
Motion Control