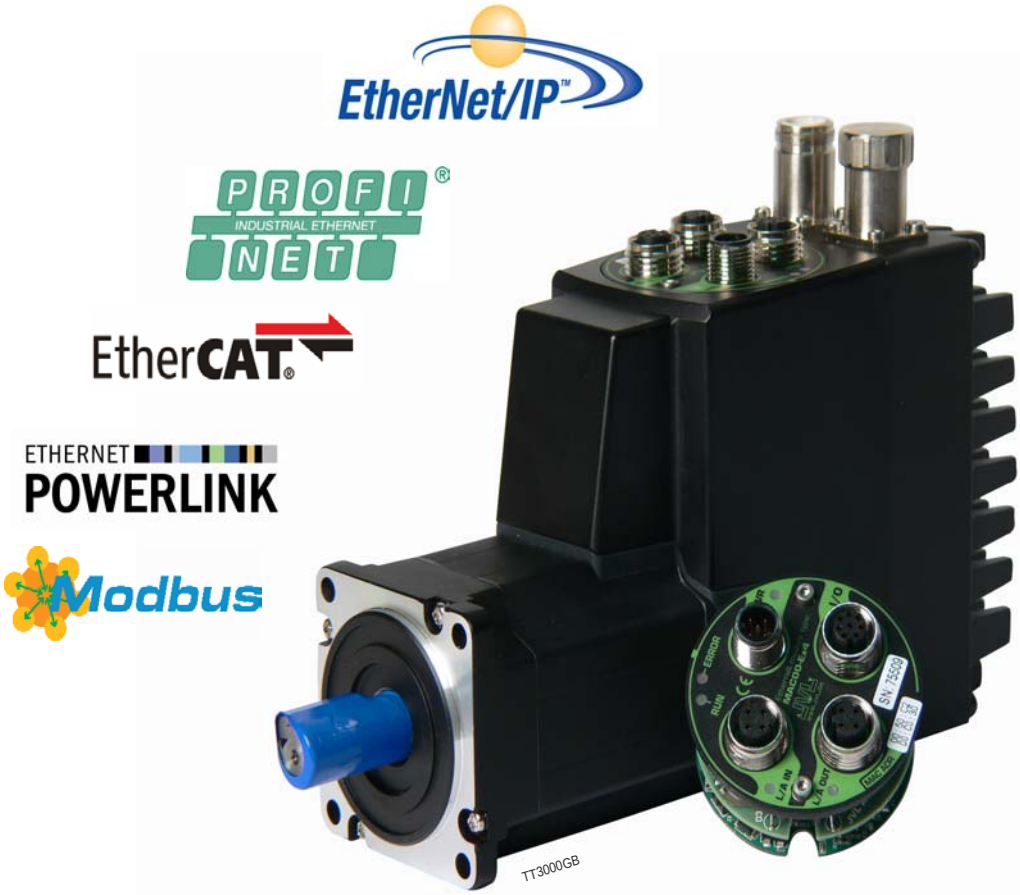
MAC00-EC4, MAC00-EI4
MAC00-EL4, MAC00-EP4
&
MAC00-EM4

Industrial Ethernet expansion modules for MAC Servo Motors

User Manual



JVL Industri Elektronik A/S

LB0056-17GB

Revised 7.5.2013

Important
User Information

Warning

The MAC series of products are used to control electrical and mechanical components of motion control systems.
You should test your motion system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

Please contact your nearest JVL representative in case of technical assistance. Your nearest contact can be found on our web site www.jvl.dk

JVL Industri Elektronik A/S
Blokken 42
DK-3460 Birkerød
Denmark
Tlf. +45 45 82 44 40
Fax. +45 45 82 55 50
e-mail: jvl@jvl.dk
Internet: http://www.jvl.dk

CANopen®	Is a registered trademark of CAN in AUTOMATION - International Users and Manufacturers Group e. V. (CiA), Nürnberg.
DeviceNet®	Is a trademark of ODVA (Open DeviceNet Vendor Association, Inc).
EtherCAT®	Is a registered trademark and a patented technology of Beckhoff Automation GmbH, Verl, Bundesrepublik Deutschland, formerly Elektro Beckhoff GmbH.
EtherNet/IP®	Is a trademark of ODVA (Open DeviceNet Vendor Association, Inc).
Modbus TCP/IP®	Is a registered trademark of Schneider Electric.
PROFINET IO®	Is a registered trademark of PROFIBUS International, Karlsruhe.
SERCOS interface®	Is a registered trademark of SERCOS International e.V., Suessen, Germany.

Contents

1 Introduction

1.1 Introduction

Industrial Ethernet is becoming more and more popular as it offers

- Very fast response time
- Predictable delay times (deterministic protocol)
- Safe transmission of data

Compared with most of the "classic" non Ethernet based protocols the indstrial Ethernet offers state of the art performance.

The MAC00-Ex4 Industrial Ethernet module can be configured by the end user to a number of different Ethernet protocols, for instance

- EtherCAT®
- EtherNetIP®
- Ethernet POWERLINK®
- PROFINET IO®
- Modbus TCP/IP®
- And more to come

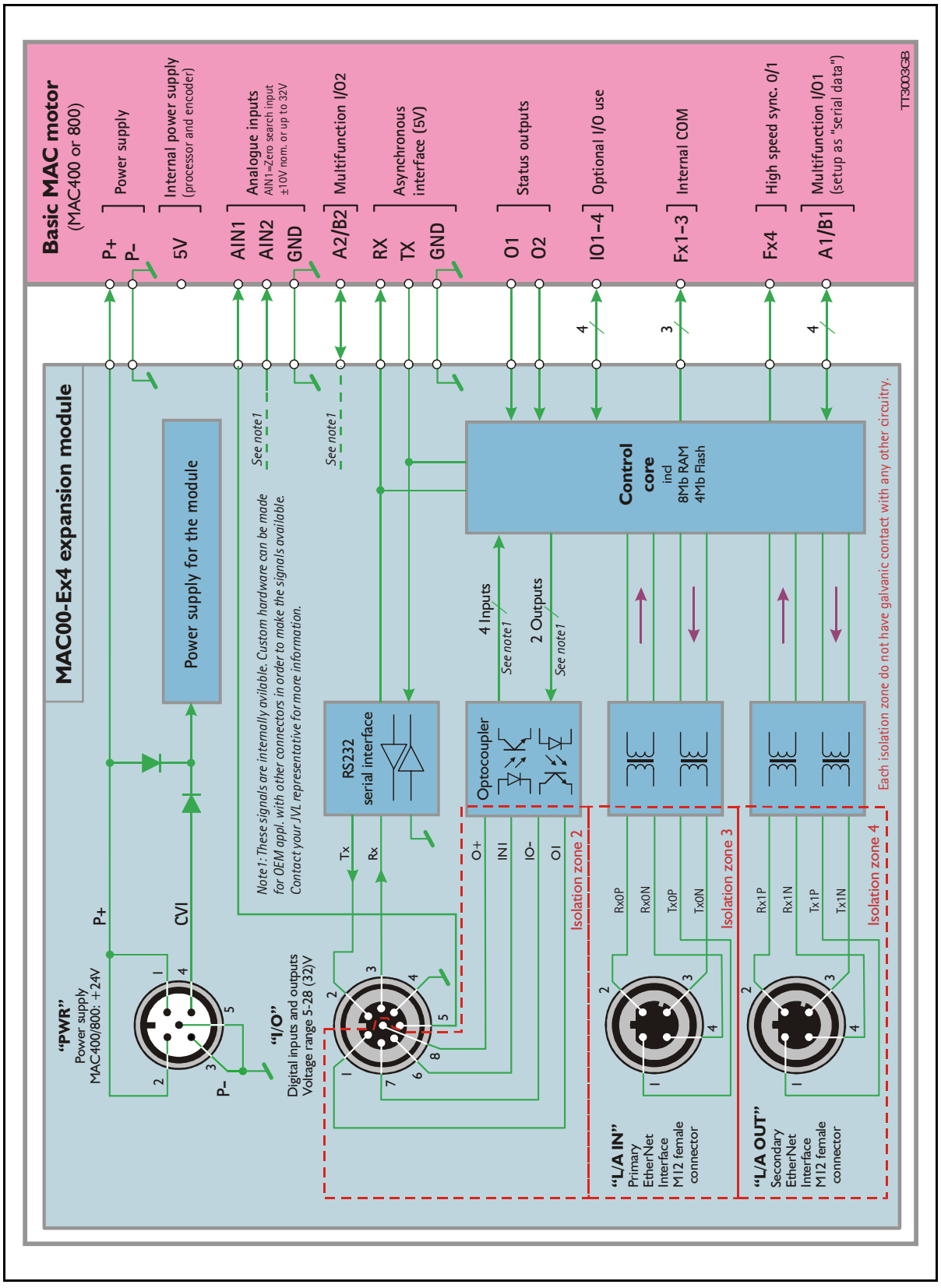
Main Features:

- High speed communication - 100Mbits/sec.
- 2 individual ports on the module offers Daisy chaining possibility.
- Standard M12 circular industrial connectors
- 1 Digital input (24V) and 1 digital output (24V) for local use around the motor
- Multiple alternative I/O possibilities available on request (OEM applications)
- LED's for easy monitoring of operation status
- Optional encoder I/O
- Rough design
- Access to all internal motor parameters and registers possible. No need of pre-setup of the motor.
- RS232 connection available for monitoring and setup use if desired.

1.2 Hardware introduction

1.2.1 Overall hardware description

All internal and external main connections can be seen in the illustration below.



2 General Hardware description

2.1 Module types

2.1.1 Module types

The MacMotor Ethernet modules are available for several Ethernet protocols.
The module used for each protocol has its own unique type number, but is based on the exactly same hardware.
A neutral module where no protocol is installed however also exist.

- **Neutral module - no protocol installed.**
  MAC00-Ex4 is a neutral module not setup-up for any particular protocol. The final user can setup it up for any of the available protocols just by using the general MacTalk windows software.
  The visible LED marking, lables etc. only states that its a neutral MAC00-Ex4 module.

- **Pre-loaded module - a specific protocol has been installed.**
  The modules MAC00-EC4 (EtherCAT), MAC00-EI4 (EtherNetIP), and MAC00-EL4 (POWERLINK), MAC00-EP (Profinet), MAC00-EM (Modbus TCP) are setup at delivery with the relevant protocol and also the right LED marking.
  The final user can setup it up for any of the available protocols just by using the general MacTalk windows software.
  The visible LED marking, and type number is unique for each module type.

All modules (when not delivered mounted in a MacMotor) is followed by a little label sheet containing labels for all the available standards and standards to come.

The overall idea is that any module can be changed to another protocol if desired, the modules can stay neutral when it passes the distribution channel and be setup by the end-user simplifying the logistics.

2.1.2 How to setup a module for a protocol.

Only 2 steps are needed in this process.

1. Install the intended protocol firmware in the module.
2. Apply or changing the label with LED marking and typenumber of the module.

The firmware can be setup as follows

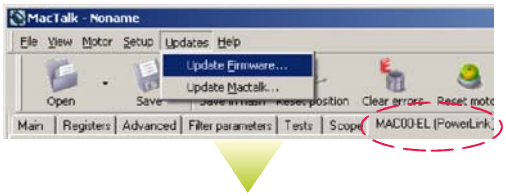
(see next page)

2.1 Module types

How to setup the module for a different/new protocol

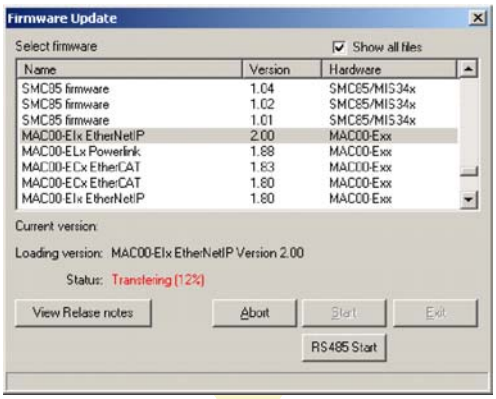
Step 1
Determine which Ethernet protocol you want to use.
Have in mind that your Ethernet module MAC00-Ex4 may already be setup for a protocol.

Step 2
As shown the module is setup as a MAC00-EL module with the Ethernet Powerlink protocol.
Choose the *Update Firmware* in the *Updates* menu to setup the module with another protocol.

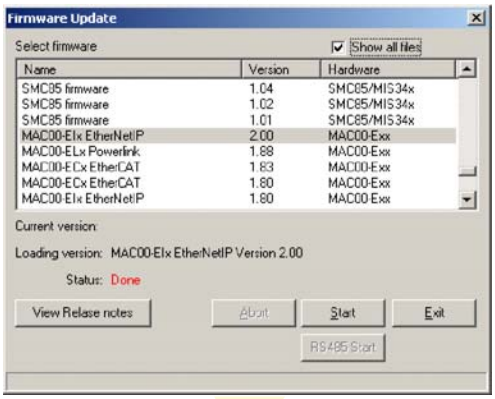


Step 3
Make sure that the checkbox *"Show all files"* is checked.
Select the desired firmware such as EtherNet-IP. Note that there may exist more than one version. Choose the newest version.

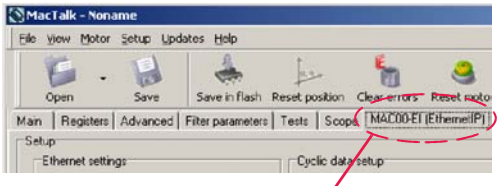
Press *Start* to download the selected firmware.
The status counter will now rise from 0 to 100%.



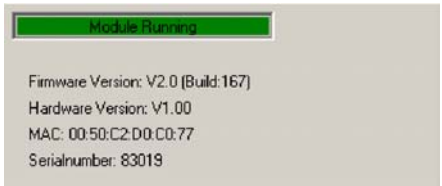
Step 4
When the download process is finished, the status shows "*Done*".
Also "*Current version*" has changed to the actual downloaded version meaning that the firmware in the module is now changed permanently.



Step 5
The module tab has now changed from MAC00-EL to MAC00-EI (EthernetIP).



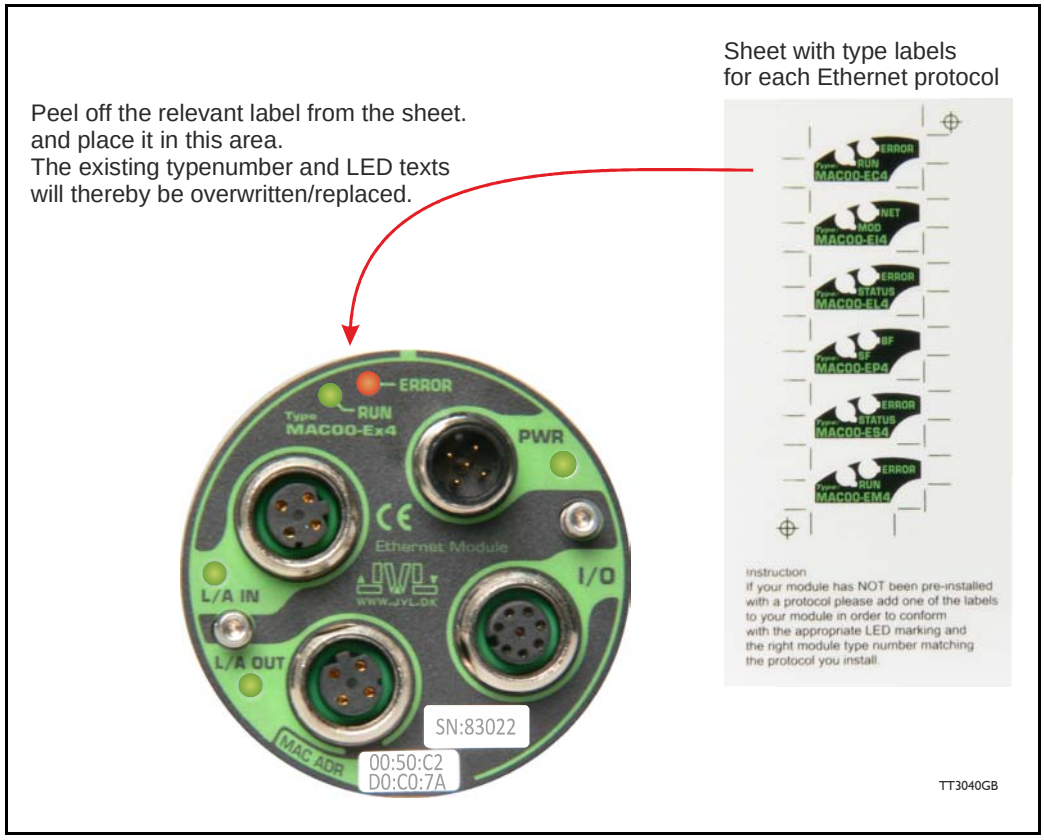
Step 6
The firmware version, MAC address etc. can be monitored on the module tab.



TT3039GB

2.1 Module types

Changing the label and typenumber
This illustration show how to apply the appropriate label in order to change the LED texts and also give the module its unique typenumber after the protocol firmware is loaded.



Typenumber overview:

Type	Ethernet Protocol
MAC00-EC4	EtherCAT
MAC00-EI4	EtherNET / IP
MAC00-EL4	EtherNet POWERLINK
MAC00-EM4	Modbus TCP
MAC00-EP4	Profinet IO
MAC00-ES4	Sercos III

2.2 I/O descriptions

2.2.1 Hardware overview

Expansion module MAC00-Ex4 front plate
(Neutral module for all the Ethernet protocols)

Module status indicators.

L/A IN
Primary Ethernet channel M12 - 4pin female (D-coded) and LED for showing activity.

L/A OUT
Secondary Ethernet channel M12 - 4pin female Ethernet coded (D coded) Used when module is daisy chained

MAC Address
Each module is having its own unique MAC address used to identify it on the Ethernet network. The MAC address can also be read electronically

PWR
Power supply connector M12 - 5pin male and Green LED for indicating power applied

I/O
I/O's and RS232 interface M12 - 8pin female connector including:
1 digital input and 1 digital output, 1 analogue input

Serial number
Eeach module have its own unique serial number which can be used for determine hardware version etc.

2.2.2 External signals available at the MAC00-Ex4

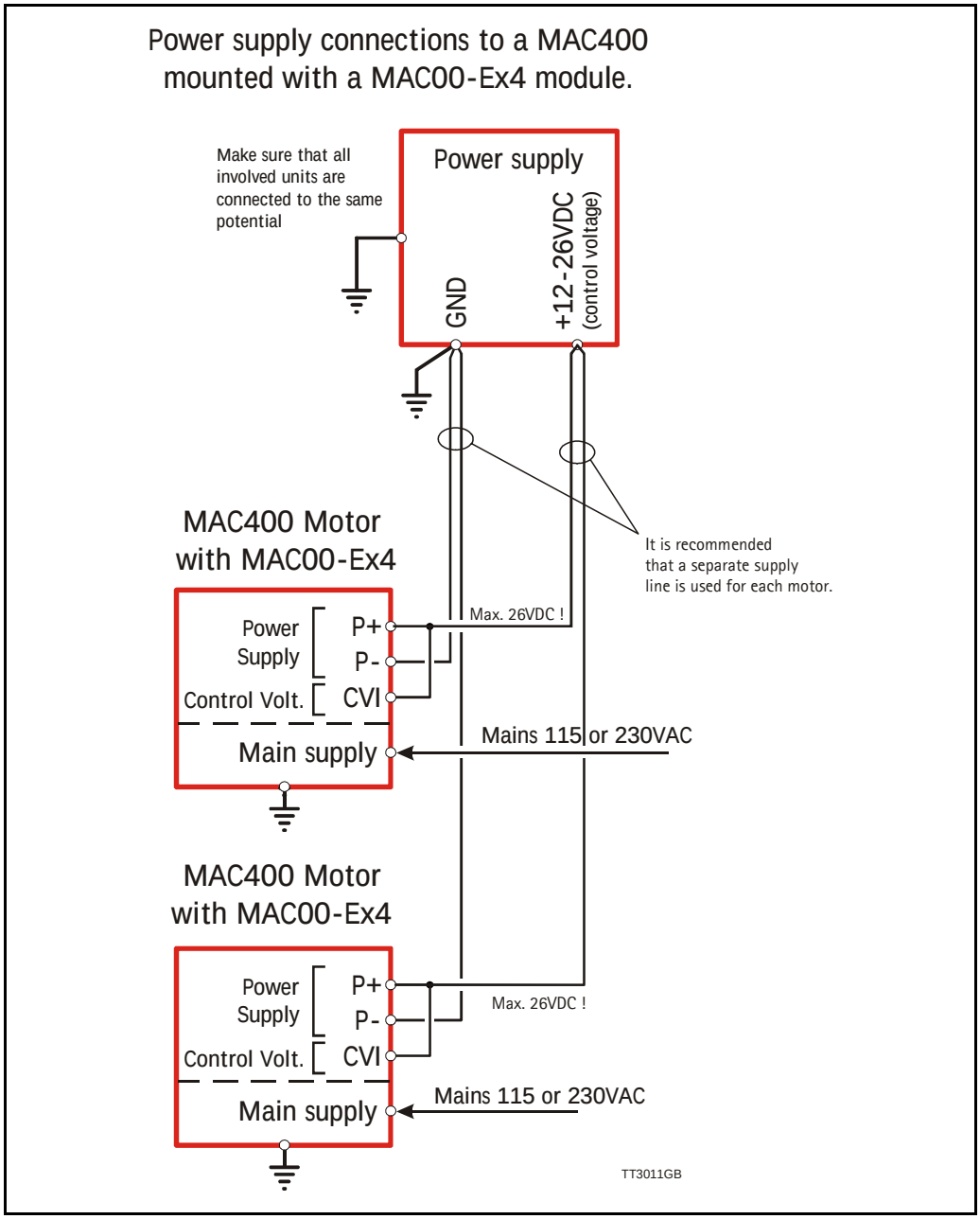
Following signals are available at the MAC00-Ex4 module.

- **"L/A IN" and L/A OUT" connector.**
  - The Ethernet connection. L/A IN is connected to the upstream master and L/A OUT can be used downstream for the next motors/units in the chain.
- **"I/O" connector.**
  - AIN - analogue input +/-10V.
    Can be used as input for the zero search sensor or as general analog input for speed or torque control depending on the what the actual operation mode in the motor has been setup for.
  - O1 - user output 1
    Can be used as dedicated "in position" output (default) or as general output controlable over the Ethernet interface.
  - RS232 Interface.
    Serial unbalanced interface for connection to a PC or a controller. The protocol is similar to the USB or RS485 interface, which means that all registers/parameters in the motor can be monitored or changed. RS232 is not recommended for long distances (>10m).
  - IN1 - User input 1.
    Can be used as general input which can be read over the Ethernet interface.
  - I/O supply and gnd (IO- and O+).
    Used as ground and supply for the user in/output (O1 and IN1).
- **"PWR" connector.**
  - 24V supply for the internal control circuitry in the motor.

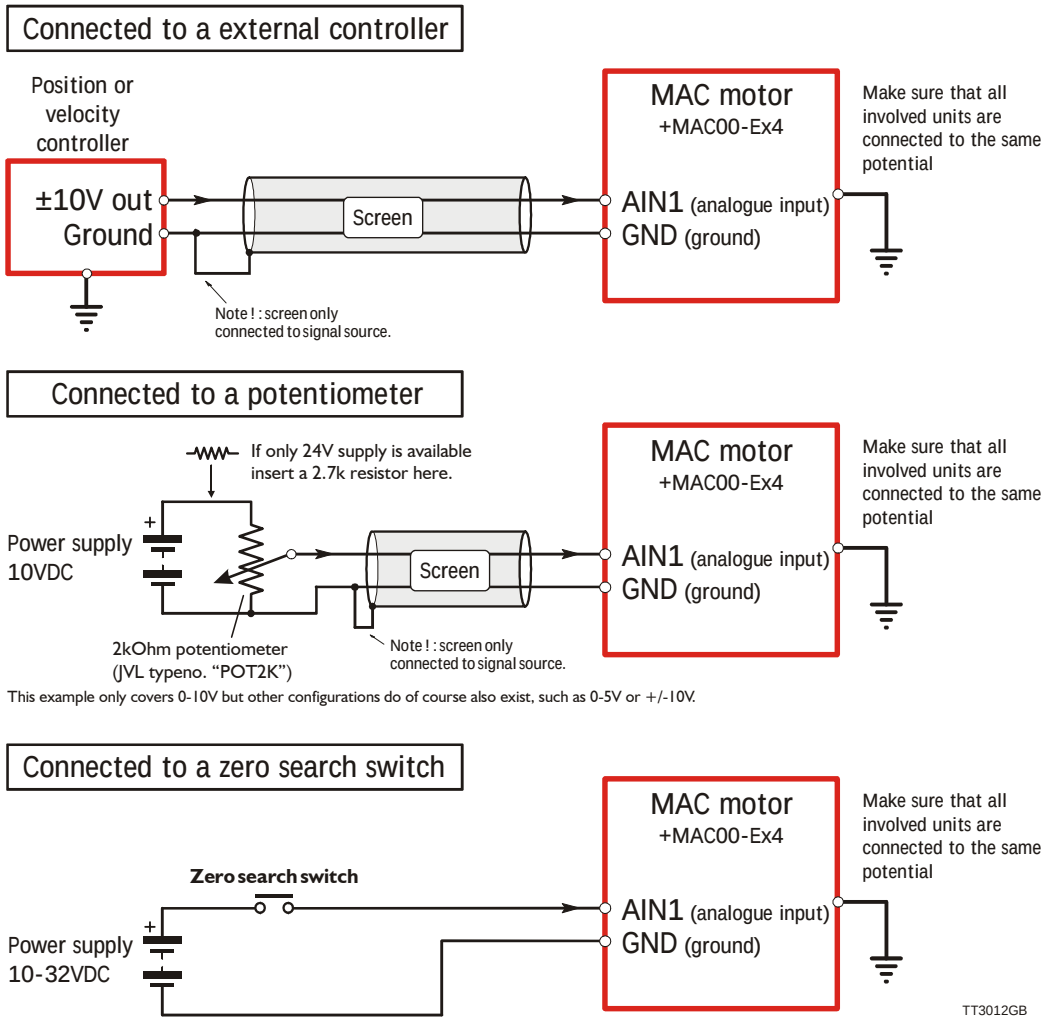
2.2.3 General power supply description

The Ethernet modules can be used in the allmost all the MAC motors but please be aware that to use the MAC50 to 141 they will need the special option : *"A009"* for example *"MAC140-A1-AAAA-A009"*
. The diagram below shows how to connect power to a MAC400 motor mounted with a MAC00-Ex4 module. Please notice that the voltage connected to P+ and/or CVI must stay in the range +12-26VDC. When using a MAC50 to 141 up to 48VDC is allowed.

See also the general power supply description in the MAC motor main manual LB0047. For further information concerning physical connections, see the *Expansion module MAC00-Ex4 connector description, page 19*.



Analogue input connection at the MAC motor mounted with a MAC00-Ex4 module.



Note: Do not apply voltages higher than 32V to the analogue input (AIN)

2.2.4 Using the analogue input (AIN1).

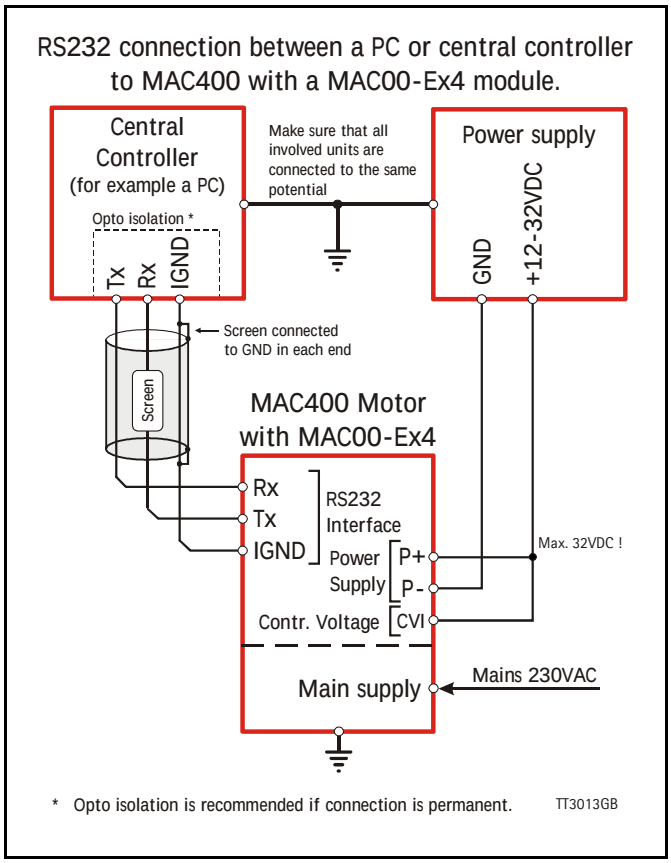
When a MAC00-Ex4 module is mounted in the MAC motor, the analogue inputs is available in the same manner as in the basic motor itself.
The analogue inputs can be used for several applications and the function of the analogue input is determined by the mode in which the motor is set to operate.
Typically the inputs is used for controlling the velocity, torque or position of the motor but the input is also used as digital input for zero search or in "Air Cylinder Mode" where it is used as trigger input for the movement done by the motor.
For further information concerning physical connections, see the *Expansion module MAC00-Ex4 connector description, page 19*.

2.2.5 RS232 - General description when using the MAC00-Ex4 module

The RS232 interface is considered the main interface to the motor when the motor is set up using the MacTalk windows software from a PC or from any kind of controller using a RS232 interface.

When connecting the RS232 interface to a PC or controller, the following rules must be followed:

1 Only one motor can be connected at the interface line.
2 Use screened cable.
3 Ensure that GND (interface ground) is also connected.
4 Ensure that all units have a proper connection to safety ground (earth) in order to refer to the same potential.
5 The RS232 interface cable length should not exceed 10 metres.



Connectors:
To see the specific connector pin-out please see the chapter *Expansion module MAC00-Ex4 connector description, page 19*.

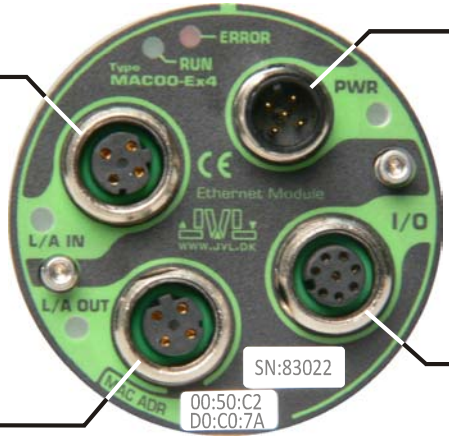
A finished RS232 cable also exist. Please see *Cables for the MAC00-Ex4, page 21*

2.3 Connector description

Expansion module MAC00-Ex4 front plate

L/A IN
Primary Ethernet channel
M12 - 4pin female
Ethernet coded
(D coded)

L/A OUT
Secondary Ethernet channel
M12 - 4pin female
Ethernet coded
(D coded)
Used when module is daisy chained



PWR
Power supply
M12 - 5pin male connector including:
P+ (primary supply), and CVI (secondary supply) and *P-*

I/O
I/O's and RS232 interface
M12 - 8pin female connector including:
1 digital input and 1 digital output, 1 analogue input

2.3.1 Expansion module MAC00-Ex4 connector description

The MAC00-Ex4 offers IP65 protection and M12 connectors which makes it ideal for automation applications where no additional protection is desired. The M12 connectors offer solid mechanical protection and are easy to unplug.

The connector layout:

"PWR" - Power input. M12 - 5pin male connector				
Signal name	**Description**	**Pin no.**	**JVL Cable WI1000-M12F5T05N**	**Isolation group**
P+	Main supply +12-24VDC. Connect with pin 2 *	1	Brown	1
P+	Main supply +12-24VDC. Connect with pin 1 *	2	White	1
P-	Main supply ground. Connect with pin 5 *	3	Blue	1
CVI	Control supply +12-24VDC. **DO NOT connect >25V to this terminal !**	4	Black	1
P-	Main supply ground. Connect with pin 3 *	5	Grey	1
* Note: P+ and P- are each available at 2 terminals. Make sure that both terminals are connected in order to split the supply current in 2 terminals and thereby avoid an overload of the connector.				

(Continued next page)

2.3 Connector description

"I/O" - I/O's and interface. M12 - 8pin female connector.				
Signal name	**Description**	**Pin no.**	**JVL Cable WI1000-M12 M8T05N**	**Isolation group (See note)**
O1	Output 1 - PNP/Sourcing output	1	White	2
RS232: TX	RS232 interface. Transmit terminal Leave open if unused.	2	Brown	1
RS232: RX	RS232 interface. Receive terminal Leave open if unused.	3	Green	1
GND	Interface ground to be used together with the other signals in this connector. Also ground for the analogue input (AIN1 - pin 5)	4	Yellow	1
AIN1	Analogue input1 ±10V or used for zero search	5	Grey	1
IN1	Digital input 1 - 12-32V tolerant.	6	Pink	2
IO-	I/O ground to be used with the I/O terminals O1 and IN1.	7	Blue	2
O+	Positive supply input to the output circuitry. Connect 5-32VDC to this terminal if using the O1 output.	8	Red	2
"L/A IN" - Ethernet port connector - M12 - 4pin female connector "D" coded				
Signal name	**Description**	**Pin no.**	**JVL Cable WI1046-M12M4S05R**	**Isolation group (See note)**
Tx0_P	Ethernet Transmit channel 0 - positive terminal	1	-	3
Rx0_P	Ethernet Receive channel 0 - positive terminal	2	-	3
Tx0_N	Ethernet Transmit channel 0 - negative terminal	3	-	3
Rx0_N	Ethernet Receive channel 0 - negative terminal	4	-	3
"L/A OUT" - Ethernet port connector. M12 - 4 pin female connector "D" coded				
Signal name	**Description**	**Pin no.**	**JVL Cable WI1046-M12M4S05R**	**Isolation group (see note)**
Tx1_P	Ethernet Transmit channel 1 - positive terminal	1	-	4
Rx1_P	Ethernet Receive channel 1 - positive terminal	2	-	4
Tx1_N	Ethernet Transmit channel 1 - negative terminal	3	-	4
Rx1_N	Ethernet Receive channel 1 - negative terminal	4	-	4

* Note: Isolation group indicate which terminals/circuits that a galvanic connected to each other. In other words group 1, 2, 3 and 4 are all fully independantly isolated from each other. Group 1 correspond to the housing of the motor which may also be connected to earth via the DC or AC input supply.

2.4 Cable accessories

2.4.1 Cables for the MAC00-Ex4

The following cables equipped with M12 connector can be supplied by JVL.

MAC00-Ex4 Connectors				Description	JVL Order no.	Picture
"L/A IN" 4pin male	**"L/A OUT"** 4pin Female	**"I/O"** 8pin Female	**"PWR"** 5pin Male			
		X		RS232 Interface cable. Connects directly from MAC00-Ex4 to a PC Length: 5m (197 inch)	**RS232-M12-1-5-8**	
		X		Cable with M12 **male** 8-pin connector loose wire ends 0.22mm² (24AWG) and screen. Length: 5m (197 inch)	**WI1000-M12M8T05N**	
		X		Same as above but 20m (787 inch)	**WI1000-M12M8T20N**	
			X	Cable (Ø5.5mm) with M12 **female** 5-pin connector loose wire ends 0.35mm² (22AWG) and foil screen. Length: 5m (197 inch)	**WI1000-M12F5T05N**	
			X	Same as above but 20m (787 inch)	**WI1000-M12F5T20N**	
X	X			Ethernet cable with M12 **female** 4pin D coded straight connector, and RJ45 connector (fits into std. Ethernetport)	**WI1046-M12M4S05NRJ45**	
X	X			Ethernet cable with M12 **female** 4pin D coded straight connector, loose ends.	**WI1046-M12M4S05R**	
X	X			Same as above but 15m (590 inch)	**WI1046-M12M4S15R**	
Protection caps. Optional if connector is not used to protect from dust / liquids.						
	X	X		IP67 protection cap for M12 **female** connector.	**WI1000-M12FCAP1**	
X			X	IP67 protection cap for M12 **male** connector.	**WI1000-M12MCAP1**	

Important: Please note that the cables are a standard type. They are not recommended for use in cable chains or where the cable is repeatedly bent. If this is required, use a special robot cable (2D or 3D cable).

3 MAC00-EC4 EtherCAT® module

3.1 Introduction to EtherCAT®

Expansion module MAC00-EC4 front plate

3.1.1 Intro to EtherCAT®.

EtherCAT® is a Real Time Ethernet technology which aims to maximize the use of the 100 Mbit, full duplex Ethernet bandwidth. It overcomes the overhead normally associated with Ethernet by employing "on the fly" processing hardware.
An EtherCAT® net consists of a master system and up to 65535 slave devices, connected together with standard Ethernet cabling.

The slave devices process the incoming Ethernet frames directly, extract or insert relevant data and transfer the frame to the next slave device, with a delay of approx. 4μs. The last slave device in the bus segment sends the processed frame back, so that it is returned by the first slave to the master as a kind of response frame.

There are several protocols that can be used as the application layer. In the CANopen over EtherCAT® (CoE) technology, the CANopen protocol is applied to EtherCAT®. CANopen defines Service Data Objects (SDO), Process Data Objects (PDO) and the Object Dictionary structure to manage the parameters. Further information about EtherCAT®, is available from the EtherCAT® technology group http://www.ethercat.org.

3.1 Introduction to EtherCAT®

3.1.2 Abbreviations

Following general used terms are usefull to know before reading the following chapters.

100Base-Tx	100 MBit Ethernet on twisted pairs
CAN	Controller Area Network
CANopen	Application layer protocol used in automation.
CoE	CANopen over EtherCAT®.
DC	Distributed Clock
EMCY	Emergency Object.
EoE	Ethernet over EtherCAT®.
ESI	EtherCAT® Slave Information
ESC	EtherCAT® Slave Controller
ETG	EtherCAT® Technology Group
EtherCAT®	Ethernet Control Automation Technologie
IP	Internet Protocol - IP address ~ the logical address of the device, which is user configurable (not used in EtherCAT®).
MAC	Media Access Controller - MAC address ~ the hardware address of the device (not used in EtherCAT®)
PDO	Process Data Object (for cyclic data)
SDO	Service Data Object (for acyclic data)
SII	Slave Infirmation Interface
XML	eXtensible Markup Language - used for the ESI file.

3.2 Protocol specifications

3.2.1 EtherCAT® - communication

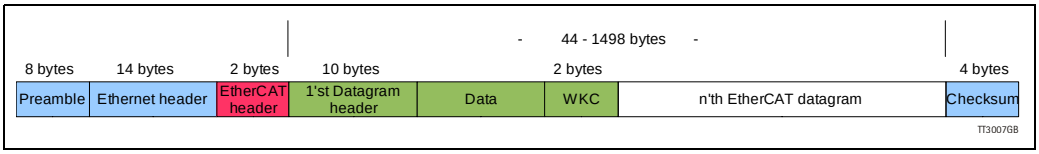
The EtherCAT® fieldbus system is standardised by the EtherCAT® user organisation (ETG). The driving force behind this is the german company, Beckhoff GmbH. Due to the advanced Ethernet technology used for EtherCAT®, in the future, customers can change from other fieldbus systems to EtherCAT® or generally equip new plant models with EtherCAT®.
Communication on EtherCAT® is based on a master/slave operation. The update cycle between master and slave depends on the number of EtherCAT® slaves, the amount of process data of the individual slaves, and the set update time of the master. Due to the ring topology, in every bus cycle only one telegram is sent on the bus. The bus cycle time thus remains exactly the same in every cycle.
Slave addressing can be done in two ways:

- Auto increment addressing
- Fixed node addressing

With Auto increment addressing the master scans the net for slaves, and the slaves are then addressed in the sequence they are physically present on the net. With fixed node addressing, the addresses that each node has programmed, is used.

3.2.2 EtherCAT® frame structure

In EtherCAT®, the data between the master and the slaves is transmitted in Ethernet frames. An EtherCAT® Ethernet frame consists of one or several EtherCAT® telegrams, each addressing individual devices and/or memory areas. The telegrams can be transported either directly in the data area of the Ethernet frame or within the data section of a UDP datagram transported via IP. The EtherCAT® frame structure is pictured in the following figure. Each EtherCAT® telegram consists of an EtherCAT® header, the data area and a working counter (WKC), which is incremented by all EtherCAT® nodes that are addressed by the telegram and have exchanged associated data.



3.2.3 Sync managers

Sync managers control the access to the application memory. Each channel defines a consistent area of the application memory. The adapter module has four sync manager channels. The mailbox protocol (SDO's) and process data (PDO's) are described later in this chapter.

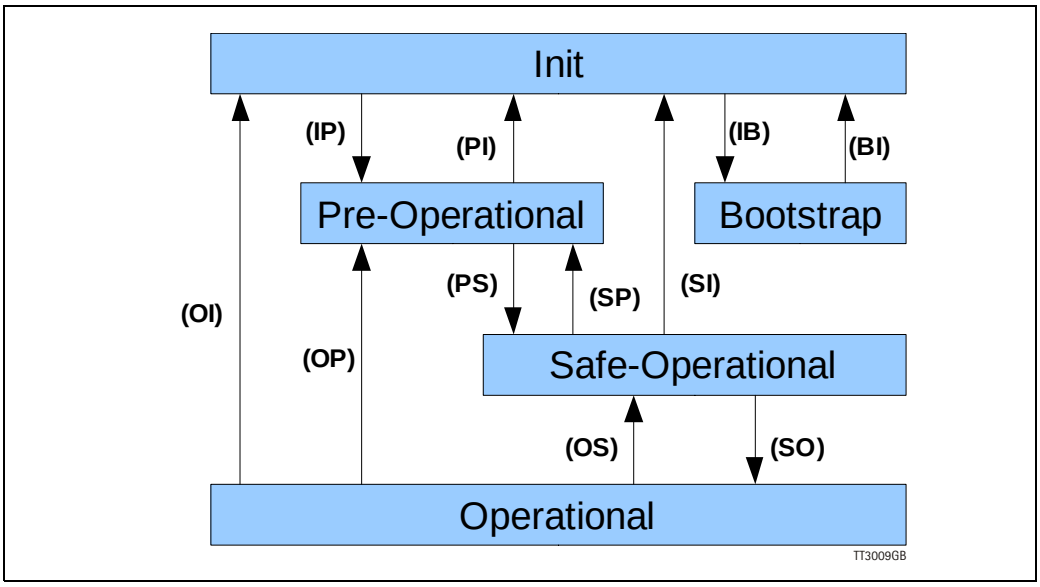
3.2.4 Sync manager watchdog

The sync manager watchdog monitors the output sync managers. If the output data is not updated by the EtherCAT® master within the configured time, the watchdog will activate time out and change the state of the adapter module from Operational to Safe-Operational.
Note: EtherCAT® has been designed so that it provides no way for a slave to monitor the connection to the master if the slave gets no output data.
Note: The drive reaction to a communication fault must be configured in the module write flag register (object 2011 subindex 6 - motor set passive or motor set velocity =0).

3.2 Protocol specifications

3.2.5 EtherCAT® - State machine

Both the master and the slaves have a state machine with the states shown below. After boot the slaves are in INIT state, and then it's up to the master to request state transitions. The standardized EtherCAT® state machine is defined in the following figure. The bootstrap state is not supported.



The module enters the Init state directly after start-up. After this, the module can be switched to the Pre-Operational state. In the Pre-operational state the EtherCAT® mailbox communication is allowed and CoE objects can be accessed by SDOs. After the master has configured the slave, it can switch the module to the Safe-Operational state. In this state input I/O data (PDOs) is sent from the adapter module to the EtherCAT® master, but there is no output I/O data from the master to the module. To communicate output I/O data the master must switch the adapter module to the Operational state.

State description table:

State	Description
Init	State after device initialisation. No Application layer communication (no SDO and PDO communication).
Pre-operational	SDO communication possible. No PDO communication.
Safe-operational	Transmit PDO operational (drive sends data to master)
Operational	Drive fully operational, responds to data via receive PDO
Boot-strap	Not used.

3.2.6 CANopen over EtherCAT®

The application layer communication protocol in EtherCAT® is based on the CANopen DS 301 communication profile and is called CANopen over EtherCAT® (CoE). The protocol specifies the Object Dictionary in the adapter module, in addition to communication objects for exchanging cyclic process data and acyclic messages. In addition to DS301 and the default JVL profile, the MAC00-ECx also supports the DSP402 drive profile See chapter *3.5 CiA® DSP-402 drive profile, page 43*.

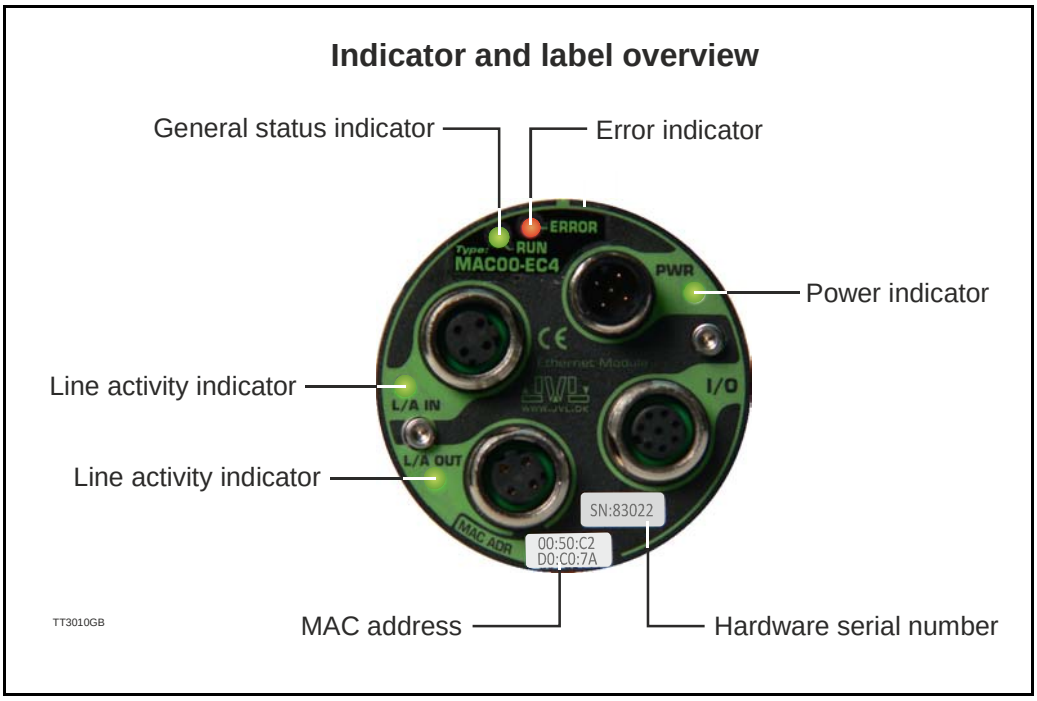
The EtherCAT® module uses the following message types:

- Process Data Object (PDO). The PDO is used for cyclic I/O communication, in other words, process data.
- Service Data Object (SDO). The SDO is used for much slower acyclic data transmission.
- Emergency Object (EMCY). The EMCY is used for error reporting when a fault has occurred in the module or in the drive.

3.3 Commisioning

3.3.1 Indicator LED's - description.

The LED's are used for indicating states and faults of module. There is one power LED, two link/activity LED's (one for each Ethernet connector), and 2 status LED's.



LED indicator descriptions

LED Text	Colour	Constant off	Constant on	Blinking	Single flash	Double flash	Flickering
L/A IN	Green	No valid Ethernet connection.	Ethernet is connected.	-	-	-	Activity on line
L/A OUT	Green	No valid Ethernet connection.	Ethernet is connected.	-	-	-	Activity on line
RUN	Green	Device state = INIT	Device state = Opera-tional	Device state = Pre-operational	Device state = Safe-opera-tional	-	-
ERROR	Red	No error	Critical com-munication or controller error	General configura-tion error	Local error	Process data watchdog timeout / EtherCAT® watchdog timeout	Booting error
PWR	Green	Power is not applied.	Power is ap-plied to both motor and module.	-	-	-	Power is applied to module but no communi-cation with motor.

Notes:
Blinking: Flashing with equal on and off periods of 200ms (2.5Hz). **Single flash:** Repeating on for 200ms and off for 1s. **Double flash:** Two flashes with a period of 200ms followed by 1s off period. **Flickering:** Rapid flash-ing with a period of approx. 50ms (10 Hz).

3.3.2 Quick start with TwinCAT.

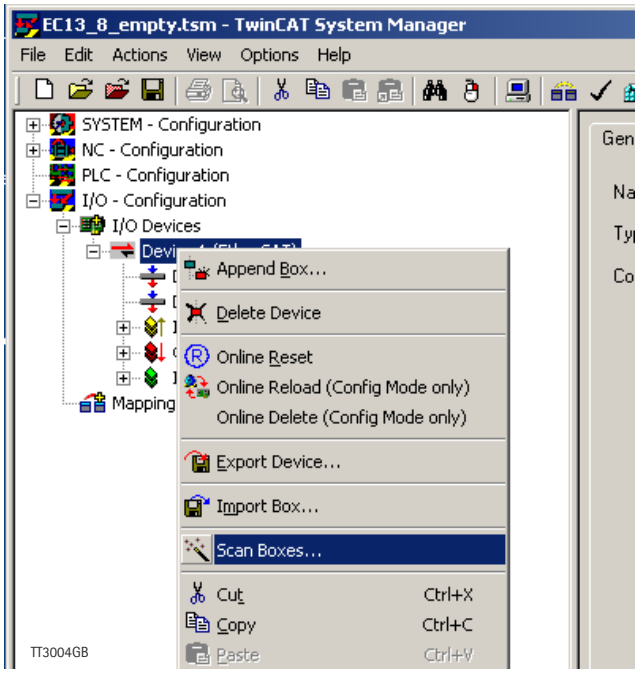
1. Copy the Ethernet slave information file (“JVL ECS V13.XML”) to the folder “..\Twincat\IO\Ethernet\” on the master PC.

2. Apply power, and make sure the *PWR* (power) LED is lit.

3. Connect the Ethernet cable from Master to the L/A IN connector, and check that the corresponding LED is lit.

4. Start TwinCAT - system manager on the master, and make sure that a proper Ethernet I/O device is appended (consult your TwinCAT manual).

5. Right click the I/O device, and select "scan boxes".



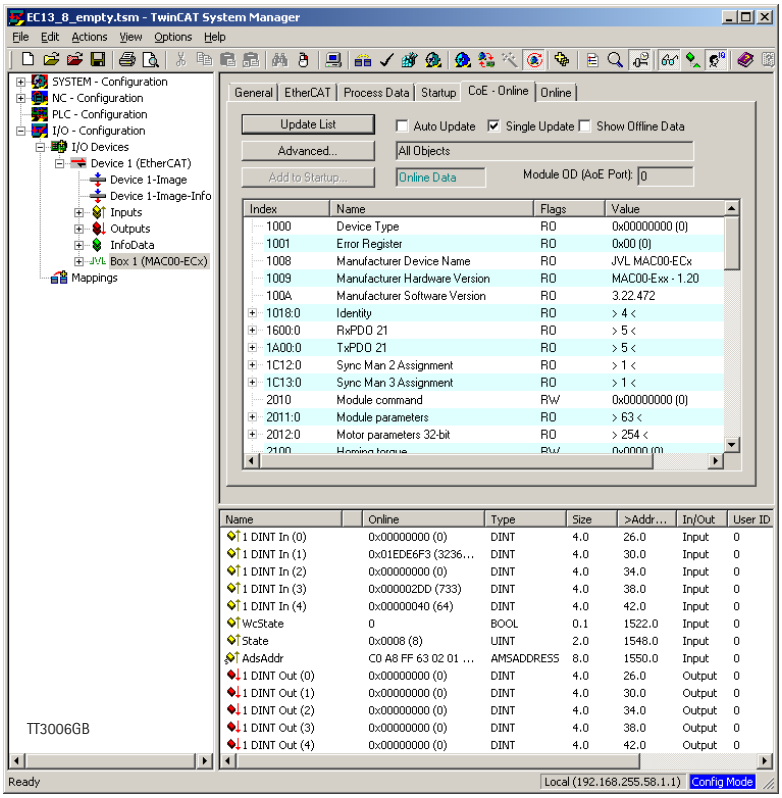
Continued next page

6. The device should now appear in the left side of the TwinCAT window, with a tiny JVL logo.

7. Press F4 (Reload I/O devices), and select the JVL device on the left side of the window.

8. The "L/A IN" LED should now be flashing and the process data should now appear on the bottom right side of the TwinCAT window.

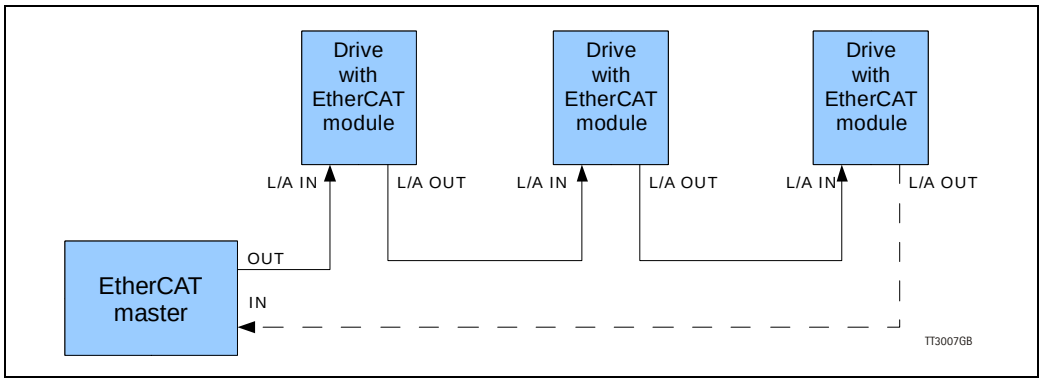
9. By pressing the "CoE online" tab, it's possibly to inspect the CANopen objects, and modify motor and module parameters.



10. If DSP402 drive profile is selected the JVL device is named "Drive" instead of "Box" as shown in the picture.

3.3.3 Mechanical installation

The network cables must be connected to the two M12 connectors (marked "L/A IN" and "L/A OUT") on the module. The cable from the EtherCAT® master is always connected to the "L/A IN" port. In the line topology, if there are more slave devices in the same line, the next slave device is connected to the port marked "L/A OUT". If there is a redundant ring, the right "L/A OUT" port of the last slave device is connected to the second port of the EtherCAT® master. See the figure below. Standard CAT 5 FTP or STP cables can be used. It is not recommended to use UTP cables in industrial environments, which is typically very noisy.



3.4 EtherCAT® objects

3.4.1 Process Data Object 21 (PDO/JVL Profile)

PDO's (Process Data Objects) are used for cyclic transfer of time-critical process data between master and slaves. There is one receive PDO and one transmit PDO which is fully user configurable. Tx PDOs are used to transfer data from the slave to the master and Rx PDOs to transfer data from the master to the slave.It is possibly to set up five or eight, 32 bit registers in each PDO, depending on the configuration (See chapter *3.4.11 Object 0x2011 - Subindex 6 Setup bits, page 39*).
The setup is done with MacTalk or via SDO object 0x2011 subindex 16-31. It requires a save in flash and a power cycle before the new configuration are used. If the configuration of the PDO's, is not altered by the user, the MAC00-EC4 module uses the default mapping shown in the tables below.

If module registers is placed in cyclic R/W, then the register number has to be calculated as follows:

Register number = 65536 x sub index.
Example: module command (sub-index 15) = 65536 x 15 = register **983040**

When module registers (register numbers above 65535) are chosen, they **have** to be placed **after** the motor registers in the list of cyclic registers.

NB! If an index is set to zero (No selection), then the following indexes is discarded. Thereby computing resources in the drive are released, which makes much faster cycle times possibly. Please see next paragraph.

Default registers in transmit PDO 21 (Slave > Master)

Object index	Register no.	Motor register short	Motor register description
0	2	MODE_REG	Operating mode
1	10	P_IST	Actual position
2	12	V_IST	Actual velocity
3	169	VF_OUT	Actual torque
4	35	ERR_STAT	Status bits
5	-	-	-
6	-	-	-
7	-	-	-

Default registers in receive PDO 21 (Master > Slave)

Object index	Register no.	Motor register short	Motor register description
0	2	MODE_REG	Operating mode
1	3	P_SOLL	Target position
2	5	V_SOLL	Maximum velocity
3	7	T_SOLL	Maximum torque
4	-	-	-
5	-	-	-
6	-	-	-
7	-	-	-

Please notice: Even though all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

3.4.2 Minimum cycle time (JVL Profile)

The minimum cycle time is the minimum amount of time between each cyclic request (PDO) on the Ethernet.
If the module is mounted in MAC050-MAC141 it is possible to add a poll division factor either in the EtherCAT tab in Mactalk or manually in module register 8 (See chapter *3.4.13 Object 0x2011 - Subindex 8 Poll division factor, page 40*).

The positions 6-8 is only transferred if enabled, See chapter *3.4.11 Object 0x2011 - Sub-index 6 Setup bits, page 39*.

If operating with values lower than those listed, data loss will occur.

No. of motor registers transmitted in each direction	Motor series MAC050 - MAC141	Motor series MAC400 to MAC3000
1/1	4mS *	360µS *
2/2	8mS *	395µS *
3/3	12mS *	430µS *
4/4	16mS *	465µS *
5/5	20mS *	500µS *
6/6	24mS *	535µS *
7/7	28mS *	570µS *
8/8	32mS *	605µS *

* The minimum cycle times, is only valid if not sending any acyclic requests while in any operating mode. MODULE registers can be appended as the last registers in the list, at no extra timing cost. Motor register 35 shall be in the cyclic read list, as it is also used internally.

3.4.3 Service Data Objects (SDO)

Service Data Objects (SDOs) are mainly used for transferring non time-critical data, for example, identification, configuration and acyclic data.

3.4.4 Emergency Objects

Emergency Objects (EMCYs) are used for sending fault information from the communication module and the drive to the EtherCAT® network. They are transmitted whenever a fault occurs in the drive or in the module. Only one Emergency Object is transmitted per fault. EMCYs are transmitted via SDO's.

The following error codes can be generated:

Errorcode 0x1001: Generic error - Motor error
Errorcode 0x1003: Generic error - Internal communication error

When the error is no longer present, the module will send a NoError EMCY object once.

The EMCY object 1001h is sent as an 8 byte message, and has the following structure:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANopen® error code: MSB (0x10)	CANopen® error code: LSB (0x01)	8-bit error Register = object 0x1001	MAC motor ERR_STAT LSB	MAC motor ERR_STAT	MAC motor ERR_STAT	MAC motor ERR_STAT MSB	Reserved

The EMCY object 1003h is sent as an 8 byte message, and has the following structure:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CANopen® error code: MSB (0x10)	CANopen® error code: LSB (0x01)	8-bit error Register = object 0x1001	Reserved	Reserved	Reserved	Reserved	Reserved

3.4.5 Object Dictionary

An important part of the CoE protocol is the Object Dictionary, which is different objects specifying the data layout. Each object is addressed using a 16-bit index and possibly a sub index. There are some mandatory objects and some manufacturer specific objects. The objects in the CoE Object Dictionary can be accessed with SDO services.

3.4.6 Mandatory objects:

Name	Index (hex)	Sub Index	Data Type	Read only	Default	Description
Device type	1000		UNSIGNED32	X	0x0	Contains information about the device type.
Error Register	1001		UNSIGNED8	X		This is the mapping error register, and it is part of the emergency object. If some of the sub index are high, an error has occured. See also See chapter *3.4.4 Emergency Objects, page 35*. Mandatory
		0				Generic error. Mandatory
		1				Current
		2				Voltage
		3				Temperature
		4				Communication (Overrun)
		5				Device profile specific
		6				Reserved
		7				Manufactor specific
Manufacturer device name	1008		VISIBLE STRING	X	JVL - MAC00-ECx	
Manufacturer hardware version	1009		VISIBLE STRING	X	1.0	
Manufacturer software version	100A		VISIBLE STRING	X	1.0	Example: Version x.x
Identity object	1018		IDENTITY	X		Contain general information about the module
		0	1..4	X	4h	Number of entries. Mandatory
		1	UNSIGNED32	X	0x0117	Vendor ID, contains a unique value allocated to each manufactor. 117h is JVLs vendor ID. Mandatory.
		2	UNSIGNED32	X	0x0200	Product Code, identifies a specific device version. The MAC00-EC4 has the product code 200h
		3	UNSIGNED32	X	0x20020	Revision number.
		4	UNSIGNED32	X		Serial number

3.4 EtherCAT® objects

3.4.7 Manufacturer specific objects.

The manufacturer specific objects, provides access to all module registers, and all motor registers, as well as a module command object.

	Index (hex)	Sub Index	Type	Read only	Default	Description
Module command	2010	0	UNSIGNED32			Module command object. See possible commands below.
Module parameters	2011	0	UNSIGNED8	X	63	Subindex count
		1	UNSIGNED32	X		High 16 bit of MAC address (placed in low 16 bit of word)
		2	UNSIGNED32	X		Low 32 bit of MAC address
		3	UNSIGNED32			IP address
		4	UNSIGNED32			Net mask
		5	UNSIGNED32			Gateway
		6	UNSIGNED32		0x0	Setup bits
		7	UNSIGNED32		0	Digital outputs on module
		8	UNSIGNED32		0	Poll division factor
		9	UNSIGNED32		0	Station alias
		10	UNSIGNED32		-	Reserved for future use
		11	UNSIGNED32		-	Input mask
		12-14	UNSIGNED32		-	Reserved for future use
		15	UNSIGNED32		-	Command register
		16	UNSIGNED32		2	Register no. to place in TxPDO 21, position 1.
		17	UNSIGNED32		10	Register no. to place in TxPDO 21, position 2.
		18	UNSIGNED32		12	Register no. to place in TxPDO 21, position 3.
		19	UNSIGNED32		169	Register no. to place in TxPDO 21, position 4.
		20	UNSIGNED32		35	Register no. to place in TxPDO 21, position 5.
		21	UNSIGNED32		-	Reserved for future use
		22	UNSIGNED32		-	Reserved for future use
		23	UNSIGNED32		-	Reserved for future use
		24	UNSIGNED32		2	Register no. to place in RxPDO 21, position 1.
		25	UNSIGNED32		3	Register no. to place in RxPDO 21, position 2.
		26	UNSIGNED32		5	Register no. to place in RxPDO 21, position 3.
		27	UNSIGNED32		7	Register no. to place in RxPDO 21, position 4.
		28	UNSIGNED32		0	Register no. to place in RxPDO 21, position 5.
		29	UNSIGNED32		-	Reserved for future use
		30	UNSIGNED32		-	Reserved for future use
		31	UNSIGNED32		-	Reserved for future use
		32	UNSIGNED32	X	-	Module serial no.
		33	UNSIGNED32	X	-	Module hardware version
		34	UNSIGNED32	X	-	Module software version
		35	UNSIGNED32	X	-	No. of internal motor communication timeouts
		36	UNSIGNED32	X	-	No. of retry frames to motor
		37	UNSIGNED32	X	-	No. of discarded frames to motor
		38	UNSIGNED32	X	-	Total no. of frames to motor
		39-46	UNSIGNED32	X	-	Reserved for future use
		47	UNSIGNED32	X	-	Digital inputs on module
		48	UNSIGNED32	X	-	Status bits
		49-63				Reserved for future use
Motor parameters	2012	0	UNSIGNED8	X	254	Subindex count
		N	UNSIGNED32			Access to the motor parameter n

Note: Module parameters are not automatically saved to permanent memory after a change. The parameters can be saved permanently by applying a "Save parameters to flash" command afterwards.

3.4.8 Object 0x2010 - Subindex 0

This object is used for sending commands to the module and is write only. The possible commands are listed in the table below.

Command no.			Command description	
	Hex	Dec	MAC050 - MAC141	MAC400 – MAC3000
Module only commands				
	0x 0000 0000	0	No operation	< Same as
	0x 0000 0001	1	Reset the module	< Same as
	0x 0000 0010	16	Save module parameters to flash	< Same as
Synchronized commands				
	0x 0000 0101	257	Simultaneous reset of the motor and the module	< Same as
	0x 0000 0110	272	Save the motor parameters in flash memory, and do a re-sync. of internal communication afterwards.	< Same as
Motor only normal commands *(via module cmd register)*				
	0x 8000 0001	2147483649	Reset motor *(not recommended, use synchronized version instead)*.	< Same as
	0x 8000 0002	2147483650	Save motor parameters in flash and reset motor *(not recommended, use synchronized version instead)*.	< Same as
Motor only FastMac commands *(via module cmd register)*				
	0x8000 00E0	2147483872	No operation	< Same as
	0x8000 00E1	2147483873	Reset error *(Clear error bits in motor register 35)*	< Same as
	0x8000 00E2	2147483874	P_SOLL = 0	< Same as
	0x8000 00E3	2147483875	P_IST = 0	< Same as
	0x8000 00E4	2147483876	P_FNC = 0	< Same as
	0x8000 00E5	2147483877	V_SOLL = 0	< Same as
	0x8000 00E6	2147483878	T_SOLL = 0	< Same as
	0x8000 00E7	2147483879	Reset IN_POS, AC C,DEC	< Same as
	0x8000 00E8	2147483880	P_FNC = (FLWERR - P7) * 16	< Same as
	0x8000 00E9	2147483881	P_FNC = (FLWERR - P8) * 16	< Same as
	0x8000 00EA	2147483882	Reserved	< Same as
	0x8000 00EB	2147483883	Reserved	< Same as
	0x8000 00EC	2147483884	Activate P1,V1,A1,T1,L1,Z1	< Same as
	0x8000 00ED	2147483885	Activate P2,V2,A2,T2,L2,Z2	< Same as
	0x8000 00EE	2147483886	Activate P3,V3,A3,T3,L3,Z3	< Same as
	0x8000 00EF	2147483887	Activate P4,V4,A4,T4,L4,Z4	< Same as
	0x8000 00F0	2147483888	Start search zero	< Same as
	0x8000 00F1	2147483889	P_SOLL = P_IST + P7;	P_SOLL = P_IST + P7 – FLWERR;
	0x8000 00F2	2147483890	P_SOLL = P_IST + P8;	P_SOLL = P_IST + P8 – FLWERR;
	0x8000 00F3	2147483891	Reserved	< Same as
	0x8000 00F4	2147483892	Select absolute position mode	< Same as
	0x8000 00F5	2147483893	Select relative position mode using P_SOLL	< Same as
	0x8000 00F6	2147483894	Select relative position mode using P_FNC	< Same as
	0x8000 00F7	2147483895	Synchronize position manually using absolute new values. P_IST = P_NEW; P_SOLL = P_NEW; P_FUNC = P_NEW * 16;	Synchronize position manually using absolute new values. P_IST = P_NEW; P_SOLL = P_NEW; P_FNC = (P_NEW + FLWERR)*16;
	0x8000 00F8	2147483896	Synchronize position manually using relative new values. (basically offset the position range with the value of P_NEW). P_IST = P_IST + P_NEW; P_SOLL = P_SOLL + P_NEW; P_FUNC = P_FUNC + (P_NEW * 16);	< Same as
	0x8000 00F9	2147483897	No operation	< Same as
	0x8000 00FA	2147483898	No operation	< Same as
	0x8000 00FB	2147483899	No operation	< Same as
	0x8000 00FC	2147483900	No operation	< Same as
	0x8000 00FD	2147483901	Reserved	< Same as
	0x8000 00FE	2147483902	Reserved	< Same as
	0x8000 00FF	2147483903	Reserved	< Same as

3.4.9 Object 0x2011

The module registers is mapped to object 0x2011. The subindex 3-31 is R/W, the rest is read only.

3.4.10 Object 0x2011 - Subindex 1-5

Reserved for future use.

3.4.11 Object 0x2011 - Subindex 6 Setup bits

This register is used to setup the module configuration and how the module should react on different events.

Bit	7-31	6	5	4	3	2	1	0
Output	Reserved	PDO - 8 registers	Mirror registers	Endless relative	Enable drive profile	Clear "Name of station"	Disable error handling	Ethernet error handling.

Ethernet error handling
0 = Set motor to passive mode when error occurs.
1 = Set velocity to 0 when error occurs (active brake).

Disable error handling
0 = Ethernet error handling enabled.
1 = Ethernet error handling disabled.

Clear "Name of station"
Only applicable to Profinet protocol.

Enable drive profile
0 = JVL drive profile.
1 = CiA® DSP-402 drive profile enabled. *Requires a save in flash and a power cycle to be activated.*

Endless relative
0 = Endless relative disabled.
1 = Endless relative enabled. If relative mode is selected in the control word, then the actual position never changes. When selecting this mode absolute positioning can no longer be used. *This bit only applies for DSP-402 profile.*

Mirror registers
Only applicable to ModbusTCP protocol.

PDO - 8 registers
0 = 5 x 32 bit registers in each PDO.
1 = 8 x 32 bit registers in each PDO. *Requires a save in flash and a power cycle to be activated. Only applicable to JVL profile (not DSP-402).*

3.4.12 Object 0x2011 - Subindex 7 Digital outputs on module

With this object the digital outputs can be controlled.
The value written to this object is directly shown on the digital outputs.

Bit	2-31	1	0
Output	Reserved	Output2* (O2)	Output1* (O1)

* The availability of the outputs depends on the actual version of the module used. Example MAC00-EC4 only support Output 1 (O1).

3.4.13 Object 0x2011 - Subindex 8 Poll division factor

With this object a poll division factor can be set. This enables use of cycle times faster than the motor is capable of. If for example having a MAC050-141 and 5 cyclic write and 5 cyclic read registers, then a minimum cycle time of 20ms is needed. Instead it is possible to have a net cycle time of 1ms, and a poll division factor of 20. Then the motor internally only get updated every 20ms.

Bit	16-31	0 - 15
R/W	Reserved	Poll division factor

Only applicable for MAC050-141. Only read at power-up, or after reset. So in order to change the value, first change this value, then issue a "save in flash" command, then reset the module.

3.4.14 Object 0x2011 - Subindex 9 Station alias (node number)

With this object a station alias (node number) is set manually.

Bit	16-31	0 - 15
R/W	Reserved	Station alias

Only read at power-up, or after reset. So in order to change the value, first change this value, then issue a "save in flash" command, then reset the module.

3.4.15 Object 0x2011 - Subindex 11 Input mask

This register is used to setup input mask on the digital inputs (IN1-4).

Bit	16-31	15-12	11-8	7-4	3-0
Output	Reserved	Reserved	PL mask	Reserved	NL mask

NL mask Bit set results in that corresponding input is configured as Negative Limit switch. Bit 0-3 corresponds to IN1-4.
PL mask Bit set results in that corresponding input is configured as positive Limit switch. Bit 8-11 corresponds to IN1-4.

3.4.16 Object 0x2011 - Subindex 15 Command register

Analogue to writing to object 0x2010. But this can be mapped in the RxPDO 21 if desired.

3.4.17 Object 0x2011 - Subindex 16-23 Register no. to place in TxPDO 21

These registers contain the numbers that define the registers which are in the TxPDO 21. That is the register's, which is transmitted from slave to master cyclically. If some of these registers are changed, it is necessary to issue a "save in flash" command and to reboot the device before the changes take effect.

3.4.18 Object 0x2011 - Subindex 24-31 Register no. to place in RxPDO 21

These registers contain the numbers that define the registers which are in the RxPDO 21. That is the register's, which is transmitted from master to slave cyclically. If some of these registers are changed, it is necessary to issue a "save in flash" command and to reboot the device before the changes take effect.

3.4.19 Object 0x2011 - Subindex 32-38

These registers contain HW, SW and communication information of the module.

3.4.20 Object 0x2011 - Subindex 47 Digital inputs on module

With this object the status of the 4 digital inputs can be read.

Bit	4-31	3	2	1	0
Input	Reserved	IN4*	IN3*	IN2*	IN1*

* The availability of the inputs depends on the actual version of the module used. Example MAC00-EC4 only support Input 1 (IN1).

3.4.21 Object 0x2011 - Subindex 48 Status bits

This register is used for miscellaneous information about the module.

Bit	8-31	7	0-6
Output	Reserved	1=No communication with the motor	Reserved

3.4.22 Object 0x2012

Object 0x2012 are for acyclic view or change of motor registers.
Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

3.4.23 EtherCAT® Slave Information file

EtherCAT® Slave Information file (ESI) is a XML file that specify the properties of the slave device for the EtherCAT® master and contains information on the supported communication objects. EtherCAT® Slave Information files for JVL drives are available through your local JVL representative. If TwinCAT is used for master then the XML-file shall be copied to the folder "..\TwinCAT\Io\EtherCAT\".

3.5 CiA® DSP-402 drive profile

3.5.1 Introduction

The MAC00-ECx supports the DSP-402 standard from CiA® http://www.can-cia.com/. Please refer to this standard for full details of the functions. The DSP-402 is only a standard proposal and might be changed in the future. We reserve the right to change future firmware versions to conform to new versions of the standard. Not all of the functionality, described in DSP-402, is supported. But all the mandatory functions are supported. The following operation modes are supported:

- Profile position mode
- Velocity mode
- Homing mode

Preconditions:
Before the DSP-402 mode can be used, the firmware in the MAC00-ECx module must be updated to at least version 3.22. Besides, version 13 of the XML file must be used "JVL ECS V13.xml" found on the web page http://www.jvl.dk.

- The start mode of the motor must be set to passive.
- No power up Zero searches must be selected.
- If absolute movement is used, the 'resynchronize after passive mode' must be set.
- The DSP-402 drive profile must be enabled and saved to flash (please see next paragraph).

When using DSP-402 mode, manipulating motor parameters with object 0x2012 can corrupt the behavior of the DSP-402 functions. Also be aware that manipulating parameters in MacTalk should be avoided when using DSP-402.

3.5.2 Selecting DSP-402 drive profile

As default the MAC00-ECx uses the JVL profile. In order to use the DSP-402 drive profile instead, it is selected in this way:
In MacTalk in the Ethernet tab the checkbox "Enable DSP402 drive profile" is checked, and the "Apply and save" button is pressed. Then after a power cycle the MAC00-ECx will wake up with DSP-402 drive profile enabled instead of the standard JVL profile.
If already having a TwinCAT project, then delete the JVL box, and do a new scan for boxes. Now the JVL device will appear as a drive instead.

3.5.3 Supported objects

Most of the DSP402 parameters start up in the module with default values. A few of them are set depending on the motor type the module is mounted in - either MAC50-141 or MAC400, 800, 1500, or 3000. None of the parameters can be saved to flash in the module. The following table shows the additional object dictionary defined for DSP-402 support.

Continued next page

3.5 CiA® DSP-402 drive profile

Index (hex)	Sub idx.	Name	Type	Attributes	Default value
Device data					
6402	0	Motor type	U16	RO	10
6403	0	Motor catalogue number	Str.	RO	MACxxx
6404	0	Motor manufacturer	Str.	RO	JVL Industri Elektronik A/S
6405	0	http motor catalogue address	Str.	RO	www.JVL.dk
6502	0	Supported drive modes	U32	RO	0x00000025
6503	0	Drive catalogue number	Str.	RO	MACxxx
6504	0	Drive manufacturer	Str.	RO	JVL Industri Elektronik A/S
6505	0	http drive catalogue address	Str.	RO	www.JVL.dk
Digital I/O					
60FD	0	Digital inputs	U32	RO, P	-
60FE	0	Digital outputs	U8	RO, P	2
	1	Physical outputs	U32	RW, P	0
	2	Bit mask	U32	RW, P	3
Device control					
6040	0	Control word	U16	RW, P	-
6041	0	Status word	U16	RW, P	-
605A	0	Quick stop option code	I16	RW	2
6085	0	Quick stop deceleration	U32	RW	50000
6060	0	Modes of operation	I8	RW, P	-
6061	0	Modes of operation display	I8	RO, P	-
6072*	0	Max torque	U16	RW, P	1000
6073**	0	Max current	U16	RW	-
6075**	0	Rated current	U32	RW	9000
607E	0	Polarity	U8	RW	0
Profile position parameters					
6064	0	Position actual value	I32	RO, P	-
6067	0	Position window	U32	RW	100
6068	0	Position window time	U16	RW	6
607A	0	Target position	I32	RW, P	-
607D	0	Software position limit	U8	RO	2
	1	Min.	I32	RW	0
	2	Max.	I32	RW	0
6080	0	Max motor speed	U32	RW	*Depending on motor type*
6081	0	Profile velocity	U32	RW, P	100
6083	0	Profile acceleration	U32	RW, P	15000
6086	0	Motion profile type	I16	RW	0

Continued next page

3.5 CiA® DSP-402 drive profile

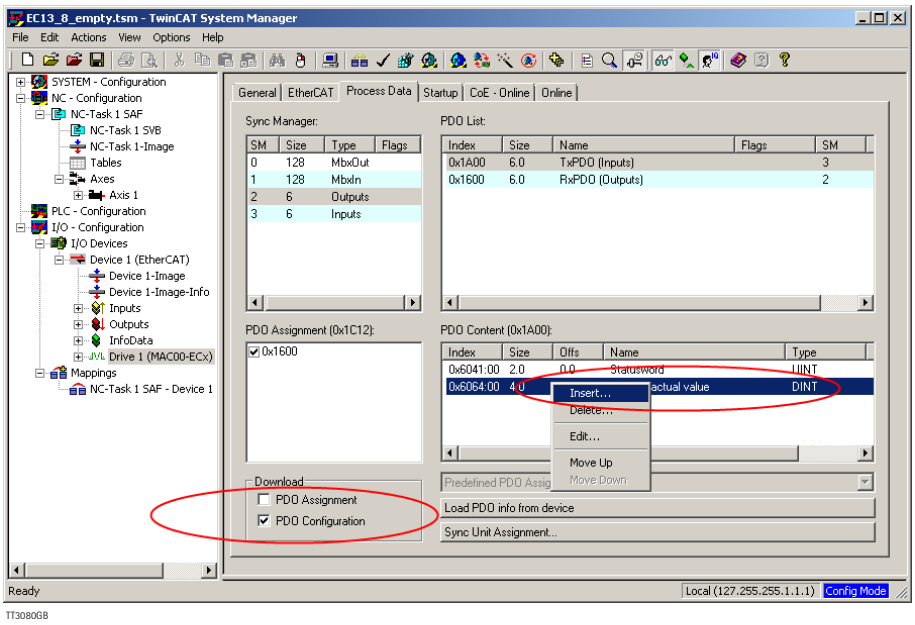
Index (hex)	Sub idx.	Name	Type	Attri-butes	Default value
Profile velocity mode					
606B	0	Velocity demand value	I32	RO, P	-
606C	0	Velocity actual value	I32	RO, P	-
606D	0	Velocity window	U16	RW	100
606E	0	Velocity window time	U16	RW	6
60FF	0	Target velocity	U32	RW, P	-
Homing mode					
2100	0	Homing torque	U16	RW	30
607C	0	Home offset	I32	RW	0
6098	0	Homing method	I8	RW	0
6099	0	Homing speeds	U8	RO	2
	1	Speed during search for switch	U32	RW	50
	2	Speed during search for zero	U32	RW	50
609A	0	Homing acceleration	U32	RW	5000
Factors					
608F	0	Position encoder resolution	U8	RO	2
	1	Encoder increments	U32	RW	*Depending on motor type*
	2	Motor revolutions	U32	RW	1
6091	0	Gear ratio	U8	RO	2
	1	Motor revolutions	U32	RW	1
	2	Shaft revolutions	U32	RW	1
6092	0	Feed constant	U8	RO	2
	1	Feed	U32	RW	*Depending on motor type*
	2	Shaft revolutions	U32	RW	1

”Str” = String, ”I” = Integer, ”U” = Unsigned integer, figures = number of bits.
“RO” = Read Only, ”RW” = Read and Writeable, ”P” = PDO map able.
* Only available in MAC00-ECx.
** Only available in MISxxxECxx.

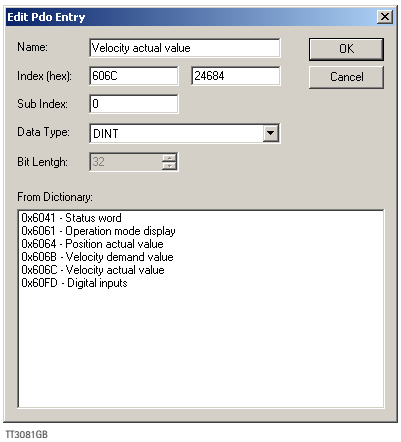
3.5.4 PDO’s (Process Data Objects)

When selecting the DSP-402 drive profile the setup and functioning of the PDO’s is very different from the default JVL profile. In the DSP-402 drive profile there is one PDO in each direction. Each PDO can hold up to eight objects and the PDO’s are fully dynamic and is altered in TwinCAT, in the “Process data” tab. Make sure the “CoE online” tab has been selected and shown first (this will take some time), in order for TwinCAT to retrieve all the objects from the MAC00-ECx device.

By right-clicking in the ”PDO Content” window a menu with options appear, and if pressing ”Insert” then a new window will open showing the possible objects to insert in the PDO.



By selecting an object and pressing “OK” then that object is inserted in the PDO and will be transferred to the MAC00-ECx device, at next “reload devices” if the “PDO configuration” checkbox is checked.



For further information about PDO configuration please consult the appropriate manual for the PLC system used.

3.5.5 Factors

Position factor
The position factor is the relation between the user unit and the internal position unit (counts). The position factor is automatically calculated when the feed constant (Object 0x6092) and gear ratio (Object 0x6091) are set.
Example:
We have a MAC motor with a 3.5:1 gear box connected to a belt drive. The diameter of the drive wheel is 12.4 cm. We want the unit of position to be in millimetres.
The circumference of the drive wheel is 389.56mm (124mm*pi). The parameters should be set as follows:

Object	Name	Value
0x6091 sub index 1	Gear ratio / Motor revolutions	35
0x6091 sub index 2	Gear ratio / Shaft revolutions	10
0x6092 sub index 1	Feed constant / Feed	38956
0x6092 sub index 2	Feed constant / Shaft revolutions	100

Please note that it is not necessary to set the encoder resolution. This is automatically set by the module.
Position factor formula:

$$Position\_factor = \frac{Gear\_ratio\_Motor\_rev.*Feed\_constant\_Shaft\_Rev.*Position\_encoder\_res.\_Encoder\_Increments}{Feed\_constant\_Feed*Feed\_constant\_Shaft\_rev.*Position\_encoder\_res.\_Motor\_rev.}$$

or as objects:

$$Position\_factor = \frac{Object\ 6091sub1*Object\ 6092sub2*Object\ 608Fsub1}{Object\ 6092sub1*Object\ 6092sub2*Object\ 608Fsub2}$$

The Position factor is calculated to in the above example:

$$Position\_factor = \frac{35*100*4096}{38956*10*1} = 36{,}8$$

The above example is for a MAC50-141. For MAC400, MAC1500 and MAC3000, the number 4096 shall be changed to 8192, for MAC800 the number is 8000.

3.5.6 Operation modes

Changing operation mode
A change of operation mode is only possible when the operation mode is not enabled. There are two exceptions and one is when changing from homing mode to profile position mode. This is possible when the homing sequence is completed and can be done even though the operation mode is enabled. The other exception is when changing from profile position mode into velocity mode.

Profile position mode
This mode can be used for positioning where a movement profile can be set up.
The acceleration and maximum velocity can be programmed.
In this mode, both absolute and relative moves are supported. The type of move is selected via bit 6 (abs/rel) in the status word. When a relative move is selected, the type of relative move is dependent on the setup in object 2011h sub index 6.
It is also possible to select different movement modes. This is done using bit 5 (change set immediately) in the status word. When this bit is 0 and a move is in progress, the new set-point is accepted. But the new set-point and profile are not activated before the previous movement is finished. When this bit is 1, the new set-point is activated instantly and the motor will move to the new position with the new profile parameters.

Please note:
- The torque limit that is used during the profile can be set via object 6072h.
- The register L1 (object 2012 subindex 81) is used to select the load factor when the profile is started. If a different load factor is required, this register must be set correctly.

Velocity mode
In this mode the motor runs at a selected velocity. A new velocity can be selected in object 0x60FF and the motor will then accelerate/decelerate to this velocity.
The maximum slippage error is not supported in this mode.

Please note:
- The torque limit that is used during the profile can be set via object 6072h.

Homing mode
In this mode different homing sequences can be initiated. The home sensor must be connected to the AIN input on the module. If end limit sensors are used during the homing sequence, then the sensors should be connected to the appropriate inputs, and they must be enabled via object 0x2011 sub index 11. In the MAC motors the module inputs is used.
In the MIS motors the registers 125 (I/O active level and I/O type), and 132 (home input mask) have to be correctly set up prior to use. Do this setup by object 0x2012 or in MacTalk in the 'I/O Setup' tab.

The torque limit used during homing is selected via object 0x2100. The unit of this object is the same as other torque objects, e.g. object 0x6072.

3.5 CiA® DSP-402 drive profile

The MAC00-ECx module and MIS34xxxxECxxx supports the following homing methods:

Method	Description	Available in MAC	Available in MIS
-4	Torque homing in positive direction.	X	-
-3	Torque homing in negative direction.	X	-
-2	Torque homing in positive direction and afterwards homing on the index pulse.	X	-
-1	Torque homing in negative direction and afterwards homing on the index pulse.	X	-
0-2	Not supported.	-	-
3	Homing on positive home switch and index pulse to the left.	X	-
4	Homing on positive home switch and index pulse to the right.	X	-
5	Homing on negative home switch and index pulse to the left.	X	-
6	Homing on negative home switch and index pulse to the right.	X	-
7	Start positive (unless home switch is active), reverse on home switch active, stop at index.	X	-
8	Start positive (unless home switch is active), stop at first index after active home switch.	X	-
9	Start positive, reverse on limit switch, stop at first index after active home switch.	X	-
10	Start positive, reverse on limit switch, reverse at homeswitch, stop at index.	X	-
11	Start negative (unless home switch is active), reverse on home switch active, stop at index.	X	-
12	Start negative (unless home switch is active), stop at first index after active home switch.	X	-
13	Start negative, reverse on limit switch, stop at first index after active home switch.	X	-
14	Start negative, reverse on limit switch, reverse at home switch, stop at index.	X	-
15-18	Not supported.	-	-
19	Homing on positive home switch.	X	X
20	Homing on positive home switch.	X	X
21	Homing on negative home switch.	X	X
22	Homing on negative home switch.	X	X
23	Start positive (unless home switch is active), stop at active home switch.	X	-
24	Start positive (unless home switch is active), stop at active home switch.	X	-
25	Start positive, reverse on limit switch, stop at active home switch.	X	-
26	Start positive, reverse on limit switch, stop at active home switch.	X	-
27	Start negative (unless home switch is active), stop at active home switch.	X	-
28	Start negative (unless home switch is active), stop at active home switch.	X	-
29	Start negative, reverse on limit switch, stop at active home switch.	X	-
30	Start negative, reverse on limit switch, stop at active home switch.	X	-

For a comprehensive description of the homing modes 3-34, please consult the CiA DSP402 version 3.0.

Please note that you should always use a home offset (object 0x607C) when using torque homing. This is to ensure that the motor moves away from the end limit. The sign of the home offset should be the opposite of the homing direction. For example, when using a negative homing direction, the home offset could be 5000.

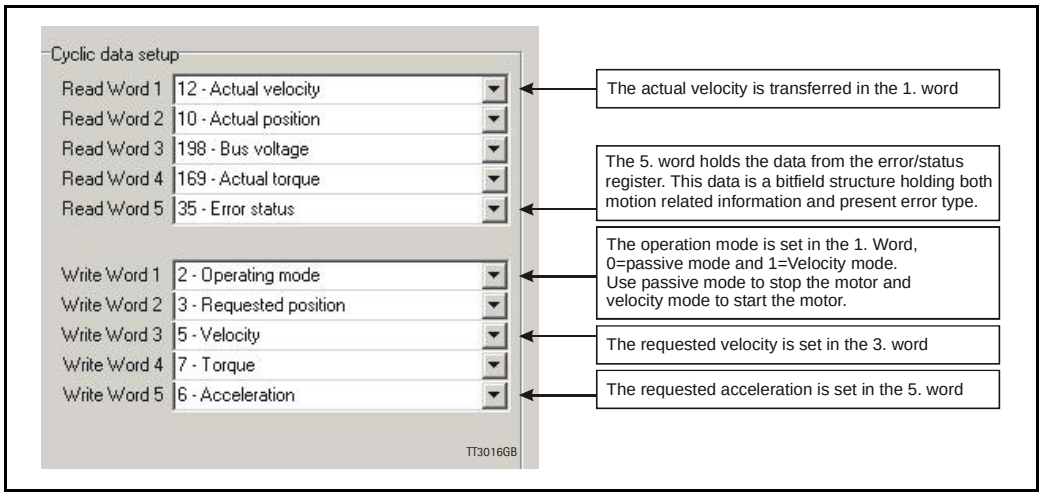
3.6 Examples

3.6.1 Running Velocity control (JVL Profile)

To use the JVL motor in velocity mode the following registers are basically of interest.

1. "Mode" - Mode register register 2
2. "V_SOLL" - Velocity register 5
3. "A_SOLL" - Acceleration register 6
4. "Error/Status" - Error and status register 35

So, to control these registers the cyclic data needs to configured.
From MacTalk the setup is configured as this.



With the settings illustrated above we initiate the velocity mode by writing 0x1 to the first word-value, this is velocity mode.

From the Master the registers is accessed using the PDO21 and accessing the registers R/W on words 1-5.

Since different PLC's have different methods of implementation the basic steps is described in the following.

1. Set the needed velocity. V_SOLL = V x 2.77 [rpm]
   Ex. We need the motor to run with a constant speed of 1200 RPM. So, V_SOLL = 1200/2,77 = 433 cnt/smp
2. Set the needed acceleration. A_SOLL = A x 271 [RPM/s^2]
   Ex. We need the motor to accelerate with 100000 RPM/s^2 so, A_SOLL = 100000/271 = 369 cnt/smp^2.
3. Now set the motor into velocity mode and thereby activate the motor.
   Ex. The motor needs to be activated by setting it into velocity mode, so we need to set the mode register to the value 1. Mode = 1 which is velocity mode, now the motor will use the acceleration and the velocity just configured.

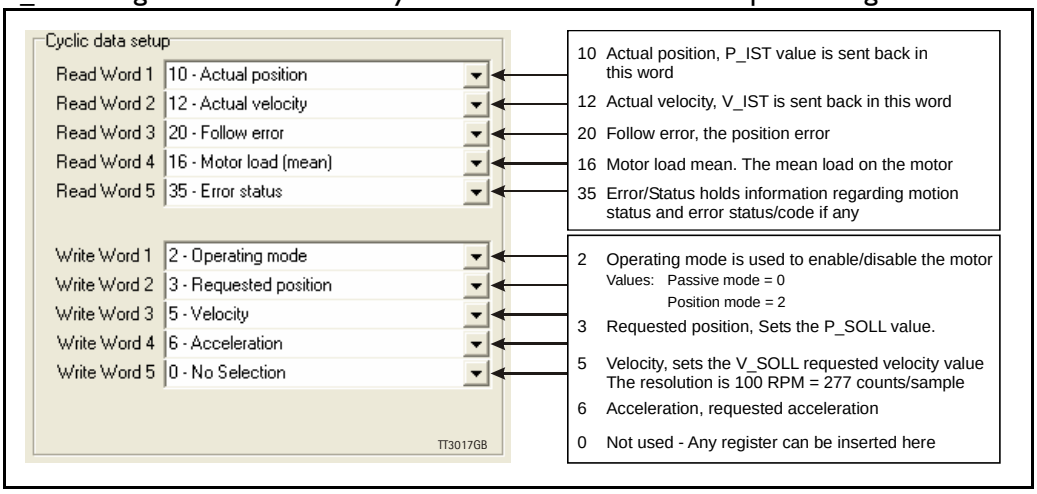
Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

3.6.2 Running Position control (JVL profile)

Running the motor in position control requires that the mode register is set for position control. The following registers is of particular interest when position mode is used.

1. "Actual position" -P_IST, register 10
2. "Actual velocity" -V_IST, register 12
3. "Follow error" - The actual position error, register 20
4. "Motor load mean" - average motor load, register 16
5. "Error/Status" -register 35
6. "Requested position" -P_SOLL, register 3
7. "Requested velocity" -V_SOLL, register 5
8. "Requested acceleration" -A_SOLL, register 6

In this mode the position is controlled by applying a requested position to the "P_SOLL" -register and the actual position is monitored in the "P_IST" register. The V_SOLL and A_SOLL registers sets the velocity and acceleration used when positioning occurs.



3.6.3 General considerations

The register 35 in the motor holds information on the actual error/status. So it is crucial that this register is configured in the cyclic data and thereby obtained and monitored in the Master. In case of an error situation the motor will stop and the cause will be present in the register 35 and hence in the I/O -data.
This register also holds information on the motion status such as:

- In position, bit 4
- Accelerating, bit 5
- Decelerating, bit 6

Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

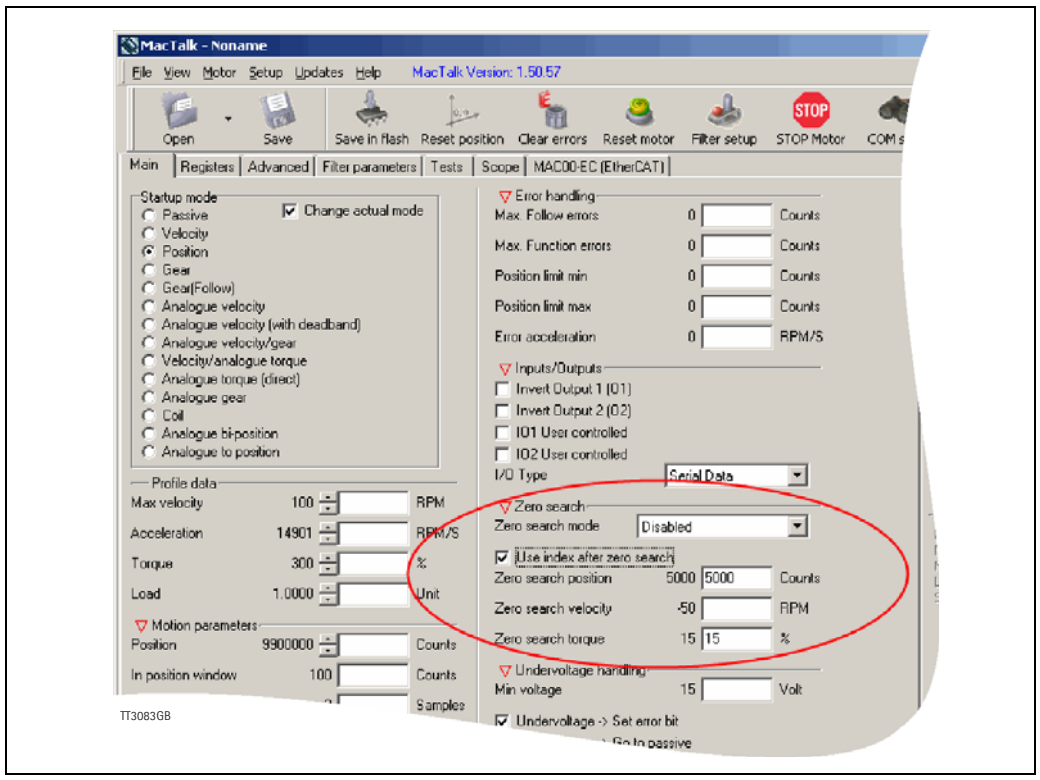
The JVL motor is basically put into a working mode and into a passive mode where the motor axle is de-energized, by setting register 2 into either 0 = "passive mode" or into one of the supported modes.

Example.
1= "Velocity mode" / 2= "Position mode" / etc.

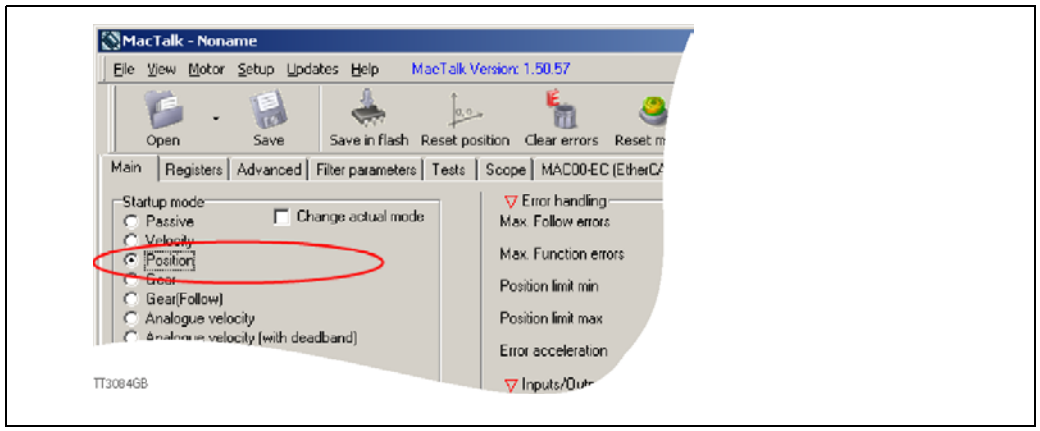
So in order to Stop or Start the motor this register can be supported in the I/O data or by sending an SDO message.

3.6.4 Homing using only cyclic I/O (JVL profile).

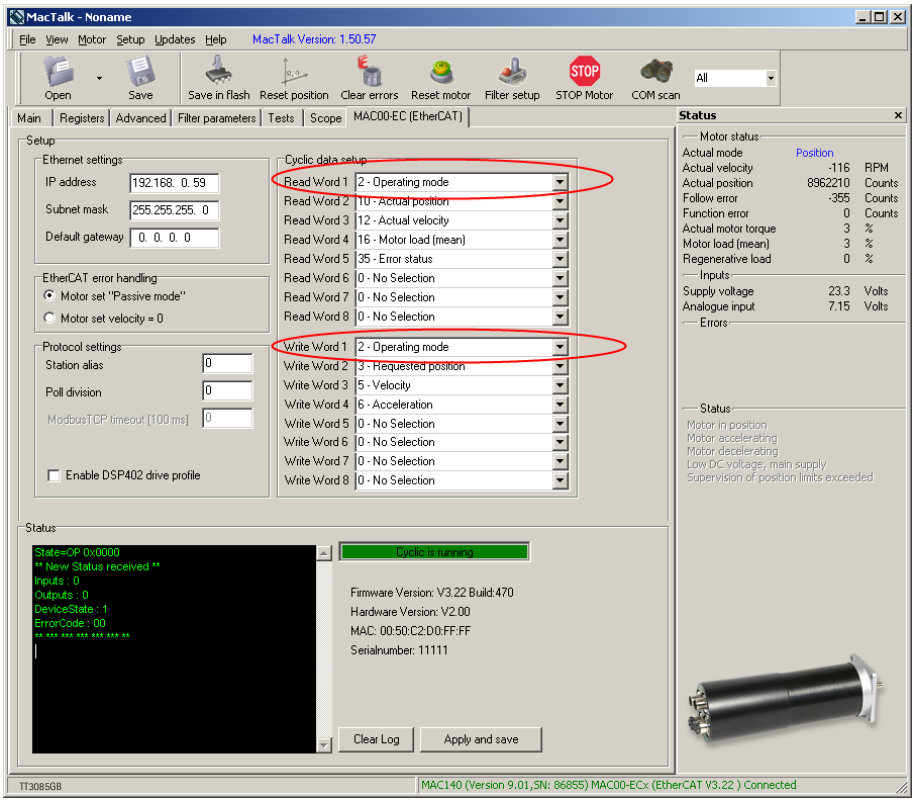
When doing a homing (Zero search), with only cyclic I/O, some preconditions have to be met:
Zero search position, zero search velocity and zero search torque (torque only for MAC motors) has to be set in MacTalk in the "Main" tab, and saved in flash in the motor once and for all.



Startup mode should be set to position, for the motor to stay in position after the homing sequence. And this setting should also be saved in flash.



Register 2 (Operating mode) has to be present in BOTH the cyclic read words and cyclic write words.



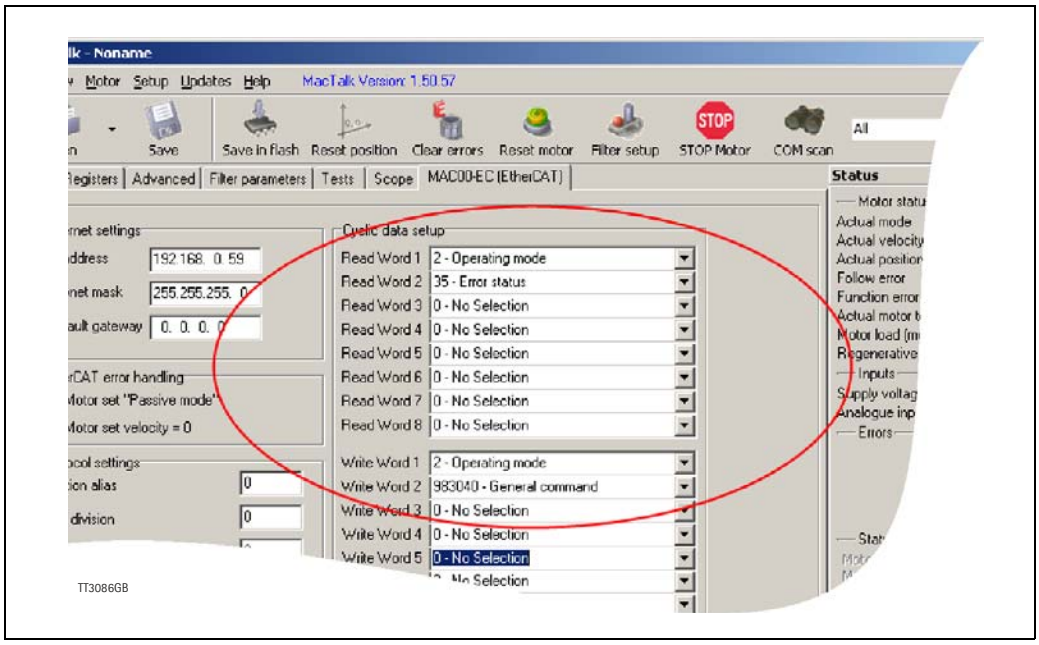
Procedure in the PLC:

- Treat the transmitted Register 2 as "Requested_Mode" and the received register 2 as "Actual_Mode".
- When homing is wanted, set the "Requested_Mode" to one of the values 12, 13 or 14 depending of the requested homing mode (12 = Torque based zero search mode (only MAC motors). 13 = Forward/only zero search mode. 14 = Forward+backward zero search mode (only MAC motors) .). For a comprehensive description of the homing modes, refer to the general MAC motor manual - LB0047-xxGB.
- Observe that the "Actual_Mode" is changing to the homing mode. Now the module is blocking cyclic writes TO the motor. Cyclic reads is still active.
- Wait for register 35 "Error status" bit 4 to be active =IN_POSITION. (Indicates that homing is finished).
- Then change "Requested_Mode" to whatever needed. The blocking of cyclic writes to the motor is then released by the module.

3.6.5 Relative positioning.

There are a number of ways to do relative positioning, but the one explained here is very simple, and can be used with a constant distance, or exchangeable distance, to move every time it is requested.

Preconditions:
Place the module command register (register 983040 in MacTalk) in the cyclic write list. The cyclic setup, could for example look like this:



Procedure in the PLC:
1. Set up register P7 in motor to requested relative offset.
2. Make sure one net cycle has passed, so P7 resides in the motor.
3. Issue command 0x800000F1 (0x80000071 if MIS34x) in module command register (register 983040 in MacTalk).
4. Make sure one net cycle has passed, so command is interpreted by the motor.
5. Set module command register to zero. This will prepare the Ethernet module for new commands.
6. If needed then monitor register 35 (Error status): When bit 4 is set (in position), then the move is finished.
7. When a new relative move is requested, go to step 3.

You may also have the P7 register in the cyclic write list, thereby enabling easy change of the relative distance to move.

4 MAC00-EI4 EthernetIP® module

4.1 Introduction to EthernetIP

Expansion module MAC00-EI4 front plate

4.1.1 Introduction to EtherNet/IP

The JVL MAC00-EI -module makes communication using EtherNet/IP possible with the JVL motor. The Ethernet technology gives the advantages of fast data access using standard off the shelf hardware which again has the advantage of large accessibility and low prices.

The JVL implementation is done in a way that minimizes the complexity of getting a system up and running but still utilizes the benefits of industrial ethernet.
The JVL EtherNet/IP implementation supports both explicit messaging and I/O messages with up to 5 free configurable input and output words.

With a basic knowledge of the JVL motor operation through the register structure and a basic knowledge of the EtherNet/IP technology, a motor can be setup and controlled in a very short time without first doing extensive studies in complex motion control standards e.t.c.

EtherNet/IP is basically divided in 2 groups of data, explicit and I/O messages in other words messages requiring fast data response time and data not so time critical typically used for configuration purposes. In the EtherNet/IP terminology these messages are also called Explicit messages (not time critical, none cyclic exchanged) and I/O messages (time critical, cyclic exchanged).

In the motion control world, time critical data would be actual position, actual status and actual speed and actual torque where data not time critical would be such as motor temperature and setup parameters.

EtherNet/IP is object based similar to DeviceNet and follows the standards issued by ODVA.
For more information on EtherNet/IP please visit www.ODVA.org for further details on EtherNet/IP and to get the EtherNet/IP standard specification issued by ODVA.

(continued next page).

4.1 Introduction to EthernetIP

The JVL implementation supports manufacture specific objects to gain access to each register in the motor.

This manual assumes that the servomotor user manual has been read and a base knowledge using the servomotor and the configuration software MacTalk is acquired.

The examples and screen shots in this manual are taken from MacTalk and a Rockwell RSLogix5000 application.

Please be aware that other PLC vendors than Rockwell exist.

4.1.2 Daisy chaining

Up to 64 units (nodes) can be daisy chained. By daisy chained means making a direct cable from the master in the system to motor 1 (L/A IN). From motor 1 (L/A OUT) to motor 2 (L/A IN) etc.
This method is saving hardware since no switch(es) and can often be the simplest way of doing the wiring. The disadvantage is that the data will be delayed slightly depending on how many motors that are daisy chained and the network load will be significant if a larger number of motors is connected this way.

Another and more common solution is to use a switch after the master and then distribute data to each node from this switch. This solution has a minimal delay of the data stream.

4.1 Introduction to EthernetIP

4.1.3 EthernetIP specification

The JVL implementation supports manufacturer specific objects to gain access to each register in the motor.

Supported standard EthernetIP classes

Type	Class
Identity Object, class	0x01
Message router object, class	0x02
Assembly object, class	0x04
TCP/IP interface object, class	0xF5
Ethernet link object, class	0xF6

On top of this the JVL manufacture specific class object 0x64 has been added.

Identity object class 0x01
Holds information about the JVL device on the network. Typical used by other devices to identify devices on the network.
(for further specification please refer to the EtherNet/IP approximately.)

Message router object class 0x02
Handles all messages to/from object's in the device.

Assembly object class 0x04
Object that binds all IO data to a connection point.

TCP/IP interface object class 0xF5
Holds all information on the Ethernet connection, such as the IP-address, Network mask and GateWay.

Ethernet link object class 0xF6
Holds information on link specific counters and instances associated with the communication interface.

To gain access to the motor registers Class object 0x64 is used.

See section “Objects accessible using Explicit messages“
for further details see *Examples, page 81*

4.2 Using none cyclic messages

4.2.1 Using none cyclic messages (Explicit messages)

None cyclic messages in the EtherNet/IP domain is called Explicit messages. This message type is typically used to perform configuration and other none-time critical operations.

Explicit messages can be send as a connected or unconnected message.

All registers in the motor can be accessed explicitly using object class 0x64. The register range in the motor is from 1-255 all 32bit size.

For a complete register list please see *Motor registers MAC050 - 141, page 171* or *Motor registers MAC400 - 3000, page 180*.

The object class 0x64 explained in details:

Service type and code supported:
Set_Attribute_Single 0x10
Get_Attribute_Single 0xE

Instances supported: 0x01-0xFF (motor registers 1-255)

4.2.2 Type definitions:

UINT 16bit
DINT 32bit
STR String of ASCII-chars

4.2.3 Identity object class 0x01

Holds data on different module specific data.
Instance = 1

Attr. ID	Access	Name	Data type	Description
1	R	Vendor ID	UINT	JVL vendor ID = 936 (0x3A8)
2	R	Device Type	UINT	Value=10
3	R	Product code	UINT	Value = 1
4	R	Revision	UINT	Major = 1.byte, minor = 2. byte
5	R	Status	UINT	Status
6	R	Serial number	DINT	Serial number
7	R	Product name	STR	“MAC00-EIx”

See the EtherNet/IP spec. for further details section Vol2 sect.5-3.

Supported Services
0x1 Get_Attribute_All
0x10 Set_Attribute_Single
0xE Get_Attribute_Single

4.2 Using none cyclic messages

4.2.4 Assembly object class 0x04

Holds pre-configured motor registers to be accessed.

Instances
0x64 Write Data to motor register.
0x65 Read motor register data.

Attr. ID	Access	Name	Data type	Description
3	R/W	Get/Set Assembly	20 bytes	Get/Set all assembly data
4	R	Bytes	UINT	Bytes transferred in assembly

Supported Services
0x10 Set_Attribute_Single
0xE Get_Attribute_Single

This object can be used to access the predefined registers, configured from MacTalk.
They are also accessed when using the implicit connection cyclically.
If other registers than the one defined in the assembly object needs to be accessed then the class 0x64 needs to be used.
This class accesses each register in the motor for a more dynamically way of controlling registers explicitly.
The vendor specific class 0x64 is specified in details later in this chapter.

4.2.5 TCP/IP object class 0xF5

Holds data on different module specific data.

Attr. ID	Access	Name	Data type	Description
1	0xE	Status	DINT	Status bit-field
2	0xE	Configuration capability	DINT	DINTbit field = 5 (BOOTP+DHCP)
3	0x10	Configuration control	DINT	Bit field = 0 (use NV-setup)
4	0xE	Physical link object	6 bytes	Size + path
5	0x10	TCP/IP interface zup	22bytes	IP + sub net + GTW info e.t.c.
6	0x10	Host name	DINT	Host name

See the EtherNet/IP spec. for further details section Vol2 sect.5-3.

Supported Services
0x1 Get_Attribute_All
0x10 Set_Attribute_Single
0xE Get_Attribute_Single

4.2 Using none cyclic messages

4.2.6 TCP/IP object class 0xF6

Holds information for a IEEE 802.3 communication interface

Attr. ID	Access	Name	Data type	Description
1	0xE	Interface speed	DINT	Speed in Mbit/s
2	0xE	Interface status	DINT	Bit field
3	0xE	MAC-address	6 bytes	MAC
4	--	Not Implemented	--	--
5	--	Not Implemented	--	--
6	0x10	Interface Control	DINT	Bit field

See EtherNet/IP spec. for further details Vol2 sect. 5-4

Supported Services

0x1 Get_Attribute_All
0x10 Set_Attribute_Single
0xE Get_Attribute_Single

4.2.7 Vendor specific JVL object class 0x64

Holds pre-configured motor registers to be accessed.

Instances

1 - 255 Motor registers

Attr. ID	Access	Name	Data type	Description
1	0xE / 0x10	Get/Set register	DINT	Get/Set the specified motor register

Supported Services

0x10 Set_Attribute_Single
0xE Get_Attribute_Single

Please notice: Please find a complete list of register descriptions in the appendix. *Motor registers MAC050 - 141, page 171* and *Motor registers MAC400 - 3000, page 180*

4.2 Using none cyclic messages

E.g. the motor shall be operated in velocity mode.
This requires that the mode register 2 = 0x1.
Velocity mode is 0x1, Position mode = 0x2 e.t.c.
All modes of operation is further described in the servo manual.
The explicit message is setup as follows.

Package:
Class: 0x64
Service: 0x10 (write data)
Instance: 0x2 (mode register in the motor)
Attribute: 0x1

Data: **0x00 00 00 01**

This will set the mode register in the motor into velocity -mode
Motor Register 2 = 1

To read a value from the motor use the service code 0xE.

After setting the motor into velocity mode it will start running. Now the actual velocity can be read while the motor is running.

Package:
Class: 0x64
Service: 0xE (Read data)
Instance: 0xC (Actual velocity)
Attribute: 0x1

Now the response data is received:

Data: **0x00 00 01 15**

This value 0x115 (hex) is the decimal value 277 which corresponds to 100 RPM. This is the default velocity value.

So basically the motor can be controlled and all needed data can be retrieved using explicit messages. This method is not suitable when data is needed fast and frequently for this purpose I/O messaging (Implicit messaging) is used.

Not only motor registers are accessible using explicit messages, also static data such as serial numbers, network status etc. are accessible. These informations are accessible according to the EtherNet/IP standard and follows the implemented classes: **0x1, 0x4, 0xF5, 0xF6.** These classes are explained in details in the EtherNet/IP standard (obtained from www.ODVA.org) and in

For further info please See “Examples” on page 81.

4.3 Using cyclic I/O-messages

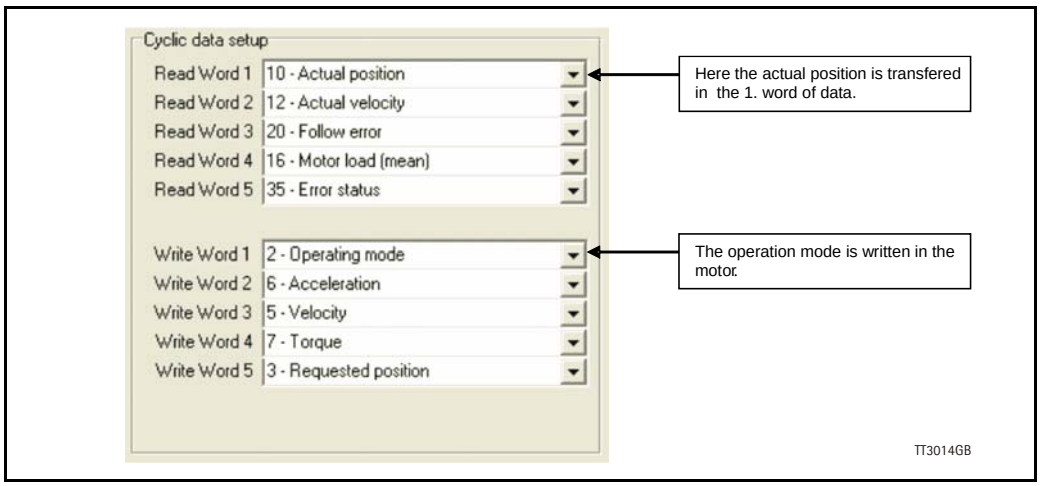
4.3.1 Cyclic messages.

I/O messaging also referred to as Implicit messages is used when data is needed fast and frequent. That is fast dynamic changing data such as position, velocity, torque etc.
It is mandatory to have the error/status register (register 35) as one of the slave to master registers. If not the motor will overrule the configuration and place register 35 anyway.
These data is sent cyclic using the assembly class object 0x04.
If module registers is placed in cyclic R/W, then the register number has to be calculated as follows:

Register number = 65536 x sub index.
Example: module command (sub-index 15) = 65536 x 15 = register **983040**

When module registers (register numbers above 65535) are chosen, they **have** to be placed **after** the motor registers in the list of cyclic registers.
The JVL assembly consists of 8I/O words that are freely configurable. This means that 8 input motor registers can be selected and another 8 motor registers for output purposes.
The terms Input and output is considered from the scanner so input is data flowing from the motor to the scanner and output is vice versa.
On the EthernetIP -tab in MacTalk these I/O's are configured.

NB! If an index is set to zero (No selection), then the following indexes is discarded. Thereby computing resources in the drive are released, which makes much faster cycle times possibly. Please see next paragraph.



All words are 4 bytes.
In the example shown above the 5 read words (data read from the motor) are:

Motor register 10 (Actual position)	The actual motor position
Motor register 12 (Actual velocity)	The actual velocity of the motor
Motor register 20 (Follow error)	The actual follow error in the motor movement
Motor register 16 (Motor load - mean)	The load the motor is experiencing over time
Motor register 35 (Error status)	Bit-field that holds both error information and status of movements etc.

The 5 write registers are configured to hold the following data:

Motor register 2 (Operating mode)	0=passive, 1=Velocity, 2=position etc
Motor register 6 (Acceleration)	The requested acceleration to be used.
Motor register 5 (Velocity)	The requested Velocity to be used.

Motor register 7 (Torque)	The max. allowed Torque to be used
Motor register 3 (Requested position)	The requested position if operating mode = 2 (position)

Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

Please notice: Even though all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

4.3.2 Minimum cycle time

The minimum cycle time is the minimum amount of time between each cyclic request on the Ethernet. If operating with values lower than those listed, data loss will occur.

No. of motor registers transmitted in each direction	Motor series MAC050 - MAC141	Motor series MAC400 to MAC3000
1/1	4ms *	1ms **
2/2	8ms *	1ms **
3/3	12ms *	1ms **
4/4	16ms *	1ms **
5/5	20ms *	1ms **
6/6	24ms *	1ms **
7/7	28ms *	1ms **
8/8	32ms *	1ms **

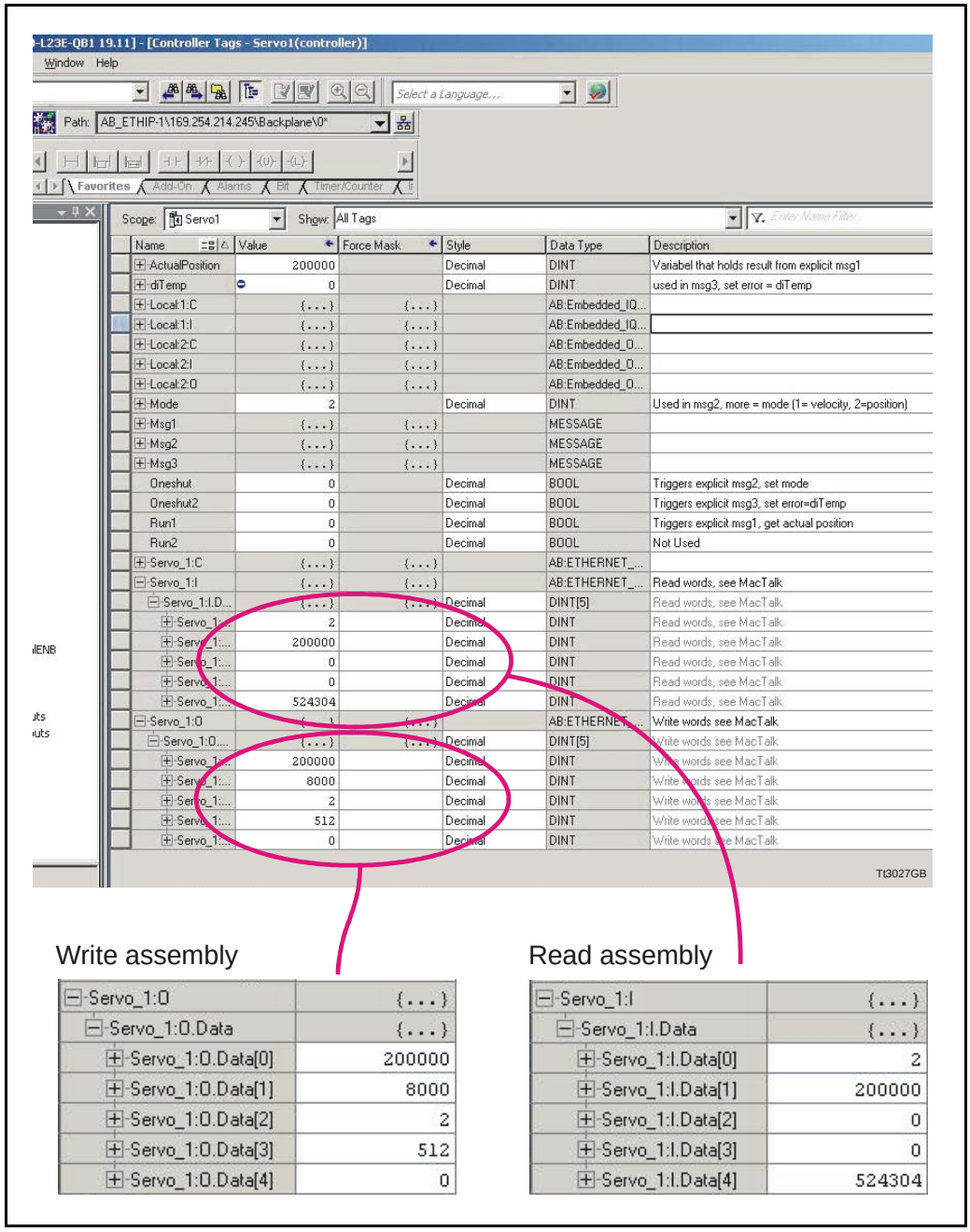
* The minimum cycle times, is only valid if not sending any acyclic requests while in any operating mode. MODULE registers can be appended as the last registers in the list, at no extra timing cost. Motor register 35 shall be in the cyclic read list, as it is also used internally.

** Restrained by the EthernetIP protocol it self.

4.3 Using cyclic I/O-messages

4.3.3 Cyclic data in the PLC

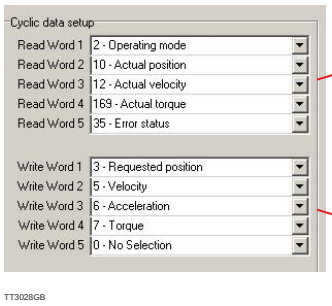
The complete list of Controller tags defined.



4.3 Using cyclic I/O-messages

MacTalk IO assembly setup, seen in the controller tag list and read from the PLC when the connection has been established.

MacTalk setup:



Servo_1:I	{...}
Servo_1:I.Data	{...}
Servo_1:I.Data[0]	2
Servo_1:I.Data[1]	200000
Servo_1:I.Data[2]	0
Servo_1:I.Data[3]	0
Servo_1:I.Data[4]	524304

Explanation

2 - Operating Mode = 2 (position mode)
10 - Actual Position = 200000
12 - Actual Velocity = 0 Cnt/s
169 - Actual Torque = 0 (1024 = 300%)
35 - Error Status = 524304 (no errors)

Servo_1:O	{...}
Servo_1:O.Data	{...}
Servo_1:O.Data[0]	200000
Servo_1:O.Data[1]	8000
Servo_1:O.Data[2]	2
Servo_1:O.Data[3]	512
Servo_1:O.Data[4]	0

Explanation

3 - Requested position = 200000
5 - Velocity = 8000 (8000 = 2820 RPM)
6 - Acceleration = 2 Cnt/s² (2 = 543 RPM/s²)
7 - Torque = 512 (512 = 150%)
0 - No Selection (value is not updated)

4.4 Commissioning

4.4.1 Necessary equipment

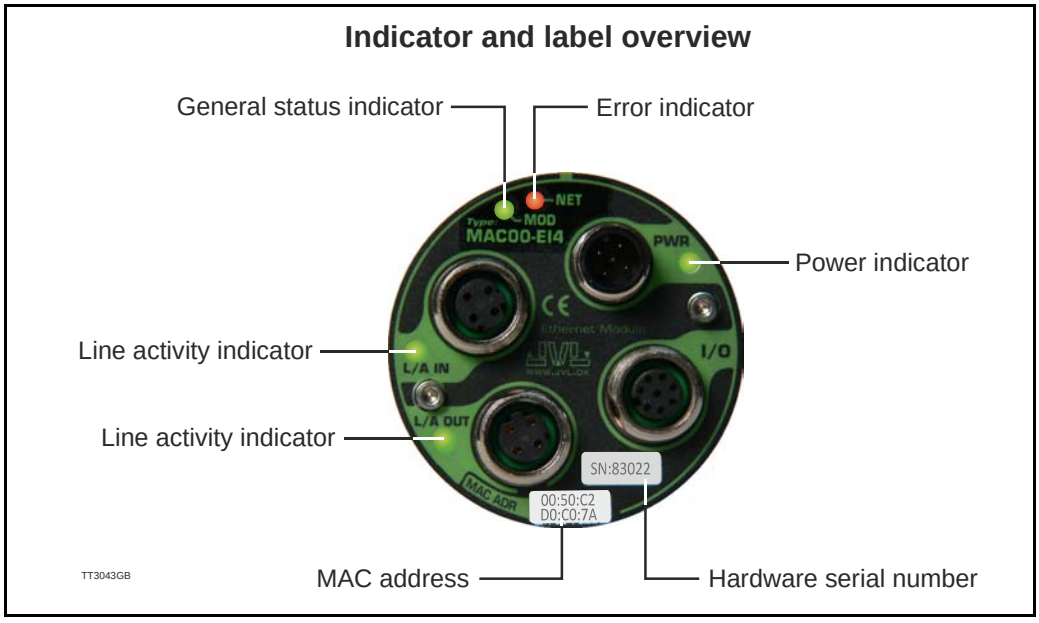
To get started you will need the following equipment.

- MAC motor with EthernetIP module (MAC00-EI.)
- A PLC or master controller with EthernetIP interface and relevant software
- A PC installed with MacTalk software in order to setup the MAC motor.
- Relevant signal and low voltage cables such as Ethernet cable, 24V power cable, RS232. Please also see the section *Cable accessories, page 21*.
- A 24VDC supply able to deliver min.1000mA@24V.
- Concerning AC high voltage supply for the MAC motor please refer to the general MAC motor user manual (LB0047-xx)

The general MAC motor user manual can be downloaded from http://www.jvl.dk

4.4.2 Indicator LED's - description.

The LED's are used for indicating states and faults of module. There is one power LED, two link/activity LED's (one for each Ethernet connector), and 2 status LED's.



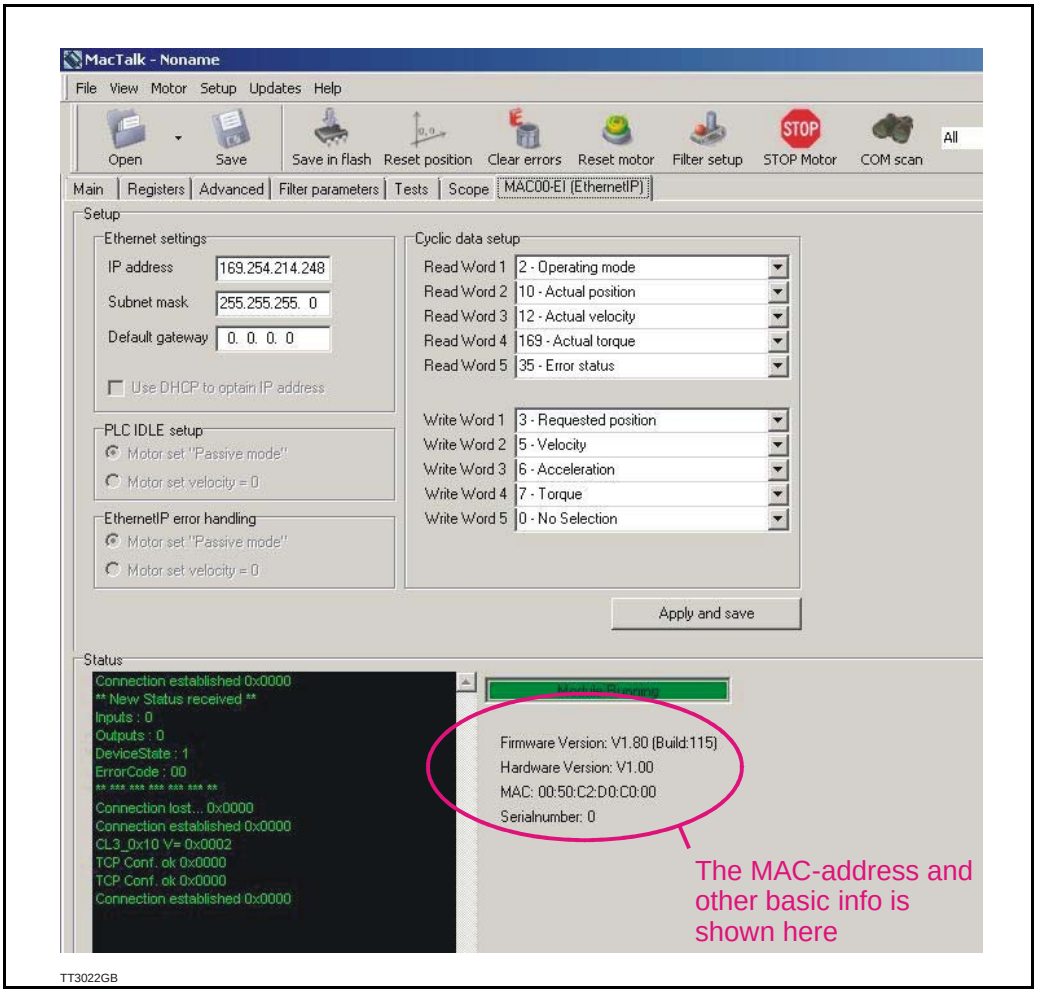
LED indicator descriptions

LED Text	Colour	Constant off	Constant on (Green)	Blinking (Green)	Constant on (Red)	Blinking (Red)	Blinking (Red/ Green)	Flickering
L/A IN	Green	No valid Ethernet connection.	Ethernet is connected.	-	-	-	-	Activity on line
L/A OUT	Green	No valid Ethernet connection.	Ethernet is connected.	-	-	-	-	Activity on line
MOD	Red/ Green	No power applied	Module status OK	Module not configured	Major module fault	Minor module fault	Self test in progress	-
NET	Red/ Green	No IP address	CIP conection established	No CIP connection	Duplicate IP address	Connection time-out	Self test in progress	-
PWR	Green	Power is not applied.	Power is applied.	-	-	-	-	Power is applied to module but no communication with motor

Notes:
Blinking: Flashing with equal on and off periods of 200ms (2.5Hz). **Flickering**: Rapid flashing with a period of approximately. 50ms (10 Hz).

4.4.3 MacTalk Ethernet configuration

The module is by default setup with the following Ethernet configuration:



After adjusting all settings press “Apply and save“
for the settings to take effect and for permanently saving the setup.

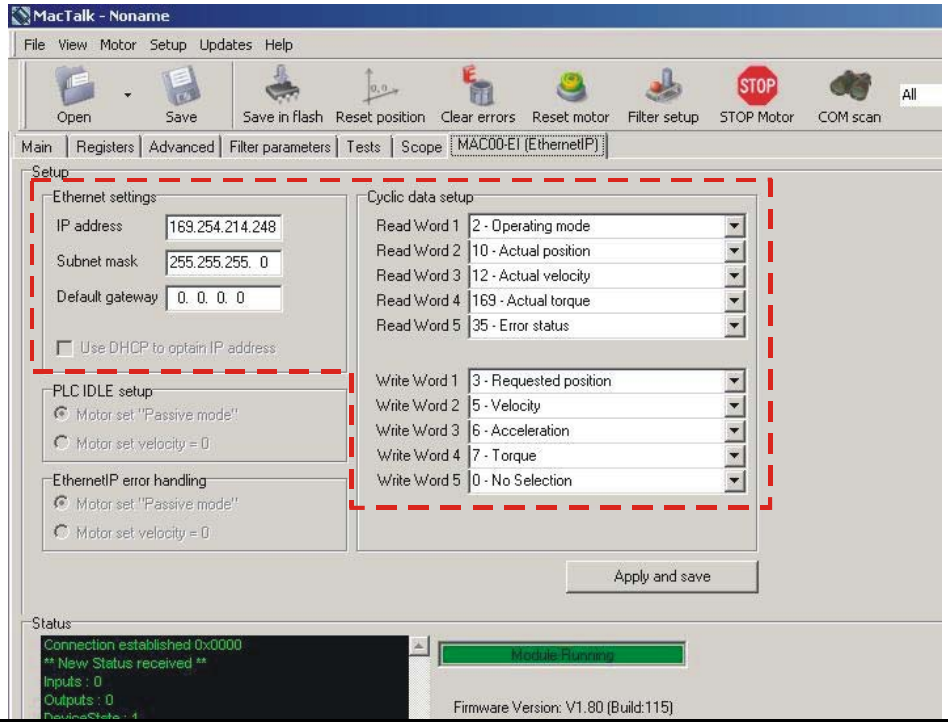
Information such as EtherNet/IP firmware version, MAC-address and module status is displayed in the “Status” -field. Notice that the MAC-address is unique for each module and can not be changed.
A label at the front plate of the module also indicate the MAC-address.

Basic use of MacTalk is described in the MAC-motor manual (lit. no. LB0047-xxGB)

Setting up IP addresses and general usage of the Rockwell CompactLogix PLC with the software package Logix5000 is beyond the scope of this example.

The following guideline is based on the JVL MAC400 motor with the factory setup.

1. Apply 24V, open MacTalk and setup the ethernet settings as required and the IO assembly (cyclic data setup) according to the following:



2. Press the “Apply and save” -button for permanent storage of the EthernetIP -settings.
3. Switch off the 24V supply while connecting the Ethernet cable to the switch/PLC.
4. Re-apply 24V set the PLC into “RUN” -mode.Now we should be able to control the motor.
5. Start by setting the profile data such as, Velocity, acceleration and Torque. According to the following:

Servo_1:O	{...}
Servo_1:O.Data	{...}
Servo_1:O.Data[0]	200000
Servo_1:O.Data[1]	8000
Servo_1:O.Data[2]	2
Servo_1:O.Data[3]	512
Servo_1:O.Data[4]	0

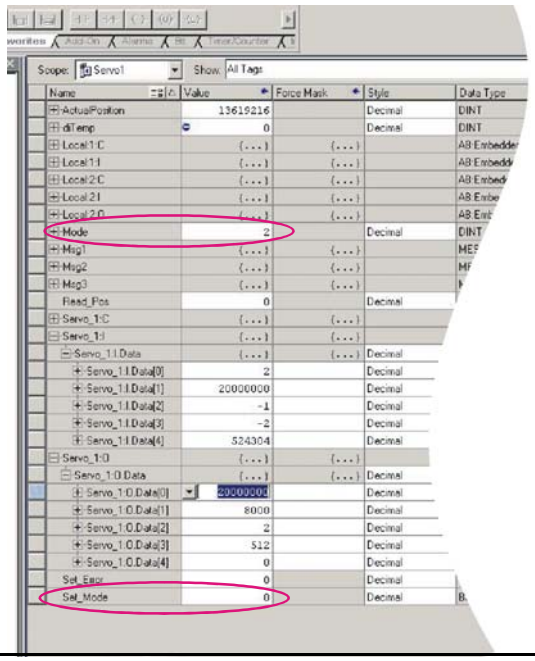
TT3031GB

Explanation

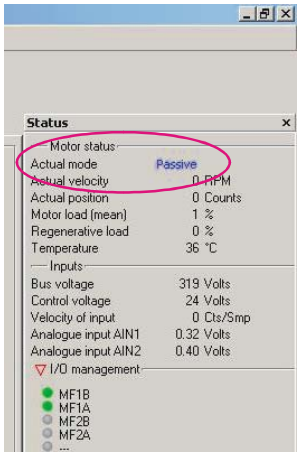
3 - Requested position = 200000
5 - Velocity = 8000 (8000 = 2820 RPM)
6 - Acceleration = 2 Cnt/s² (2 = 543 RPM/s²)
7 - Torque = 512 (512 = 150%)
0 - No Selection (value is not updated)

6. Now we will set the motor into an active mode (position mode), find the Controller tag "Mode" enter 2, find the tag "Set_Mode" enter 1. Now the motor is active and will start moving to the entered position in the "Servo_1:O_Data[0]" which is assigned to the requested position register in the motor. When the motor reaches the position it will stop and hold this position.
From MacTalk the actual mode (see the status-panel) is changed from "Passive" to Position and the motion progress can be followed. Remember to change the "Set_Mode" tag back to 0 to stop the sending of Msg2 -messages.

Logix 5000 tag list



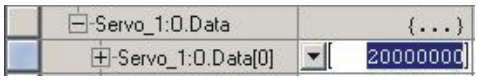
MacTalk status bar



Changing the "Servo_1:O_Data[0]"-tag will result in an immediate repositioning of the axle in the motor. This value is defined in the IO assembly and is interchanged cyclic.

To stop the motor set "Mode" = 0 and set "Set_Mode" = 1 to apply the mode setting. Reset "Set_Mode" to 0 again to stop sending Msg2. -messages.

7. To activate the explicit message Msg1 set the commanded position to a far greater value. For example 200000000 as illustrated below.

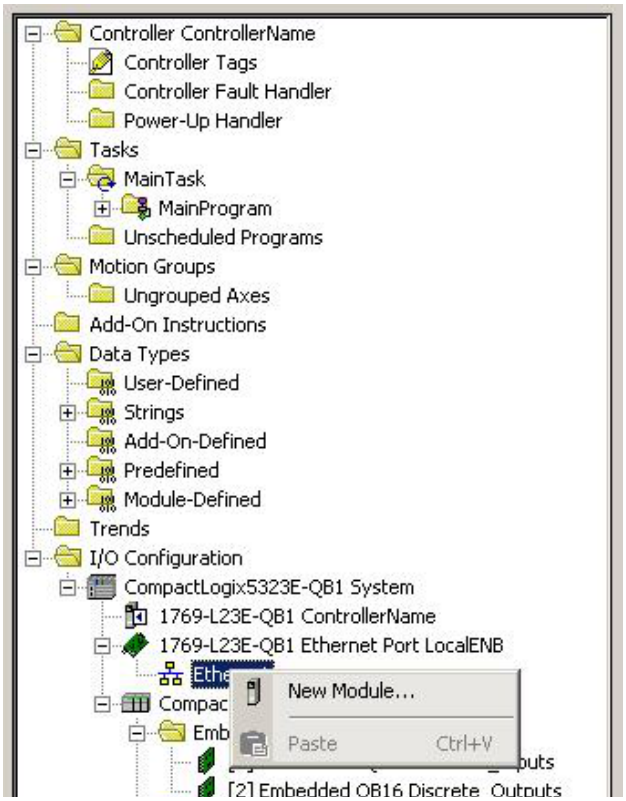


8. Find the "Read_Pos" -tag and set this to 1. Now the current position of the motor is seen in the "Actual Position" -tag.

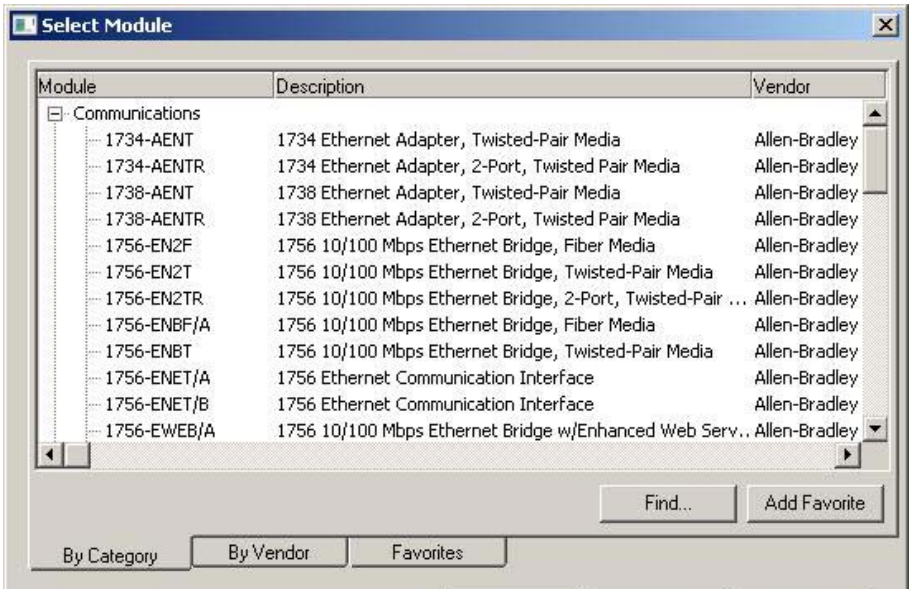
4.4.4 How to setup a Rockwell RSLogix5000 Project.

After creating a new project in the RSLogix5000 application the JVL motor must be added to the Ethernet bus-system in the project.

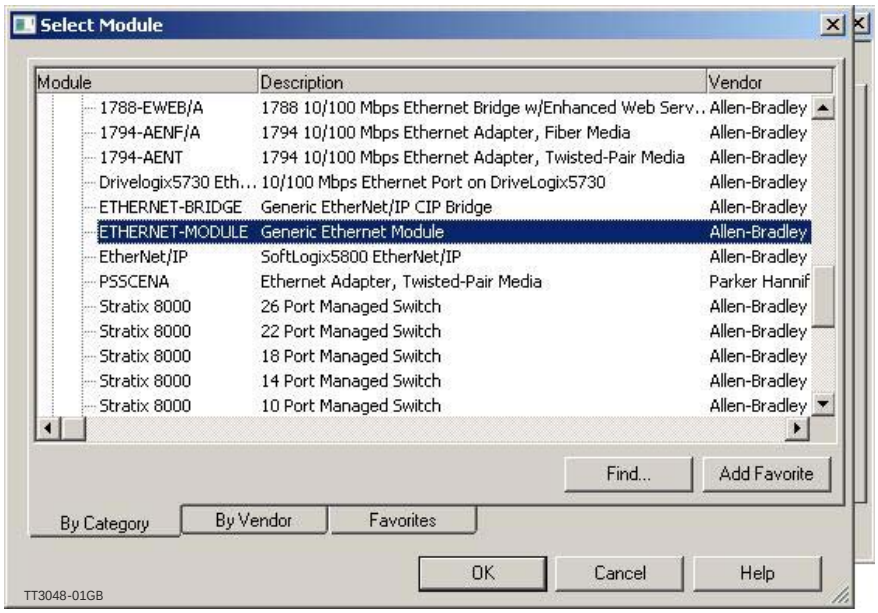
This is done by right clicking the "Ethernet-Module" icon in the project manager as illustrated below:



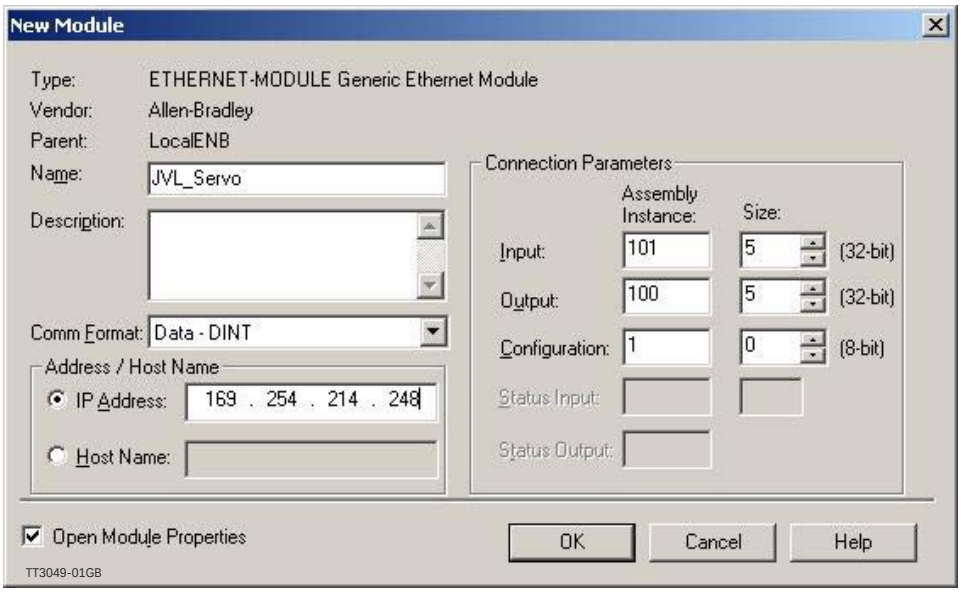
Select "New Module" and the following screen appears:
Expand the "Communications" - list.



Find and select the “Generic Ethernet module”.



Now the module parameter needs to be entered.
Fill in the information as illustrated below:



The IP-address illustrated is the factory default and may be changed according to the local settings.

After pressing “Ok” the JVL motor is added to the project and can now be reached by the PLC.

A demonstration video showing how to set-up the system can be seen using this link: http://www.jvl.dk

4.5 Implementation guidelines

4.5.1 Introduction

The following chapters describe the typical usage of the JVL Motor and which registers to use in the different applications.
The chapter should be considered as a general guideline to get started with the EthernetIP integration of the JVL Motor.

IMPORTANT!: Please notice that the motor will be active and may start moving when the mode register (reg. 2) is set to anything than 0 (passive mode). The MAC400, 800, 1500 and 3000 will require AC supply in order to be active.

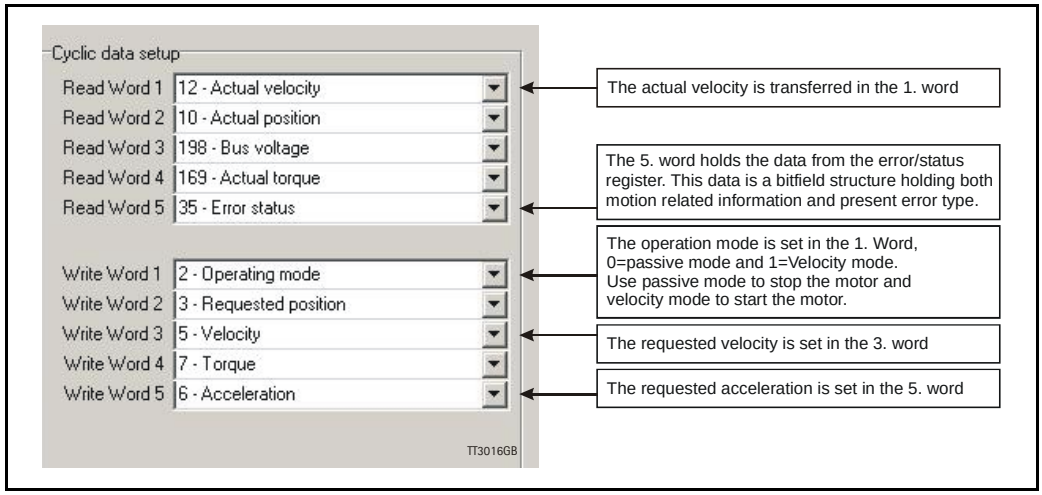
4.5 Implementation guidelines

4.5.2 Running Velocity control

To use the JVL motor in velocity mode the following registers are basically of interest.

1. "Mode" - Mode register 2
2. "V_SOLL" - Velocity register 5
3. "A_SOLL" - Acceleration register 6
4. "Error/Status" - Error and status register 35

So, to control these registers the assembly object needs to configured.
From MacTalk the setup is configured as this.



With the settings illustrated above we initiate the velocity mode by writing 0x1 to the first word-value, this is velocity mode.

From the scanner the registers is accessed using the assembly object and accessing the registers R/W on words 1-5.

1. Set the needed velocity. V_SOLL = V x 2.77 [rpm]
   Ex. We need the motor to run with a constant speed of 1200 RPM. So, V_SOLL = 1200/2,77 = 433 counts/sample
2. Set the needed acceleration. A_SOLL = A x 271 [RPM/s^2]
   Ex. We need the motor to accelerate with 100000 RPM/s^2 so, A_SOLL = 100000/ 271 = 369 counts/sample^2.
3. Now set the motor into velocity mode and thereby activate the motor.
   Ex. The motor needs to be activated by setting it into velocity mode, so we need to set the mode register to the value 1. Mode = 1 which is velocity mode, now the motor will use the acceleration and the velocity just configured.

Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

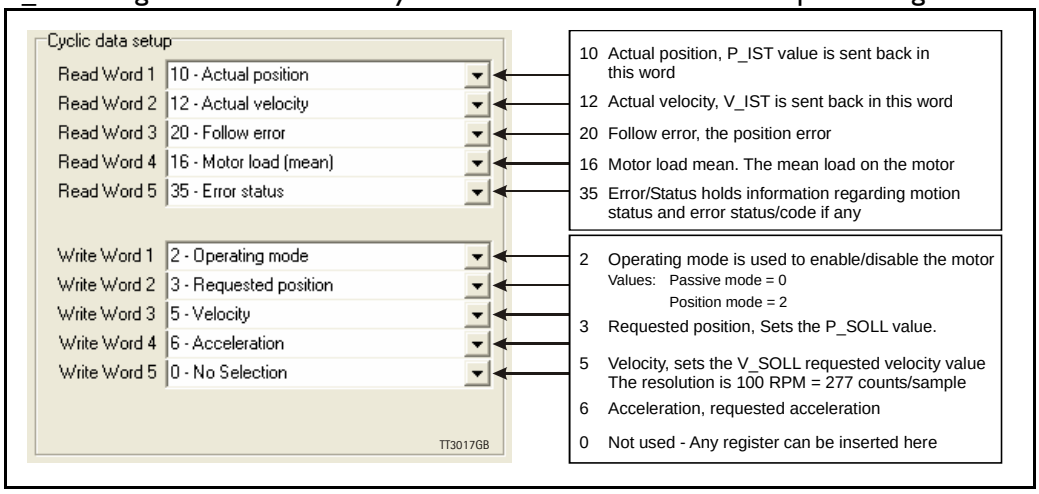
4.5 Implementation guidelines

4.5.3 Running Position control

Running the motor in position control requires that the mode register is set for position control. The following registers is of particular interest when position mode is used.

1. "Actual position" -P_IST, register 10
2. "Actual velocity" -V_IST, register 12
3. "Follow error" - The actual position error, register 20
4. "Motor load mean" - average motor load, register 16
5. "Error/Status" -register 35
6. "Requested position" -P_SOLL, register 3
7. "Requested velocity" -V_SOLL, register 5
8. "Requested acceleration" -A_SOLL, register 6

In this mode the position is controlled by applying a requested position to the "P_SOLL" -register and the actual position is monitored in the "P_IST" register. The V_SOLL and A_SOLL registers sets the velocity and acceleration used when the positioning occurs.



4.5.4 Error/status handling.

The register 35 in the motor holds information on the actual error/status. So it is crucial that this register is configured in the assembly object and thereby obtained and monitored in the scanner. In case of an error situation the motor will stop and the cause will be present in the register 35 and hence in the I/O -data.
This register also holds information on the motion status such as:

- In position, bit 4
- Accelerating, bit 5
- Decelerating, bit 6

Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

The JVL motor is basically put into a working mode and into a passive mode where the motor axle is de-energized, by setting register 2 into either 0 = "passive mode" or into one of the supported modes.

Example.
1= "Velocity mode" / 2= "Position mode" / etc.

So in order to Stop or Start the motor this register can be supported in the I/O data or by sending an explicit message.

4.6 Configuration using different methods

Basically a JVL motor works by loading a configuration into RAM memory from the none volatile flash memory when 24V power is applied and the motor is initialized.

The motor only holds one configuration and this configuration can be stored into the NV flash memory.

Several approaches can be used to configure the motor with data and finally saving them permanently in the NV flash.
A very general approach could be by using the PC-based software tool MacTalk, which offers both basic motor setup and control and the possibility to save all parameters in a separate file for backup purposes.
This software package utilizes the serial connection to communicate with the motor from any standard Windows PC.

Configuration over EtherNet/IP is possible by using explicit messages to address each register to be setup and then command the motor to save the configuration in flash afterwards for permanent storage.

Using this method the motor only needs to be setup once and is easy achievable from the scanner itself either as an initialization routine each time the PLC initializes, and thereby avoiding the permanent storage in the motor or simply using a configuration routine that sends the required explicit messages to address the needed registers followed by the message to save the settings permanently.

IP-address and other network settings still needs to be setup using MacTalk.

E.g. Setting up a motor sending messages explicitly

We want to change the default motor settings and save them permanently into flash.
The following registers needs to be saved:

The registers needed to be addressed are:

Velocity (V_SOLL) = Register 5
Acceleration (A_SOLL) = Register 6
Torque (T_SOLL) = Register 7

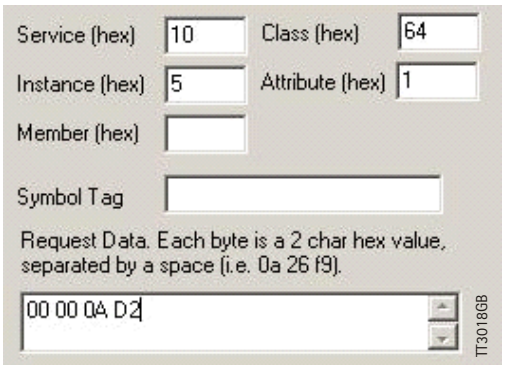
To address individual registers explicitly we use the class 0x64 for the purpose.

4.6 Configuration using different methods

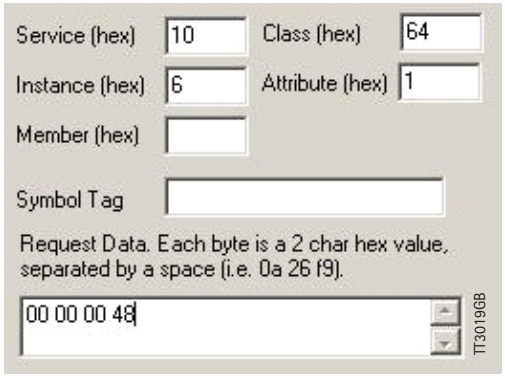
First we change the Velocity setting, we want the motor to spin with 1000 RPM.

The message for addressing V_SOLL is formed:

We need to scale 1000 RPM to the correct value in the motor the factor is 1 RPM = 2.77 counts/sample so we need to send the value 2770 = 0x00000AD2.
The instance refers to the register number, so we need to set instance to 5 (V_SOLL)
Please notice that the value is represented as 32bit.

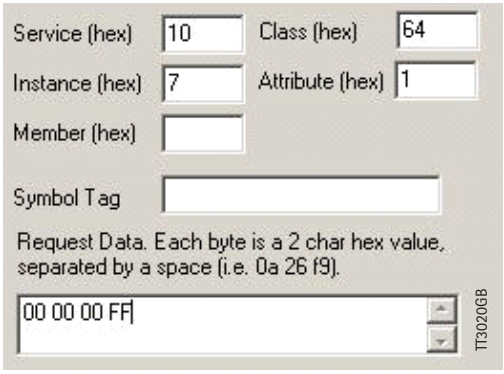


Next we set the acceleration to be used.
We need the acceleration to be 20000 RPM /s2
This value also needs to be scaled, the factor is 1 RPM/s^2 = 0.0036 counts/sample^2 so, in order to reach 20000 we need to send the value 72 = 0x00000048.
Acceleration is instance 6 (A_SOLL).

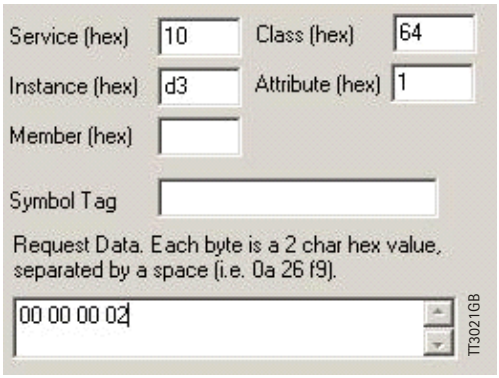


4.6 Configuration using different methods

Then configure the maximum motor torque to be used.
The motor can reach a peak torque of 300% the rated value. This value corresponds to 1023 in the register. We need 25% so we write 255 = 0x000000FF to instance 7 (T_SOLL).



And finally we send the command that saves the settings permanently in flash. This is basically a matter of writing the "save in flash" command to the command register 211 in the motor. The command is 2 and the instance is 211 = 0xD3. Value = 0x00000002. Now the motor saves the setting and resets.
It is required to toggle the 24V power in order to do a internal synchronization.



4.7 Using and Selecting an Ethernet switch

Depending on the network size and requested package interval (RPI) a suitable switch must be used. Also if multiple separated networks needs to be connected a switch is used.

Depending on the actual size of the network different requirements needs to be meet. Generally using EtherNet/IP with a fair package interval a 1 Gbps switch is typical adequate along with the following features:

- Auto negotiation, full duplex 100 MBit
- Port mirroring for network analysing and troubleshooting purposes. This feature makes it possible to route traffic out on a separate port connected to a network analyser for debugging purposes and general performance monitoring.

The JVL EthernerNetIP module has a small build in 2 port switch useful if a small amount of motors is connected in a daisy chaining topology.

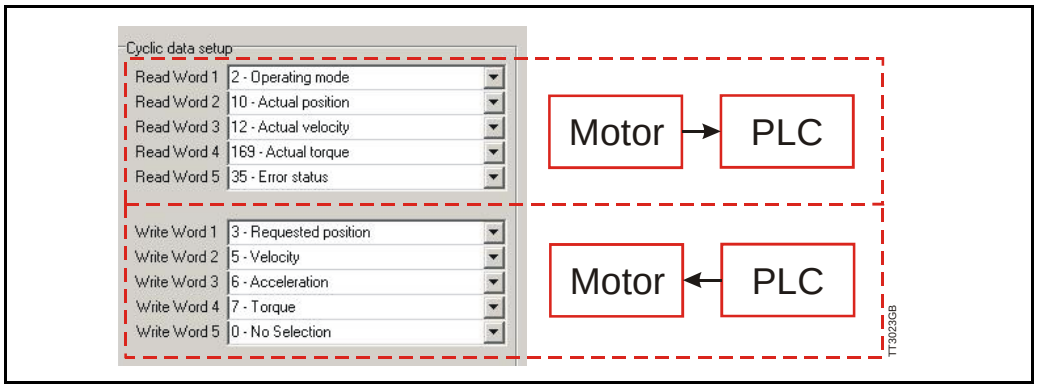
The disadvantage of this approach is that the package RPI timing is reduced as each motor needs to handle the incoming traffic for the other motors connected on the line.

4.8 Examples

4.8.1 Rockwell RSLogix example 1.

This is a simple example demonstrating the usage of both explicit messages and IO-assemblies to control a JVL MAC400 servo motor.
This example holds a few tags to control the inputs and outputs and a 3 rung ladder program to demonstrate the explicit message usage.
With this example it is possible to control the positioning of the motor using the “Position -mode” and set profile data such as velocity, acceleration and torque parameters using the IO-assembly.

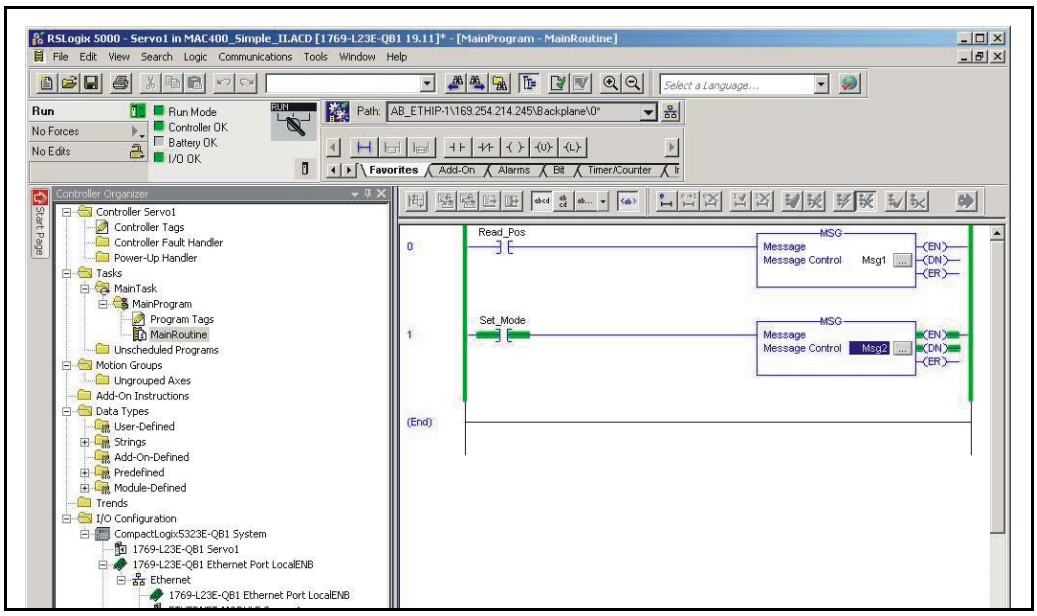
The example is developed for use on a CompactLogix L23E PLC using the Rockwell Logix500 software package and MacTalk from JVL.
The JVL MacTalk application is used to setup the IO assembly to fit the example.
Although this example expects default setup in the JVL motor, the IO assembly needs to be setup according to the following MacTalk setup (located at the EthernetIP tab).



The fixed sized assembly instances is divided into 5 read words and 5 write words.

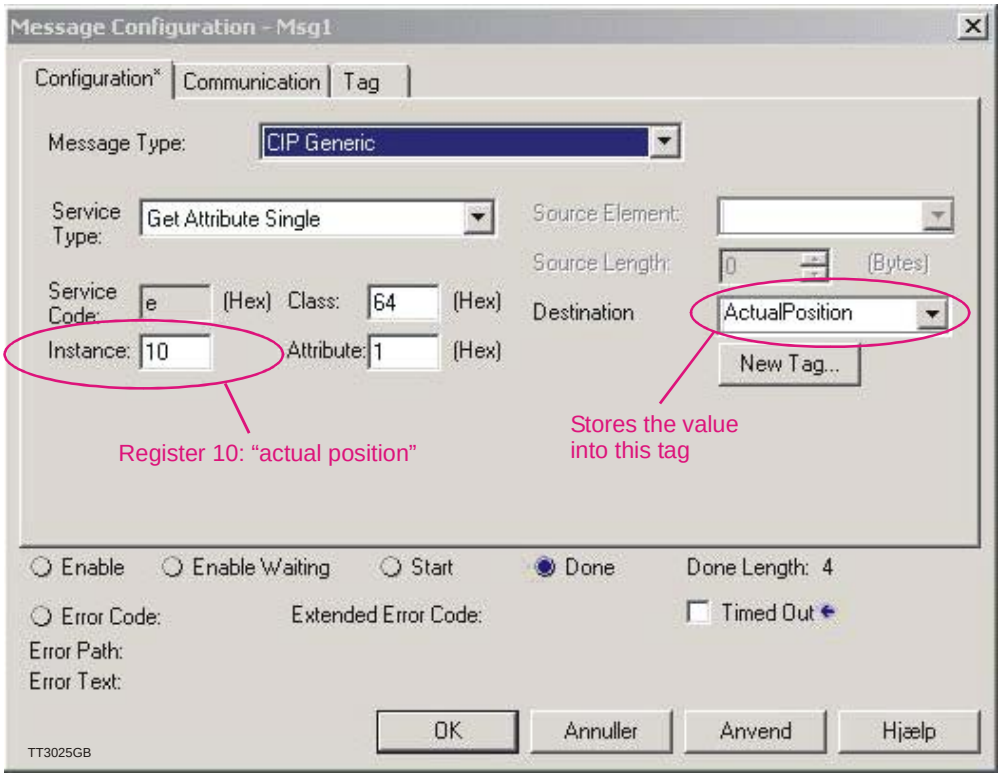
4.8.2 The RSLogix ladder program.

3 different messages for both setting data and retrieving data from the motor. All 3 messages are triggered by separate variables from the controller tag-list.



4.8.3 Message descriptions.

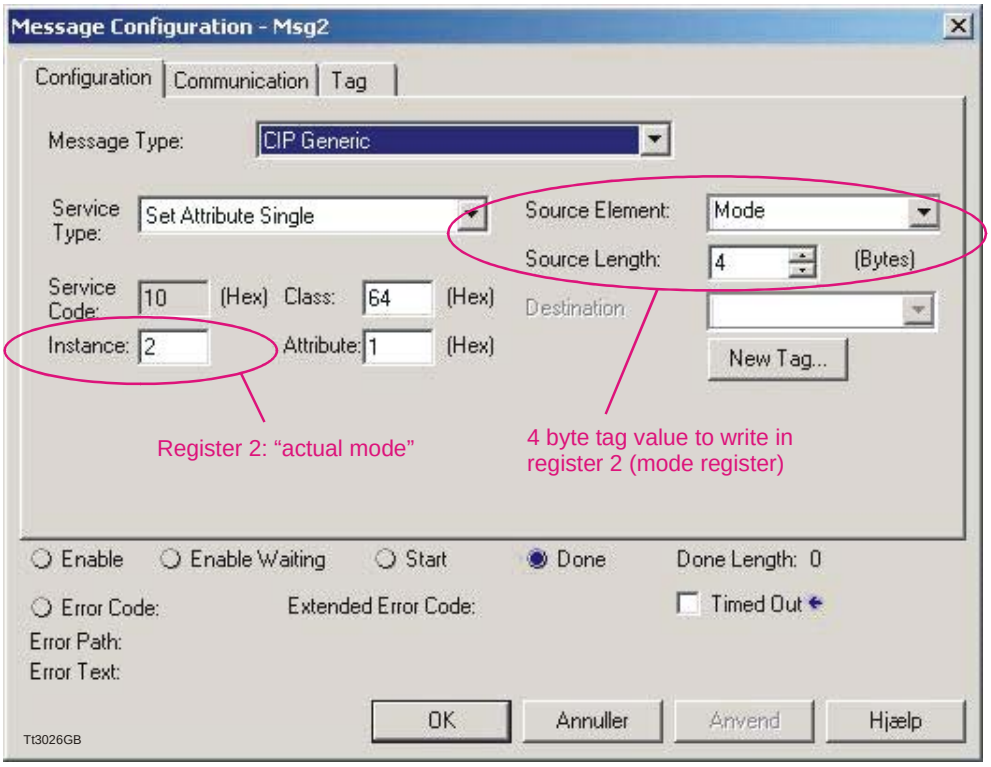
Msg1 reads information from the motor and is setup in the following way:
Reads (GET_ATTRIBUTE_SINGLE) the actual position register in the motor (instance 10) and stores the 4 byte value in the “ACTUAL POSITION” tag.



Please notice the mixed decimal and hexadecimal notation used.

4.8 Examples

Message 2 and 3 (Msg2, Msg3) are writing values to specific registers in the motor. They are configured in the following way:
Writes (SET_ATTRIBUTE_SINGLE) the value from the "MODE"-tag into the motor register 2 (Operation mode).



Please notice the mixed decimal and hexadecimal notation used.

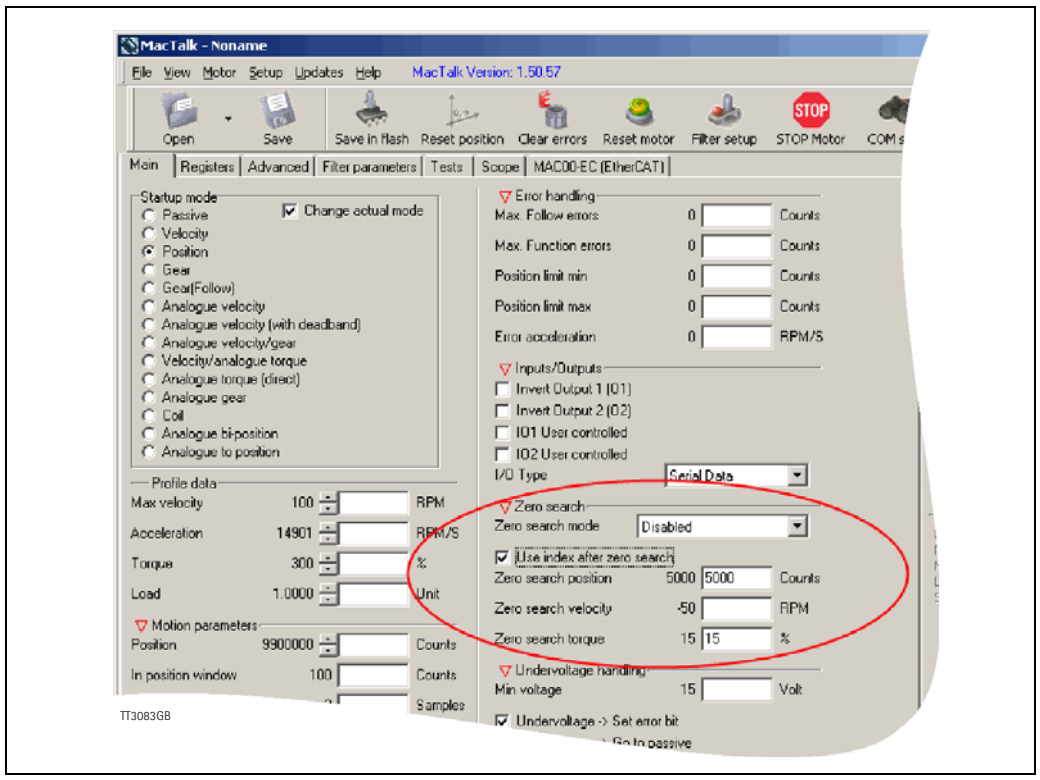
Explicit messages are always 4 bytes long and uses Class 0x64 to access the internal motor registers.
The instance refers to the actual motor register.

Instance = 2 points to the motor active mode -register.

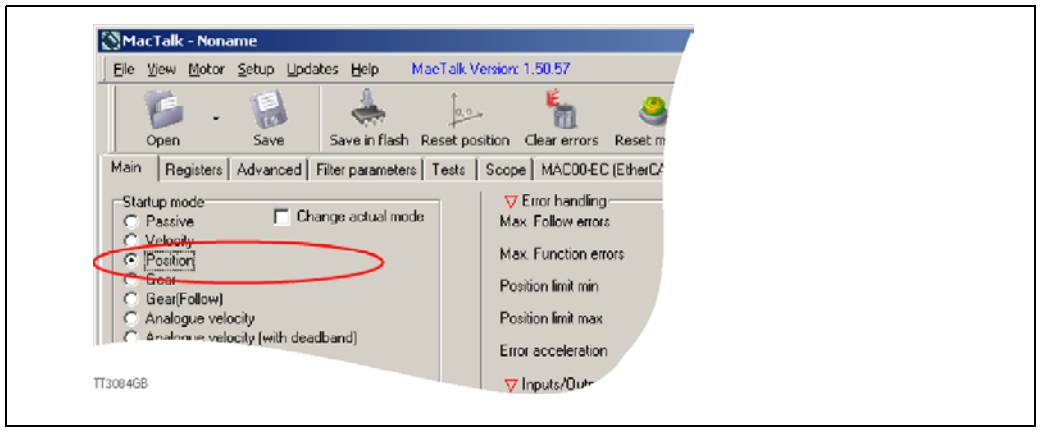
Explicit messages are typical used for configuration purpose or for rare data update situation that does not require a cyclic update timing.

4.8.4 Homing using only cyclic I/O (JVL profile).

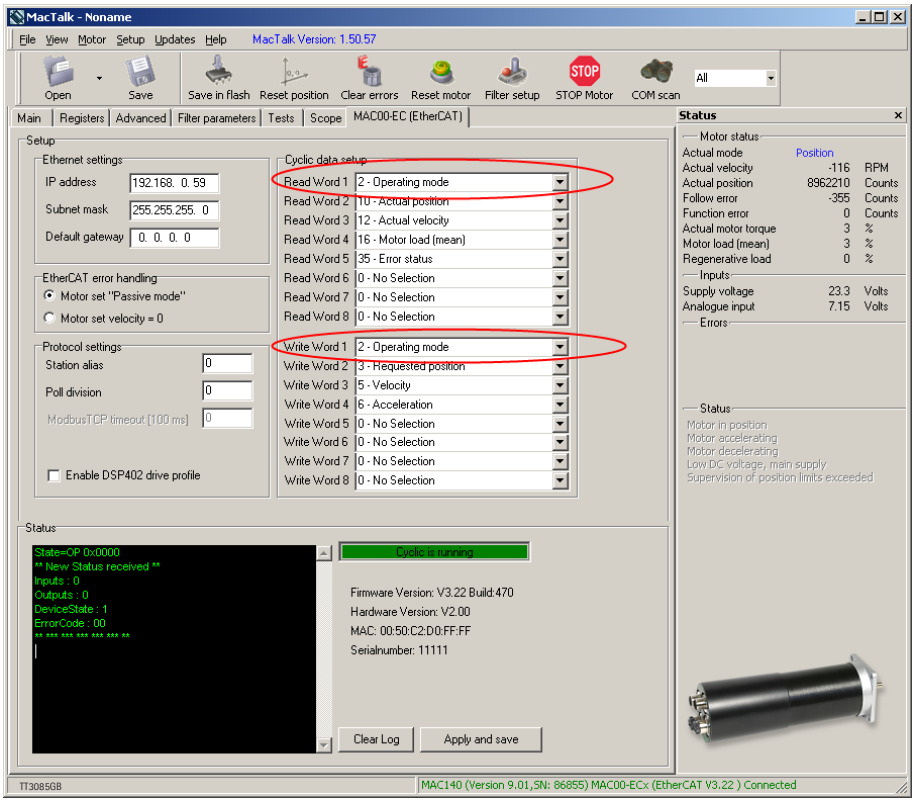
When doing a homing (Zero search), with only cyclic I/O, some preconditions have to be met:
Zero search position, zero search velocity and zero search torque (torque only for MAC motors) has to be set in MacTalk in the "Main" tab, and saved in flash in the motor once and for all.



Startup mode should be set to position, for the motor to stay in position after the homing sequence. And this setting should also be saved in flash.



Register 2 (Operating mode) has to be present in BOTH the cyclic read words and cyclic write words.



Procedure in the PLC:

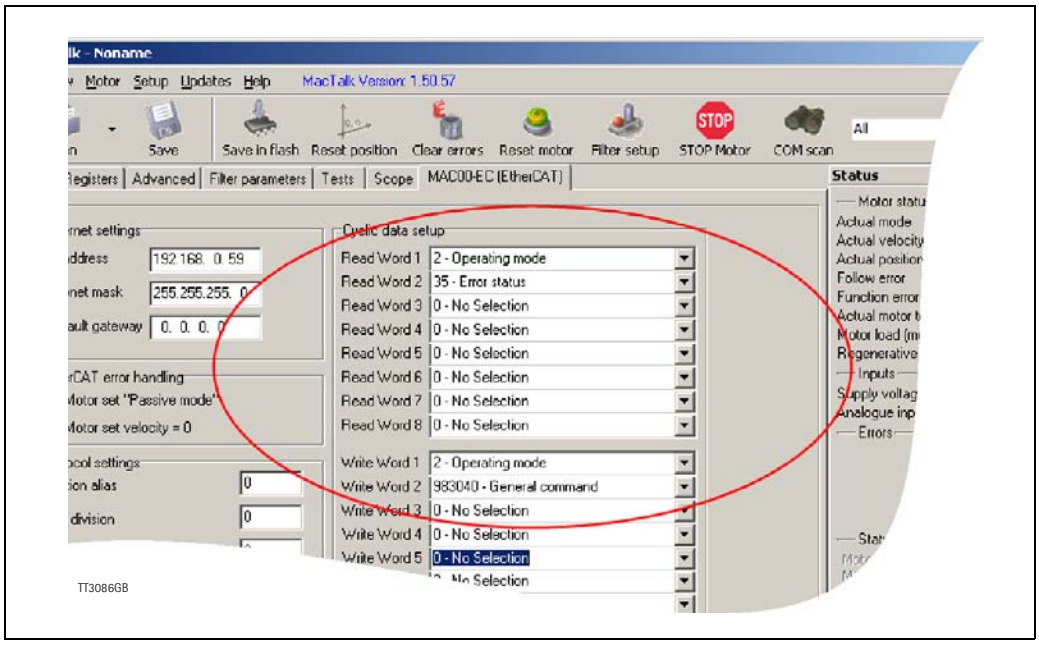
- Treat the transmitted Register 2 as "Requested_Mode" and the received register 2 as "Actual_Mode".
- When homing is wanted, set the "Requested_Mode" to one of the values 12, 13 or 14 depending of the requested homing mode (12 = Torque based zero search mode (only MAC motors). 13 = Forward/only zero search mode. 14 = Forward+backward zero search mode (only MAC motors) .). For a comprehensive description of the homing modes, refer to the general MAC motor manual - LB0047-xxGB.
- Observe that the "Actual_Mode" is changing to the homing mode. Now the module is blocking cyclic writes TO the motor. Cyclic reads is still active.
- Wait for register 35 "Error status" bit 4 to be active =IN_POSITION. (Indicates that homing is finished).
- Then change "Requested_Mode" to whatever needed. The blocking of cyclic writes to the motor is then released by the module.

4.8 Examples

4.8.5 Relative positioning.

There are a number of ways to do relative positioning, but the one explained here is very simple, and can be used with a constant distance, or exchangeable distance, to move every time it is requested.

Preconditions:
Place the module command register (register 983040 in MacTalk) in the cyclic write list. The cyclic setup, could for example look like this:



Procedure in the PLC:
1. Set up register P7 in motor to requested relative offset.
2. Make sure one net cycle has passed, so P7 resides in the motor.
3. Issue command 0x800000F1 (0x80000071 if MIS34x) in module command register (register 983040 in MacTalk).
4. Make sure one net cycle has passed, so command is interpreted by the motor.
5. Set module command register to zero. This will prepare the Ethernet module for new commands.
6. If needed then monitor register 35 (Error status): When bit 4 is set (in position), then the move is finished.
7. When a new relative move is requested, go to step 3.

You may also have the P7 register in the cyclic write list, thereby enabling easy change of the relative distance to move.

5 MAC00-EL4 POWERLINK® module

5.1 Introduction to POWERLINK®

Expansion module MAC00-EL4 front plate

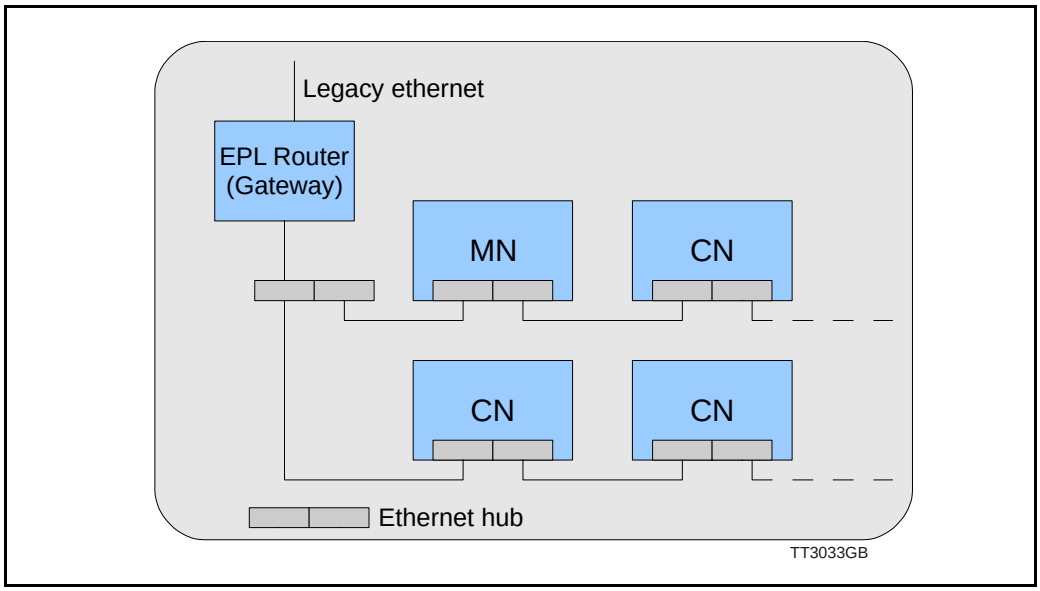
5.1.1 Introduction.

Ethernet Powerlink (EPL) is a proven technology, working in real applications world-wide. It embraces standard Ethernet technology and infrastructure, uses standard CAT5 shielded cabling and does not compromise standard Ethernet frames in order to achieve its results.
Ethernet Powerlink is a truly open technology independently managed by the Ethernet Powerlink Standardization Group (http://www.ethernet-powerlink.org).
Powerlink operates as a protected segment by design, and connects to a none-deterministic Ethernet network via a gateway/router device. This gateway acts as a defensive barrier against attacks by providing fire wall security measures.

5.1 Introduction to POWERLINK®

Unlike standard Ethernet, the Slot Communication Network Management (SCNM) ensures that only one node is accessing the network at a time. The schedule is divided into an isochronous phase and an asynchronous phase. During the isochronous phase, time-critical data is transferred, while the asynchronous phase provides bandwidth for the transmission of data that is not time-critical. The Managing Node (MN) grants access to the physical medium via dedicated poll request messages. As a result, only one Controlled Node (CN) has access to the network at a time, and thus no collisions occur. Ethernet POWERLINK applies the same protocol technology as CANopen. It defines SDOs (Service Data Objects), PDOs (Process Data Objects) and the Object Dictionary structure to manage the parameters.

For general technical data please see *MAC00-EL4 Powerlink - Technical specifications, page 169*.



5.1 Introduction to POWERLINK®

5.1.2 Abbreviations

Following general used terms are useful to know before reading the following chapters.

100Base-Tx	100 MBit Ethernet on twisted pairs
ASnd	Asynchronous Send (POWERLINK frame type)
CAN	Controller Area Network
CANopen	Application layer protocol used in automation.
CN	Controlled Node (slave on Ethernet Powerlink network)
EN	Exception New (flag in POWERLINK frame)
EMCY	Emergency Object.
EPL	Ethernet PowerLink
EPSG	Ethernet PowerLink Standardisation Group
ID	Identifier
IP	Internet Protocol - IP address ~ the logical address of the device, which is user configurable.
MAC	Media Access Controller - MAC address ~ the hardware address of the device.
MacTalk	A windows PC based program supplied from JVL. This is an overall program to install, adjust and monitor the function of the motor and a module installed in the motor.
MN	Managing Node (master on Ethernet Powerlink network)
NAT	Network Address Translation (used in EPL router, to reach destinations outside EPL segment)
NMT	Network Management
PDO	Process Data Object (for cyclic data)
PReq	Poll Request. A frame used in the isochronous phase of the cyclic communication. With Poll Request, the MN requests the CN to send its data.
PRes	Poll Response. A frame used in the isochronous phase of the cyclic communication. The CN responses with a Poll Response frame when it receives a Poll Request from the MN.
SCNM	Slot Communication Network Management; In a POWERLINK network, the MN allocates data transfer time for data from each node in a cyclic manner within a guaranteed cycle time. Within each cycle there are slots for Isochronous Data, and for Asynchronous Data for ad-hoc communication. The SCNM mechanism ensures that there are no collisions during physical network access in any of the net worked nodes thus it provides deterministic communication via Legacy Ethernet.
SDO	Service Data Object (for acyclic data)
SoA	Start of Asynchronous (POWERLINK frame type)
SoC	Start of Cyclic (POWERLINK frame type)
TCP	Transfer Control Protocol (an IP based protocol used widely on the internet)
UDP	User Datagram (an IP based protocol used widely on the internet)
XDD	File extension for the device description file.
XML	Extensible Markup Language - used for the device description file.

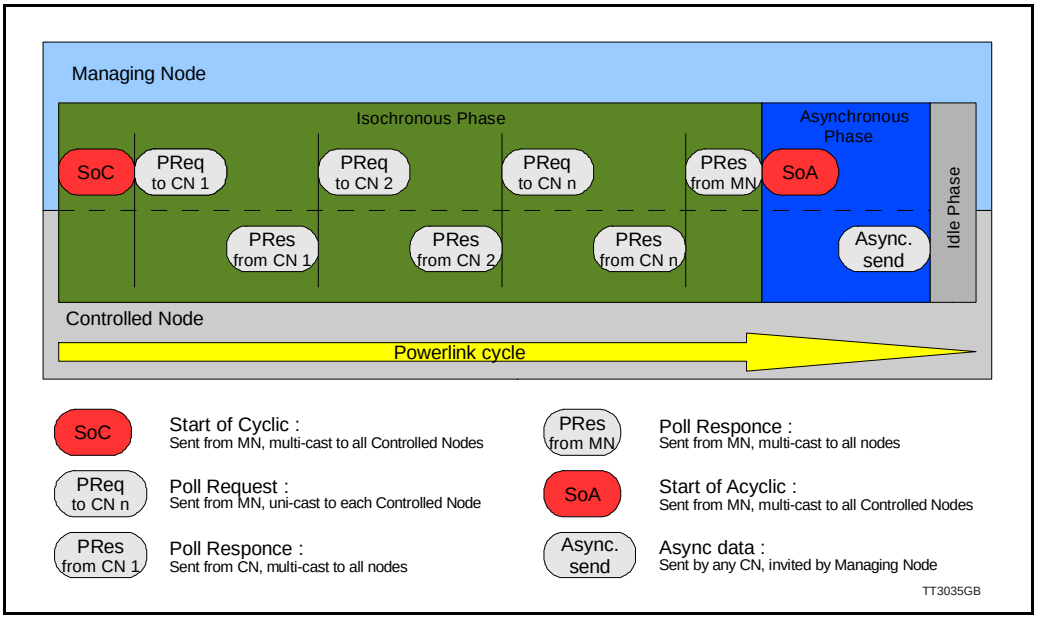
5.2 Protocol specifications

5.2.1 Ethernet Powerlink communication

In an Ethernet POWERLINK network, one of the nodes, for example a PLC, is designated to function as the MN, the master in the network. All other devices operate as CNs, slaves in the network. The MN defines the clock pulse for the synchronization of all devices and manages the data communication cycle. In the course of one clock cycle within which all nodes are addressed, the MN sends Poll Requests (PReq) to all CNs, one after another. They reply immediately to the prompts with Poll Responses (PRes).
The following time phases exist within one cycle:

- Isochronous phase
- Asynchronous phase
- Idle phase

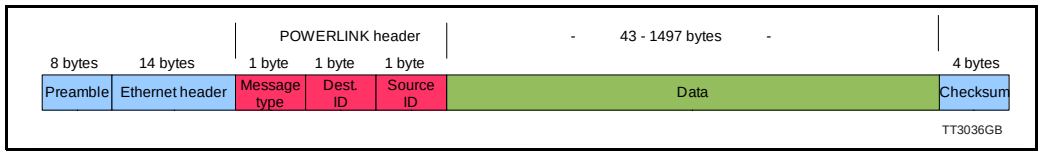
The MN first sends a Start of Cycle Frame (SoC) signal to all CNs to synchronize the devices. Payload data exchange then proceeds in the isochronous phase. The asynchronous phase, allows the transfer of large packets that are not time-critical, for example parameterisation data or transfer of IP-based protocols like TCP or UDP. The Idle phase can be 0. It's possibly for the MN to multiplex the time slots in the isochronous phase, in order to service some CN's more often than others. During system start-up the MN applies a reduced POWERLINK cycle, without the isochronous phase, in order to configure the CNs with SDO communication.
For further information, please refer to the Ethernet POWERLINK communication profile specification "EPSG_DS_301_V-1-1-0_01.pdf", available at the EPSG website http://www.ethernet-powerlink.org.



5.2 Protocol specifications

5.2.2 Ethernet POWERLINK® frame structure

POWERLINK messages are encapsulated in Ethernet II frames. The length of the frame is restricted to the configured size, in order to guarantee the cycle time. Ethernet frames have a minimum length of 64 bytes and a maximum of 1518 (exclusive preamble). The Ethernet POWERLINK header contains only 3 bytes. Message type, destination ID and Source ID. That leaves up to 1497 bytes of payload.



5.2.3 Ethernet POWERLINK CN State machine

In Ethernet POWERLINK, a Controlled Node starts up by a common initialization process. All the states are valid when the device is powered and they are sub-states of the NMT_GS_POWERED superstate.

NMT_GS_INITIALISATION

After system start, the device automatically assumes this state and network functionality begins. NMT_GS_INITIALISATION and all its sub-states are only internal states of the device. In the NMT_GS_RESET_CONFIGURATION sub-state, the node address of the device is identified and it is determined whether it is configured as a MN or CN. The JVL MAC00-ELx is a CN and thus, it enters the NMT CN state machine in the NMT_GS_COMMUNICATING super-state.

NMT_GS_COMMUNICATING

NMT_CS_NOT_ACTIVE

This is a none-permanent state that allows a starting node to recognize the current network state. Time out for SoC, PReq, PRes and SoA frames trigger the device to enter state NMT_CS_BASIC_ETHERNET.

The NMT_CS_PREOPERATIONAL_I

Transition from NMT_CS_NOT_ACTIVE to NMT_CS_PRE_OPERATIONAL_1 is triggered by a SoA or SoC frame being received. In this state CN may send a frame only if the MN has authorized it to do so by a SoA command. There is no PDO communication in this state. Receiving a SoC frame triggers the transition from NMT_CS_PREOPERATIONAL_1 to NMT_CS_PREOPERATIONAL_2.

The NMT_CS_PREOPERATIONAL_2

In this state PReq and PRes data may be invalid because PDO mappings may differ. In NMT_CS_EPL_MODE, error recognition (for example, loss of SoC or PReq) always triggers the transition to NMT_CS_PREOPERATIONAL_1.

The NMT_CS_READY_TO_OPERATE

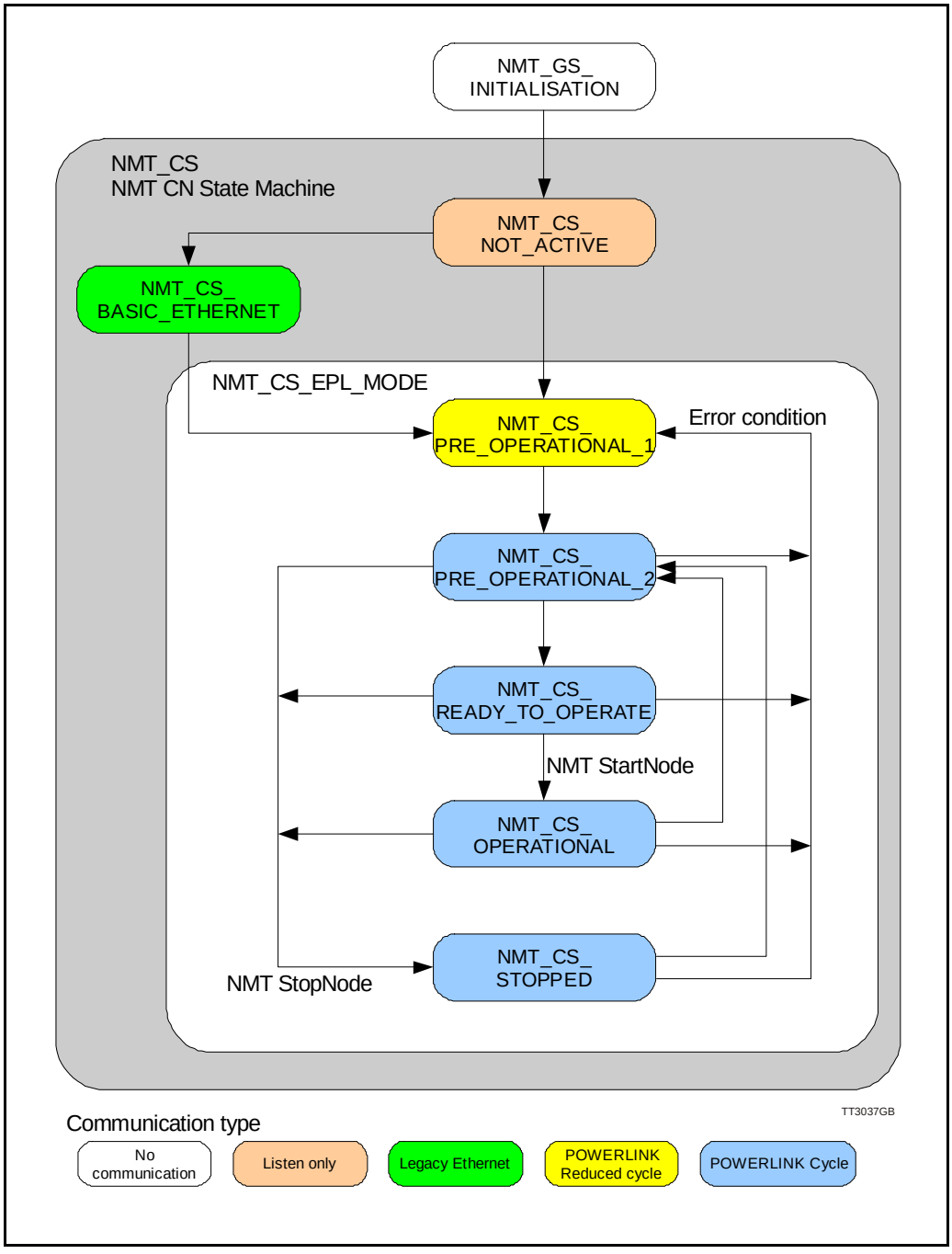
In this state, the CN signals that it is ready to operate to the MN. It responds to the PReq query of the MN by sending a PRes frame.

The NMT_CS_OPERATIONAL

NMT Start Node command triggers the transition from NMT_CS_READY_TO_OPERATE to the NMT_CS_OPERATIONAL. This is the normal operating state of the CN.

The NMT_CS_STOPPED

This state is used for controlled shutdown of a selected CN while the system is still running. In this state, the CN does not participate in cyclic frame exchange, but it still observes SoA frames.



5.2 Protocol specifications

5.2.4 Application layer communication

The application layer communication protocol in Ethernet POWERLINK is based on the CANopen DS 301 communication profile. The protocol specifies the Object Dictionary in the adapter module, in addition to communication objects for exchanging cyclic process data and acyclic messages.
The MAC00-ELx module uses the following message types:

- Process Data Object (PDO). The PDO is used for cyclic I/O communication, in other words, process data.

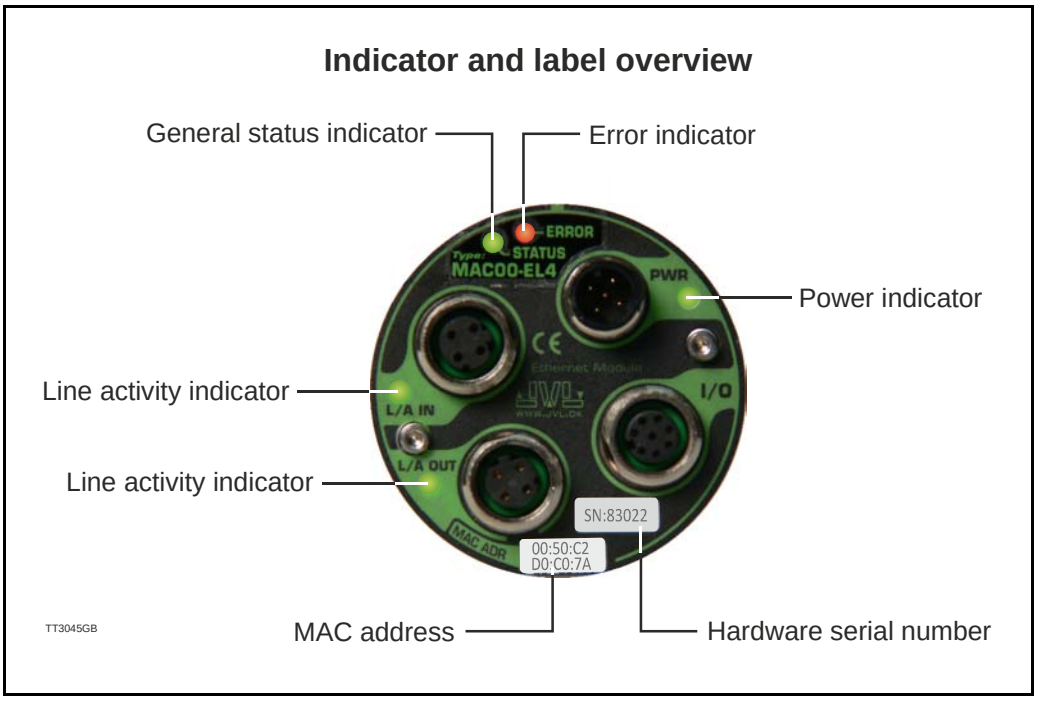
- Service Data Object (SDO). The SDO is used for much slower acyclic data transmission.

- NMT response services. Used for identity and status signalling during start-up and runtime.

5.3 Commissioning

5.3.1 Indicator LED's - description.

The LED's are used for indicating states and faults of module. There is one power LED, two link/activity LED's (one for each Ethernet connector), and 2 status LED's.



LED indicator descriptions

LED Text	Colour	Constant off	Constant on	Blinking	Single flash	Double flash	Triple flash	Flickering
L/A IN	Green	No valid Ethernet connection.	Ethernet is connected.	-	-	-	-	Activity on line
L/A OUT	Green	No valid Ethernet connection.	Ethernet is connected.	-	-	-	-	Activity on line
STATUS	Green	NMT_CS_NOT_ACTIVE	NMT_CS_OPERATION-AL	NMT_CS_STOPPED	NMT_CS_PREOPERATIONAL1	NMT_CS_PREOPERATIONAL2	NMT_CS_READY_TO_OPERATE	NMT_CS_BASIC_ETHERNET
ERROR	Red	No error	Error					Booting error
PWR	Green	Power is not applied.	Power is ap-plied to both motor and module.					Power is applied to module but no communi-cation with motor.

Notes:
Blinking: Flashing with equal on and off periods of 200ms (2.5Hz). **Single flash:** Repeating on for 200ms and off for 1s. **Double flash:** Two flashes with a period of 200ms followed by 1s off period. **Triple flash:** Two flash-es with a period of 200ms followed by 1s off period.**Flickering:** Rapid flashing with a period of approximately 50ms (10 Hz).

5.3.2 Mechanical installation

The network cables must be connected to the two M12 connectors (marked “L/A IN” and “L/A OUT”) on the module. The cable from the MN is connected to either of the two ports. In the line topology, if there are more slave devices in the same line, the next slave device is connected to the second port. If there is a redundant ring, the second port of the last slave device is connected to the second port of the MN.
See also the illustration in the chapter *Introduction., page 88*
Standard CAT 5 FTP or STP cables can be used. It is not recommended to use UTP cables in industrial environments, which is typically very noisy.

5.3.3 Quick start

This section describes the steps to configure the PLC, B&R X20 CP1485, with B&R Automation Studio PC software, so that it can be used to control the drive.

Set node ID

1. Connect the RS232 communication cable.
2. Apply power to the motor, and make sure the PWR LED is lit.
3. Open MacTalk and select the “MAC00-EL (Powerlink)” tab.
4. Change the last number in the IP address (= node ID), to one that doesn't conflict with other devices on the sub net.
5. Press “Apply and save”.

Installation

6. Connect an Ethernet RJ45-M12 cable to IF3 on the X20 and to L/A IN or L/A OUT on MAC00-ELx.
7. Connect power to the X20, and communication cable from the PC with B&R Automation Studio installed to the X20 PLC (either Ethernet or RS232).
8. Make sure power is applied to all devices.

PLC configuration

9. Create a new project in Automation Studio for your PLC, or open an existing project. See B&R documentation for more information.
10. In the Project Explorer window, open the Physical View tab
11. Right-click the node representing the CPU (in this example, X20CP1485-1), and in the pop-up menu, select Open IF3 POWERLINK Configuration. The POWERLINK Configuration window is opened.
12. Make sure that “Activate POWERLINK communication” is set to “on”.
13. Close the window and save changes.

Add the XDD file (contains info on the capabilities of the device)

14. In the Tools menu of Automation Studio, select Import fieldbus device...
15. In the Open window find and select the "00000117_MAC00-ELx.xdd" file, and click Open.
    This link can be used : http://www.jvl.dk/default.asp?Action=Details&Item=428

(continued next page)

Associating with MAC00-ELx

16. In the physical view of the project explorer window, right click the CPU node and click Open POWERLINK in the pop-up menu.
17. Right click IF3 in the opened CPU POWERLINK window, and click Insert in the pop-up menu.
18. Select "MAC00-ELx", situated under POWERLINK devices, and click Next.
19. Enter the node ID of the device (set earlier with MacTalk) and optionally a name, and click Next.
20. The "MAC00-ELx" should now be visible in the physical view of the project explorer window.

Building project and transfer to PLC

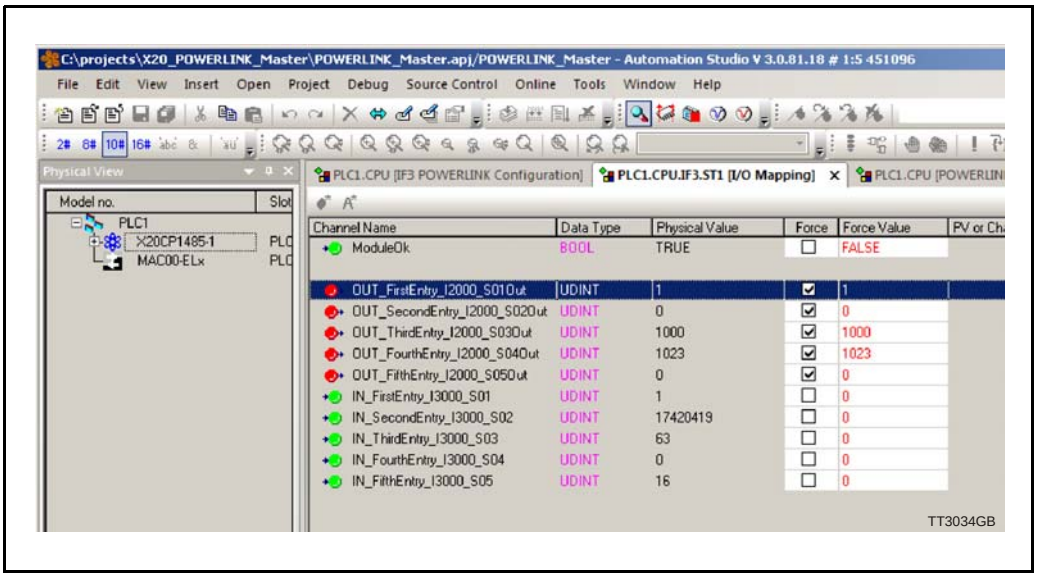
21. Select Build configuration in the Project menu.
22. When the build is finished then click the Transfer button.
23. There may appear a warning. Just ignore and click OK.

Investigating cyclic data

24. Right click "MAC00-ELx" in the physical view of the project explorer window and click Open I/O Mapping.
25. In the View menu click Monitor.
26. You should now be able to see the cyclic I/O registers like in the below picture.
27. If Force is checked for the cyclic outputs, then it's possibly to set register values in the Force Value column that is transferred to the motor.

Start motor

28. If the default register settings is not changed it is possibly to start motor by entering values in the Force Value column.
29. Enter 1023 in OUT_FourthEntry (Torque = 300%).
30. Enter 1000 in OUT_ThirdEntry (477 RPM if MAC140).
31. Enter 1 in OUT_FirstEntry (Mode = Velocity).



5.4 Ethernet POWERLINK objects

5.4.1 Process data objects

PDO's (Process Data Objects) are used for cyclic transfer of time-critical process data between master and slaves. Tx PDOs are used to transfer data from the slave to the master and Rx PDOs to transfer data from the master to the slave.

PDO 21
PDO 21 is fully user configurable. There is one receive PDO and one transmit PDO.
It is possibly to set up five, 32 bit registers in each direction.
The setup is done with MacTalk or via SDO object 0x2011 subindex 16-31. It requires a save in flash and a power cycle before the new configuration are used. If the configuration of the PDO's, is not altered by the user, the MAC00-ELx uses the default mapping shown in the tables below.

If module registers is placed in cyclic R/W, then the register number has to be calculated as follows:

Register number = 65536 x sub index.
Example: module command (sub-index 15) = 65536 x 15 = register **983040**

When module registers (register numbers above 65535) are chosen, they **have** to be placed **after** the motor registers in the list of cyclic registers.

NB! If an index is set to zero (No selection), then the following indexes is discarded. Thereby computing resources in the drive are released, which makes much faster cycle times possibly. Please see next paragraph.

Default registers in transmit PDO 21 (Slave > Master) / Read words in MacTalk

Object index	Register no.	Motor register short	Motor register description
0	2	MODE_REG	Operating mode
1	10	P_IST	Actual position
2	12	V_IST	Actual velocity
3	169	VF_OUT	Actual torque
4	35	ERR_STAT	Status bits

Default registers in receive PDO 21 (Master > Slave)

Object index	Register no.	Motor register short	Motor register description
0	2	MODE_REG	Operating mode
1	3	P_SOLL	Target position
2	5	V_SOLL	Maximum velocity
3	7	T_SOLL	Maximum torque
4	-	-	-

Please notice: Even though all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

5.4 Ethernet POWERLINK objects

5.4.2 Minimum cycle time

The minimum cycle time is the minimum amount of time between each cyclic request (PDO) on the Ethernet.
If operating with values lower than those listed, data loss will occur.

No. of motor registers transmitted in each direction	Motor series MAC050 - MAC141	Motor series MAC400 to MAC3000
1/1	4mS *	360µS *
2/2	8mS *	395µS *
3/3	12mS *	430µS *
4/4	16mS *	465µS *
5/5	20mS *	500µS *

* The minimum cycle times, is only valid if not sending any SDO requests while in any operating mode. MODULE registers can be appended as the last registers in the list, at no extra timing cost. If motor register 35 is not in the list it will be added internally anyway, and has to be added to the minimum cycle time with 2.0ms if MAC050-MAC141, and with 30μs if MAC400-MAC3000.

5.4 Ethernet POWERLINK objects

5.4.3 Service Data Objects

Service Data Objects (SDOs) are mainly used for transferring non time-critical data, for example, identification, configuration and acyclic data.

5.4.4 Object Dictionary

An important part of the protocol is the Object Dictionary, which is different objects specifying the data layout. Each object is addressed using a 16-bit index and possibly a sub index. There are some mandatory objects and some manufacturer specific objects. The objects in the Object Dictionary can be accessed with SDO services.

Mandatory objects:

Name	Index (hex)	Sub Index	Data Type	Read only	Default	Description
Device type	1000		UNSIGNED32	X	0x0	Contains information about the device type.
Error Register	1001		UNSIGNED8	X		This is the mapping error register, and it is part of the emergency object. If some of the sub index are high, an error has occured.
		0				Generic error. Mandatory
		1				Current
		2				Voltage
		3				Temperature
		4				Communication (Overrun)
		5				Device profile specific
		6				Reserved
		7				Manufactor specific
Identity object	1018		IDENTITY	X		Contain general information about the module
		0	1..4	X	4h	Number of entries. Mandatory
		1	UNSIGNED32	X	0x0117	Vendor ID, contains a unique value allocated to each manufactor. 117h is JVLs vendor ID. Mandatory.
		2	UNSIGNED32	X	0x0200	Product Code, identifies a specific device version. The MAC00-EL4 has the product code 200h
		3	UNSIGNED32	X	0x20020	Revision number.
		4	UNSIGNED32	X		Serial number

5.4 Ethernet POWERLINK objects

5.4.5 Manufacturer specific objects.

The manufacturer specific objects, provides access to all module registers, and all motor registers, as well as a module command object.

	Index (hex)	Sub Index	Type	Read only	Default	Description
Module command	2010	0	UNSIGNED32			Module command object. See possible commands below.
Module parameters	2011	0	UNSIGNED8	X	63	Subindex count
		1	UNSIGNED32	X		High 16 bit of MAC address (placed in low 16 bit of word)
		2	UNSIGNED32	X		Low 32 bit of MAC address
		3	UNSIGNED32		192.168.100.xxx	IP address / Node ID (The least significant 8 bits is node ID)
		4	UNSIGNED32	X	255.255.255.0	Net mask
		5	UNSIGNED32	X	192.168.100.254	Gateway
		6	UNSIGNED32		0x0	Setup bits
		7	UNSIGNED32		0	Digital outputs on module
		8-14	UNSIGNED32		-	Reserved for future use
		15	UNSIGNED32		-	Command register
		16	UNSIGNED32		2	Register no. to place in TxPDO 21, position 1.
		17	UNSIGNED32		10	Register no. to place in TxPDO 21, position 2.
		18	UNSIGNED32		12	Register no. to place in TxPDO 21, position 3.
		19	UNSIGNED32		169	Register no. to place in TxPDO 21, position 4.
		20	UNSIGNED32		35	Register no. to place in TxPDO 21, position 5.
		21	UNSIGNED32		-	Reserved for future use
		22	UNSIGNED32		-	Reserved for future use
		23	UNSIGNED32		-	Reserved for future use
		24	UNSIGNED32		2	Register no. to place in RxPDO 21, position 1.
		25	UNSIGNED32		3	Register no. to place in RxPDO 21, position 2.
		26	UNSIGNED32		5	Register no. to place in RxPDO 21, position 3.
		27	UNSIGNED32		7	Register no. to place in RxPDO 21, position 4.
		28	UNSIGNED32		0	Register no. to place in RxPDO 21, position 5.
		29	UNSIGNED32		-	Reserved for future use
		30	UNSIGNED32		-	Reserved for future use
		31	UNSIGNED32		-	Reserved for future use
		32	UNSIGNED32	X	-	Module serial no.
		33	UNSIGNED32	X	-	Module hardware version
		34	UNSIGNED32	X	-	Module software version
		35	UNSIGNED32	X	-	No. of internal motor communication timeouts
		36	UNSIGNED32	X	-	No. of retry frames to motor
		37	UNSIGNED32	X	-	No. of discarded frames to the motor
		38	UNSIGNED32	X	-	Total no. of frames to motor
		39-46	UNSIGNED32	X	-	Reserved for future use
		47	UNSIGNED32	X	-	Digital inputs on module
		48	UNSIGNED32	X	-	Status bits
		49-63				Reserved for future use
Motor parameters	2012	0	UNSIGNED8	X	254	Subindex count
		N	UNSIGNED32			Access to the motor parameter n

Note: Module parameters are not automatically saved to permanent memory after a change. The parameters can be saved permanently by applying a "Save parameters to flash" command afterwards.

5.4 Ethernet POWERLINK objects

5.4.6 Object 0x2010 - Subindex 0

This object is used for sending commands to the module and is write only. The possible commands are listed in the table below.

Command no.			Command description	
	Hex	Dec	MAC050 - MAC141	MAC400 – MAC3000
Module only commands				
	0x 0000 0000	0	No operation	< Same as
	0x 0000 0001	1	Reset the module	< Same as
	0x 0000 0010	16	Save module parameters to flash	< Same as
Synchronized commands				
	0x 0000 0101	257	Simultaneous reset of the motor and the module	< Same as
	0x 0000 0110	272	Save the motor parameters in flash memory, and do a re-sync. of internal communication afterwards.	< Same as
Motor only normal commands *(via module cmd register)*				
	0x 8000 0001	2147483649	Reset motor *(not recommended, use synchronized version instead)*.	< Same as
	0x 8000 0002	2147483650	Save motor parameters in flash and reset motor *(not recommended, use synchronized version instead)*.	< Same as
Motor only FastMac commands *(via module cmd register)*				
	0x8000 00E0	2147483872	No operation	< Same as
	0x8000 00E1	2147483873	Reset error *(Clear error bits in motor register 35)*	< Same as
	0x8000 00E2	2147483874	P_SOLL = 0	< Same as
	0x8000 00E3	2147483875	P_IST = 0	< Same as
	0x8000 00E4	2147483876	P_FNC = 0	< Same as
	0x8000 00E5	2147483877	V_SOLL = 0	< Same as
	0x8000 00E6	2147483878	T_SOLL = 0	< Same as
	0x8000 00E7	2147483879	Reset IN_POS, AC C,DEC	< Same as
	0x8000 00E8	2147483880	P_FNC = (FLWERR - P7) * 16	< Same as
	0x8000 00E9	2147483881	P_FNC = (FLWERR - P8) * 16	< Same as
	0x8000 00EA	2147483882	Reserved	< Same as
	0x8000 00EB	2147483883	Reserved	< Same as
	0x8000 00EC	2147483884	Activate P1,V1,A1,T1,L1,Z1	< Same as
	0x8000 00ED	2147483885	Activate P2,V2,A2,T2,L2,Z2	< Same as
	0x8000 00EE	2147483886	Activate P3,V3,A3,T3,L3,Z3	< Same as
	0x8000 00EF	2147483887	Activate P4,V4,A4,T4,L4,Z4	< Same as
	0x8000 00F0	2147483888	Start search zero	< Same as
	0x8000 00F1	2147483889	P_SOLL = P_IST + P7;	P_SOLL = P_IST + P7 – FLWERR;
	0x8000 00F2	2147483890	P_SOLL = P_IST + P8;	P_SOLL = P_IST + P8 – FLWERR;
	0x8000 00F3	2147483891	Reserved	< Same as
	0x8000 00F4	2147483892	Select absolute position mode	< Same as
	0x8000 00F5	2147483893	Select relative position mode using P_SOLL	< Same as
	0x8000 00F6	2147483894	Select relative position mode using P_FNC	< Same as
	0x8000 00F7	2147483895	Synchronize position manually using absolute new values. P_IST = P_NEW; P_SOLL = P_NEW; P_FUNC = P_NEW * 16;	Synchronize position manually using absolute new values. P_IST = P_NEW; P_SOLL = P_NEW; P_FNC = (P_NEW + FLWERR)*16;
	0x8000 00F8	2147483896	Synchronize position manually using relative new values. (basically offset the position range with the value of P_NEW). P_IST = P_IST + P_NEW; P_SOLL = P_SOLL + P_NEW; P_FUNC = P_FUNC + (P_NEW * 16);	< Same as
	0x8000 00F9	2147483897	No operation	< Same as
	0x8000 00FA	2147483898	No operation	< Same as
	0x8000 00FB	2147483899	No operation	< Same as
	0x8000 00FC	2147483900	No operation	< Same as
	0x8000 00FD	2147483901	Reserved	< Same as
	0x8000 00FE	2147483902	Reserved	< Same as
	0x8000 00FF	2147483903	Reserved	< Same as

5.4 Ethernet POWERLINK objects

5.4.7 Object 0x2011

The module registers is mapped to object 0x2011. The subindex 3, 6-31 is R/W, the rest is read only.

5.4.8 Object 0x2011 - Subindex 1 MAC address MSB.

The 2 most significant bytes of the module MAC address is placed here.

Bit	16-31	0-15
Output	Reserved	16 Most significant bits of MAC address.

5.4.9 Object 0x2011 - Subindex 2 MAC address LSB.

The 2 most significant bytes of the module MAC address is placed here.

Bit	0-31
Output	32 Least significant bits of MAC address.

5.4.10 Object 0x2011 - Subindex 3 IP address.

This is the combined IP address and node ID of the device. Only the node ID part is writeable the rest of the IP address is fixed.

Bit	24-31	16-23	8-15	0-7
I/O	192	168	100	Node ID

5.4.11 Object 0x2011 - Subindex 4 Netmask.

This is the netmask of the device. The netmask is fixed.

Bit	24-31	16-23	8-15	0-7
I/O	255	255	255	0

5.4 Ethernet POWERLINK objects

5.4.12 Object 0x2011 - Subindex 5 Gateway.

This is the gateway address of the device. The gateway address is also fixed.

Bit	24-31	16-23	8-15	0-7
I/O	192	168	100	254

5.4.13 Object 0x2011 - Subindex 6 Setup bits

This register is used to setup how the module should react on different events.

Bit	1-31	0
Output	Reserved	0 : Ethernet error handling = motor set passive mode 1 : Ethernet error handling = motor set velocity to 0

5.4.14 Object 0x2011 - Subindex 7 Digital outputs on module

With this object the digital outputs can be controlled.
The value written to this object is directly shown on the digital outputs.

Bit	2-31	1	0
Output	Reserved	Output2* (O2)	Output1* (O1)

* The availability of the outputs depends on the actual version of the module used. Example MAC00-EL4 only support Output 1 (O1).

5.4.15 Object 0x2011 - Subindex 15 Command register

Analogue to writing to object 0x2010. But this can be mapped in the RxPDO 21 if desired.

5.4.16 Object 0x2011 - Subindex 16-23 Register no. to place in TxPDO 21

These registers contain the numbers that define the registers which are in the TxPDO 21. That is the register's, which is transmitted from slave to master cyclically. If some of these registers are changed, it is necessary to issue a "save in flash" command and to reboot the device before the changes take effect.

5.4.17 Object 0x2011 - Subindex 24-31 Register no. to place in RxPDO 21

These registers contain the numbers that define the registers which are in the RxPDO 21. That is the register's, which is transmitted from master to slave cyclically. If some of these registers are changed, it is necessary to issue a "save in flash" command and to reboot the device before the changes take effect.

5.4.18 Object 0x2011 - Subindex 32-38

These registers contain HW, SW and communication information of the module.

5.4.19 Object 0x2011 - Subindex 47 Digital inputs on module

With this object the status of the 4 digital inputs can be read.

Bit	4-31	3	2	1	0
Input	Reserved	IN4	IN3	IN2	IN1

Note: Please notice that the number of inputs available is depending on which version of the module which is used.

5.4 Ethernet POWERLINK objects

5.4.20 Object 0x2011 - Subindex 48 Status bits

This register is used for miscellaneous information about the module.

Bit	8-31	7	0-6
Output	Reserved	1=No communication with the motor	Reserved

5.4.21 Object 0x2012

Object 0x2012 are for acyclic view or change of motor registers.

Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

5.5 Network Management Services

Ethernet POWERLINK Network Management (NMT) is node oriented and follows a master/slave relationship. MAC00-ELx is administered as an NMT slave by the master. Ethernet POWERLINK defines five categories of NMT services:

- NMT State Command Services
- NMT Managing Command Services (not supported)
- NMT Response Services
- NMT Info Services (not supported)
- NMT Guard Services (not supported)

NMT State Command Services
The MN controls the state of the CN via NMT State Command Services. See section Ethernet POWERLINK state machine for more information.

NMT Response Services
NMT Response Services are used by the MN to query NMT information from the CN, such as current state, error and setup data. Ethernet POWERLINK specifies the following NMT Response Services:

- NMT State Response
- IdentResponse
- StatusResponse

Via NMT State Response service, the CNs signals their states to the MN. IdentResponse Service is used by the MN to identify configured but unrecognized CNs at system start-up or after loss of communication. See Appendix: IdentResponse Frame for more information. The StatusResponse Service is used by the MN to query the current status of CNs that is not communicating isochronously. It is used for error signaling in runtime. If an error occurs, the EN (Error New) flag in the PRes frame is toggled. This notifies the MN that an error has occurred and the MN polls the CN for a StatusResponse that includes error information.

5.6 XML Device Description File

XML Device Description Files (XDD) are XML files that specify the properties of the slave device for the Ethernet POWERLINK master (MN). The description files contain information on the supported communication objects. XDD files for JVL Drives are available through your local JVL representative and http://www.jvl.dk.

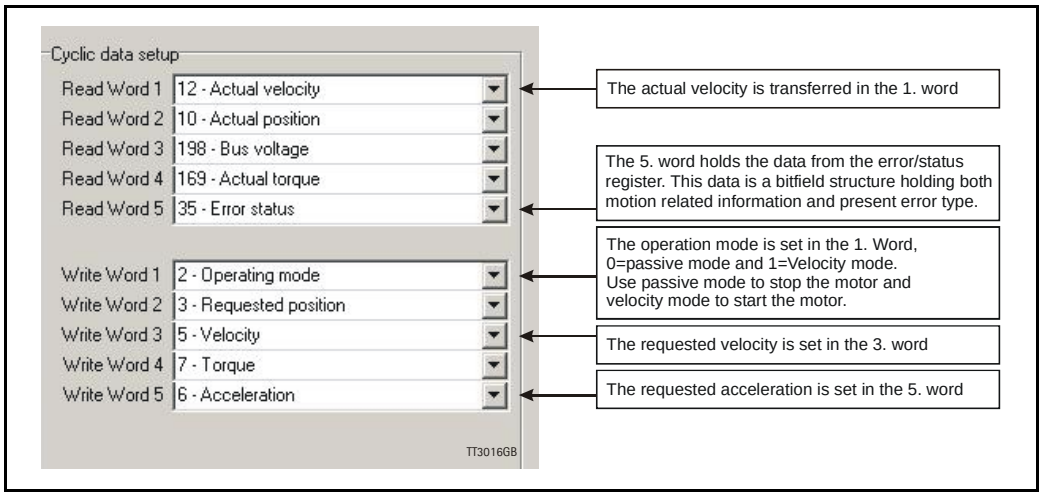
5.7 Examples

5.7.1 Running Velocity control

To use the JVL motor in velocity mode the following registers are basically of interest.

1. "Mode" - Mode register register 2
2. "V_SOLL" - Velocity register 5
3. "A_SOLL" - Acceleration register 6
4. "Error/Status" - Error and status register 35

So, to control these registers the cyclic data needs to configured.
From MacTalk the setup is configured as this.



With the settings illustrated above we initiate the velocity mode by writing 0x1 to the first word-value, this is velocity mode.

From the Master the registers is accessed using the PDO21 and accessing the registers R/W on words 1-5.

Since different PLC's have different methods of implementation the basic steps is described in the following.

1. Set the needed velocity. V_SOLL = V x 2.77 [rpm]
   Ex. We need the motor to run with a constant speed of 1200 RPM. So, V_SOLL = 1200/2,77 = 433 cnt/smp
2. Set the needed acceleration. A_SOLL = A x 271 [RPM/s^2]
   Ex. We need the motor to accelerate with 100000 RPM/s^2 so, A_SOLL = 100000/271 = 369 cnt/smp^2.
3. Now set the motor into velocity mode and thereby activate the motor.
   Ex. The motor needs to be activated by setting it into velocity mode, so we need to set the mode register to the value 1. Mode = 1 which is velocity mode, now the motor will use the acceleration and the velocity just configured.

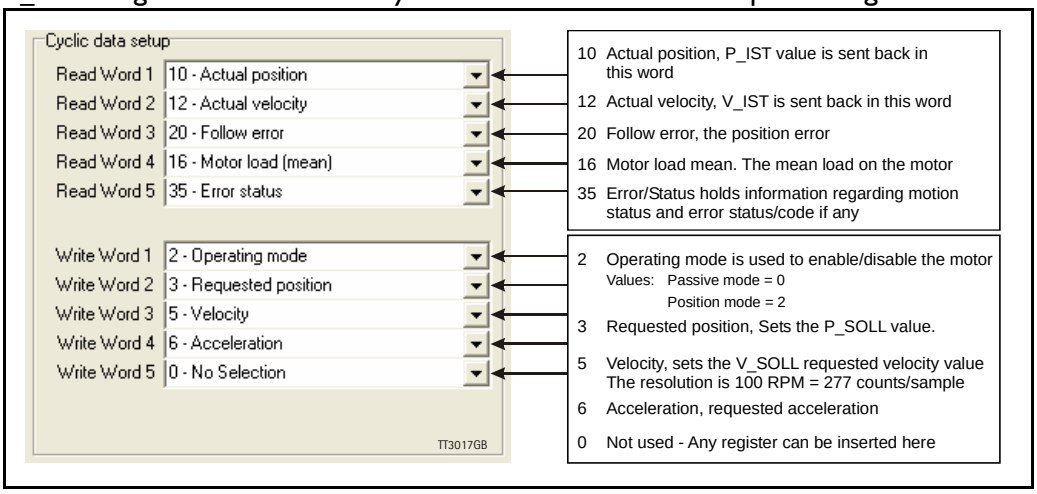
Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

5.7.2 Running Position control

Running the motor in position control requires that the mode register is set for position control. The following registers is of particular interest when position mode is used.

1. "Actual position" -P_IST, register 10
2. "Actual velocity" -V_IST, register 12
3. "Follow error" - The actual position error, register 20
4. "Motor load mean" - average motor load, register 16
5. "Error/Status" -register 35
6. "Requested position" -P_SOLL, register 3
7. "Requested velocity" -V_SOLL, register 5
8. "Requested acceleration" -A_SOLL, register 6

In this mode the position is controlled by applying a requested position to the "P_SOLL" -register and the actual position is monitored in the "P_IST" register. The V_SOLL and A_SOLL registers sets the velocity and acceleration used when positioning occurs.



5.7.3 General considerations

The register 35 in the motor holds information on the actual error/status. So it is crucial that this register is configured in the cyclic data and thereby obtained and monitored in the Master. In case of an error situation the motor will stop and the cause will be present in the register 35 and hence in the I/O -data.
This register also holds information on the motion status such as:

- In position, bit 4
- Accelerating, bit 5
- Decelerating, bit 6

Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

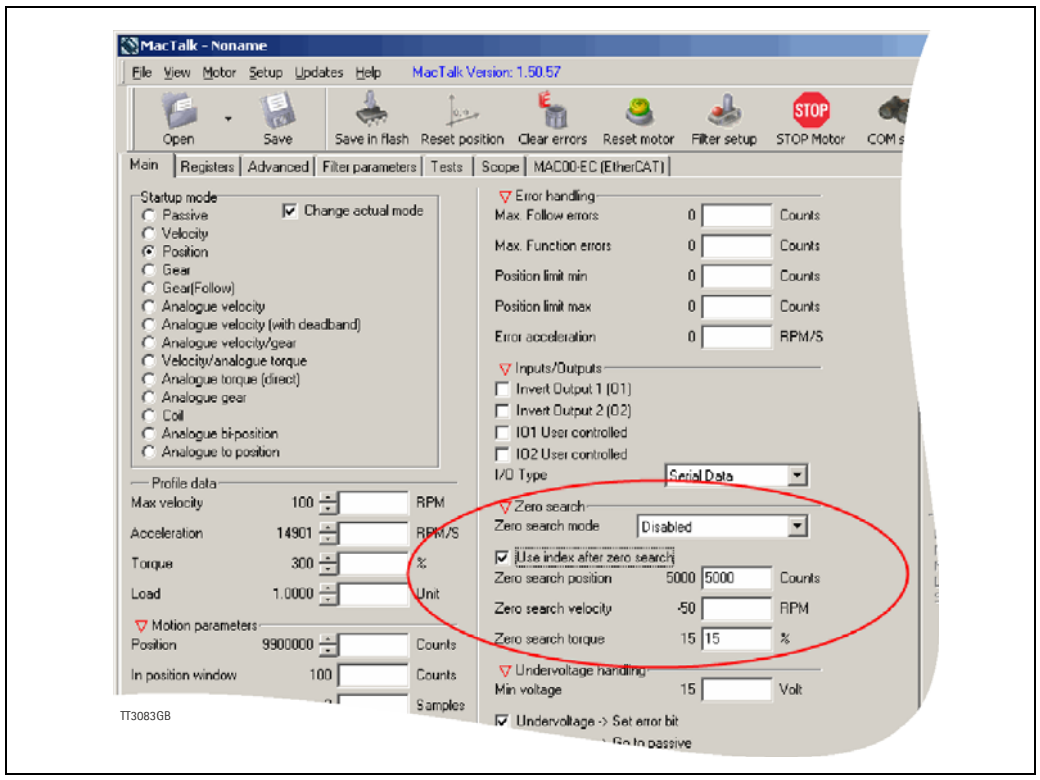
The JVL motor is basically put into a working mode and into a passive mode where the motor axle is de-energized, by setting register 2 into either 0 = "passive mode" or into one of the supported modes.

Example.
1= "Velocity mode" / 2= "Position mode" / etc.

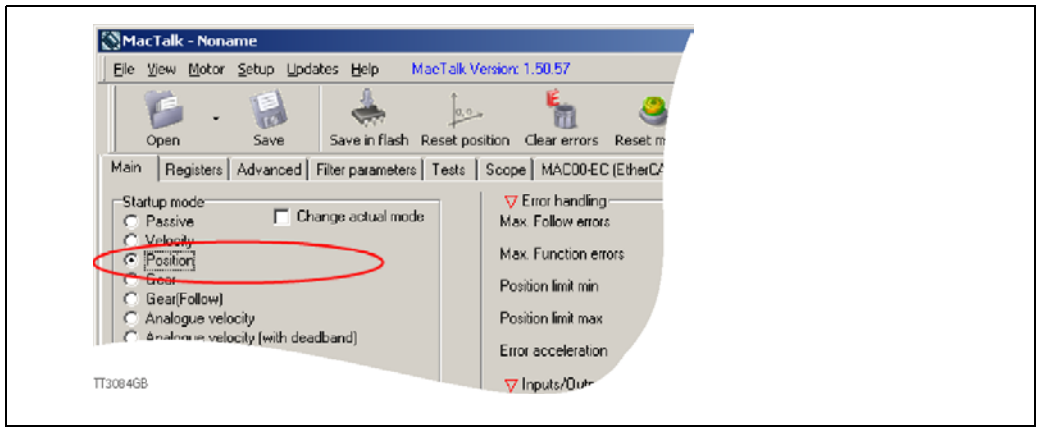
So in order to Stop or Start the motor this register can be supported in the I/O data or by sending an SDO message.

5.7.4 Homing using only cyclic I/O (JVL profile).

When doing a homing (Zero search), with only cyclic I/O, some preconditions have to be met:
Zero search position, zero search velocity and zero search torque (torque only for MAC motors) has to be set in MacTalk in the "Main" tab, and saved in flash in the motor once and for all.

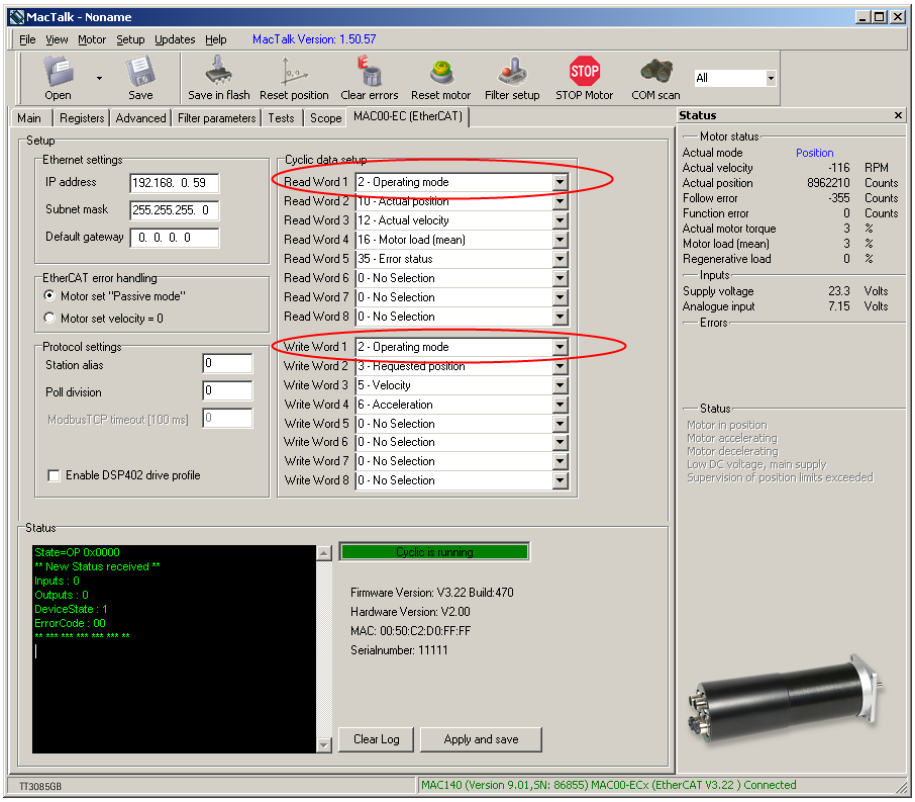


Startup mode should be set to position, for the motor to stay in position after the homing sequence. And this setting should also be saved in flash.



5.7 Examples

Register 2 (Operating mode) has to be present in BOTH the cyclic read words and cyclic write words.



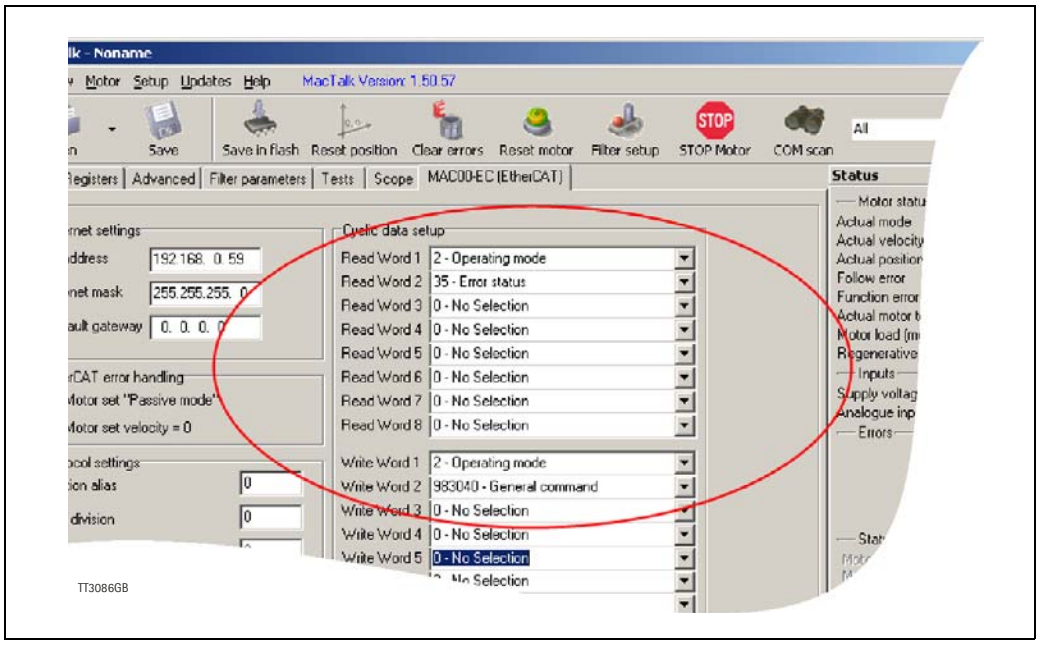
Procedure in the PLC:

- Treat the transmitted Register 2 as "Requested_Mode" and the received register 2 as "Actual_Mode".
- When homing is wanted, set the "Requested_Mode" to one of the values 12, 13 or 14 depending of the requested homing mode (12 = Torque based zero search mode (only MAC motors). 13 = Forward/only zero search mode. 14 = Forward+backward zero search mode (only MAC motors) .). For a comprehensive description of the homing modes, refer to the general MAC motor manual - LB0047-xxGB.
- Observe that the "Actual_Mode" is changing to the homing mode. Now the module is blocking cyclic writes TO the motor. Cyclic reads is still active.
- Wait for register 35 "Error status" bit 4 to be active =IN_POSITION. (Indicates that homing is finished).
- Then change "Requested_Mode" to whatever needed. The blocking of cyclic writes to the motor is then released by the module.

5.7.5 Relative positioning.

There are a number of ways to do relative positioning, but the one explained here is very simple, and can be used with a constant distance, or exchangeable distance, to move every time it is requested.

Preconditions:
Place the module command register (register 983040 in MacTalk) in the cyclic write list. The cyclic setup, could for example look like this:



Procedure in the PLC:
1. Set up register P7 in motor to requested relative offset.
2. Make sure one net cycle has passed, so P7 resides in the motor.
3. Issue command 0x800000F1 (0x80000071 if MIS34x) in module command register (register 983040 in MacTalk).
4. Make sure one net cycle has passed, so command is interpreted by the motor.
5. Set module command register to zero. This will prepare the Ethernet module for new commands.
6. If needed then monitor register 35 (Error status): When bit 4 is set (in position), then the move is finished.
7. When a new relative move is requested, go to step 3.

You may also have the P7 register in the cyclic write list, thereby enabling easy change of the relative distance to move.

6 MAC00-EP4 PROFINET® module

6.1 Introduction to PROFINET IO

Expansion module MAC00-EP4 front plate

6.1.1 Overview

PROFINET IO is a fieldbus protocol that enables communication between programmable controllers and distributed field devices in Ethernet networks.
PROFINET IO uses traditional Ethernet hardware and software to define a network that structures the task of exchanging data, alarms and diagnostics with Programmable Controllers and other automation controllers.
PROFINET IO can be thought of, as PROFIBUS on Ethernet. The protocol classifies devices into IO controllers, IO supervisors and IO devices, which have a specific collection of services.
PROFINET IO uses three different communication channels to exchange data.

- The standard UDP/IP and TCP/IP channel is used for parameterization and configuration of devices and for acyclic operations.
- The Real Time (RT) channel is used for cyclic data transfer and alarms.
- The third channel, Isochronous Real Time (IRT) channel, is used e.g. in some motion control applications (not implemented in JVL MAC00-EP4).

PROFINET IO devices are structured in slots, and sub-slots, which can contain modules and sub-modules correspondingly. Devices can have almost any number of slots and sub-slots and they can be virtual or real. Device specific data is represented in slot 0, module and sub-module specific data in subsequent slots and sub-slots. One of the benefits of PROFINET IO is the diagnostics and alarm mechanism. Every module and sub-module provides alarm data to the IO controller using the cyclic channel. Diagnostic data can be read non-cyclically from the device by using record data. Properties and services of a PROFINET IO device are described in a GSD file that is written in GSDML (General Station Description Markup Language). The GSD file describes the device specific modules and the method of assigning modules and sub-modules to predefined slots and sub-slots. There is no theoretical limit for the amount of connected nodes in PROFINET IO network, but in practice, the programmable controllers and number of available network addresses limits the size. The PROFINET IO protocol is specified in the IEC standards 61158 and 61784.
Further information can be obtained from www.PROFINET.com.

6.1 Introduction to PROFINET IO

6.1.2 Definitions and abbreviations

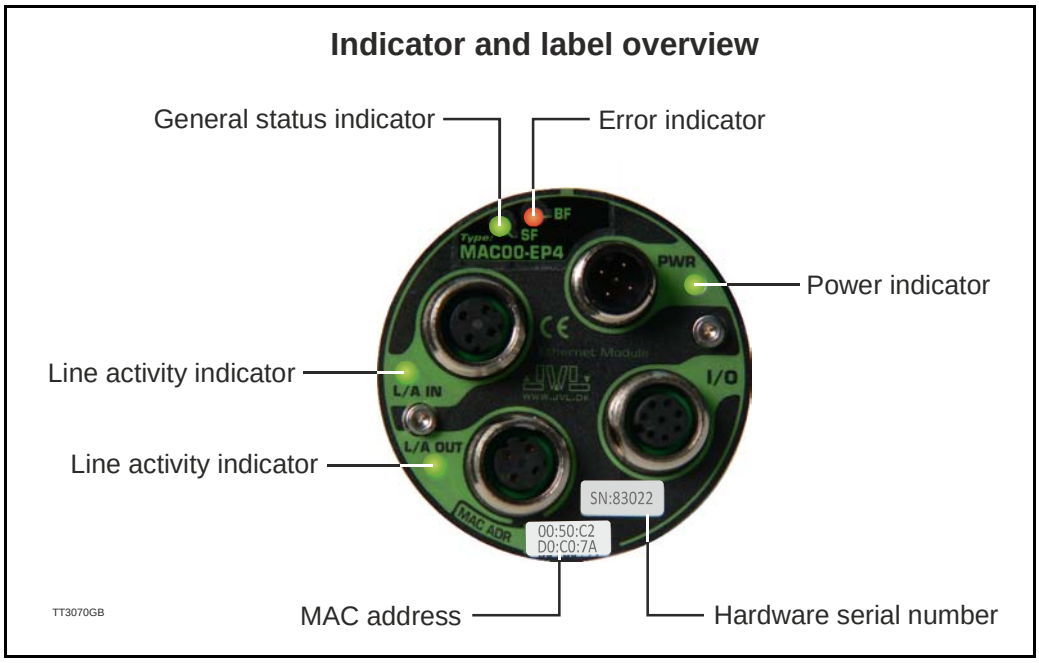
Following general used terms are useful to know before reading the following chapters.

100Base-Tx	100 MBit Ethernet on twisted pairs.
Acyclic communication	Communication in which messages are sent once per request.
Cyclic communication	Communication in which process data are sent cyclically at predefined intervals.
DAP	Device Access Point.
DCP	Discovery and Configuration Protocol.
GSD	General Station Description. Device description file in a specified form. Each device (active & passive stations) on PROFINET has to have its own GSD File. GSD files in PROFINET are written in GSDML.
GSDML	General Station Description Markup Language - is a XML based language used for the device description file.
IO-Controller	Control system with bus initiative. In PROFINET IO terminology, IO-controllers are also called master stations.
IOPS	IO Provider State (state of the provider of cyclic IO data).
IOCS	IO Consumer State (state of the consumer of cyclic IO data).
IP	Internet Protocol - IP address ~ the logical address of the device, which is user configurable.
MAC	Media Access Controller - MAC address ~ the hardware address of the device.
PZD	Process Data
TCP	Transfer Control Protocol (an IP based protocol used widely on the internet)
UDP	User Datagram Protocol (an IP based protocol used widely on the internet)

6.2 Commissioning

6.2.1 Indicator description

The LED's are used for indicating states and faults of module. There is one power LED, two link/activity LED's (one for each Ethernet connector), and 2 status LED's.



LED Text	Colour	Constant off	Constant on	Blinking	Flickering
L/A IN	Green	No valid Ethernet connection.	Ethernet is connected.	-	Activity on line
L/A OUT	Green	No valid Ethernet connection.	Ethernet is connected.	-	Activity on line
SF	Red	No System failures	System failures	-	-
BF	Red	No Bus failures	Bus failures	-	-
PWR	Green	Power is not applied.	Power is applied to both motor and module.	-	Power is applied to module but no communication with motor

Notes:
Blinking: Flashing with equal on and off periods of 200ms (2.5Hz). **Single flash** : Repeating on for 200ms and off for 1s. **Double flash** : Two flashes with a period of 200ms followed by 1s off period. **Triple flash** : Three flashes with a period of 200ms followed by 1s off period. **Flickering** : Rapid flashing with a period of approx. 50ms (10 Hz).

6.2.2 Mechanical installation

The network cables must be connected to the two M12 connectors (marked "L/A IN" and "L/A OUT") on the module. The cable from the IO CONTROLLER is connected to either of the two ports.
In the line topology, if there are more slave devices in the same line, the next slave device is connected to the second port.
If there is a redundant ring, the second port of the last slave device is connected to the second port of the IO CONTROLLER. See also figure in the introduction section.
Standard CAT 5 STP cables can be used. It is not recommended to use UTP cables in industrial environments, which is typically very noisy.

6.2.3 Network configuration

To enable communication through the Ethernet network, the module needs a valid IP address. This is done either by MacTalk, see the quick start guide or is done by DCP. In the PROFINET IO protocol, also a device name is required to identify the drive. IO-controllers and some configuration tools have a protocol called Discovery and Configuration Protocol (DCP) for assigning the IP address and the device name.

The IP address shown in MacTalk, is only a power on default. When a PLC is connected the actual used IP can be another one configured by the PLC.

PROFINET IO and DCP
When the module is initialized, the IP address is transferred to the PROFINET IO communication stack. If there is a need to change the IP address it should be done with a DCP tool (like Siemens Step7). If some of the other methods are used to change the IP address, the module must be restarted to enable any changes.

6.2.4 Configuring the system

After the MAC00-EPx module has been mechanically and electrically installed according to the instructions in previous chapters, and has been initialized by the drive, the master station must be prepared for communication with the module. Configuration of the master station requires a type definition (GSD) file. In PROFINET IO the GSD file is written in XML based language called GSDML. MAC00-EPx has a GSD file, which is available from www.jvl.com or your local JVL representative.
The filename is ***GSDML-V2.2-JVL-MAC00-EPx-yyyymmdd.xml***.
The GSD file describes vendor specific features of the module. Please refer to the master station documentation for more information on activating PROFINET IO devices with GSD file.

6.2.5 PROFINET IO in MAC00-EP4

MAC00-EP4 uses slots 0 and 1. Slot 0 does not have any sub-slots and the DAP module attached to it represents the device itself. Other functional modules and sub-modules, which are described in the GSDML file, can be assigned to slot 1 and its sub-slots:

- Slot 0 = Device access point (DAP)
- Slot 1, sub-slot 1 = Vendor object
- Slot 1, sub-slot 1 = Acyclic parameter access

The MAC00-EP4 module provides the following services:

- Cyclic messaging
- Acyclic parameter access mechanism
- Identification & Maintenance functions (I&M)

6.2.6 Dynamic IP and naming

With DCP (Discovery and Configuration Protocol) the IP address and 'Name of station' in the device can be changed on the fly, by the PLC. Therefore the IP address shown in MacTalk is only the power up default, and may not be the actual used IP address after the PLC has established communication.

If checking the "Power up with blank name of station" in MacTalk and save the configuration in flash, then the MAC00-EPx will always start up without a station name. This enables the possibility of having new devices on stock, and if needed exchange them in the machine without any setup, as the PLC can be programmed to automatically assigning the correct name, when it finds a device without name.

6.2.7 Quick start guide

This section describes the steps to configure the Siemens ET200S PLC and TIA Portal v11 software, so it can be used to control the drive.

Set IP address

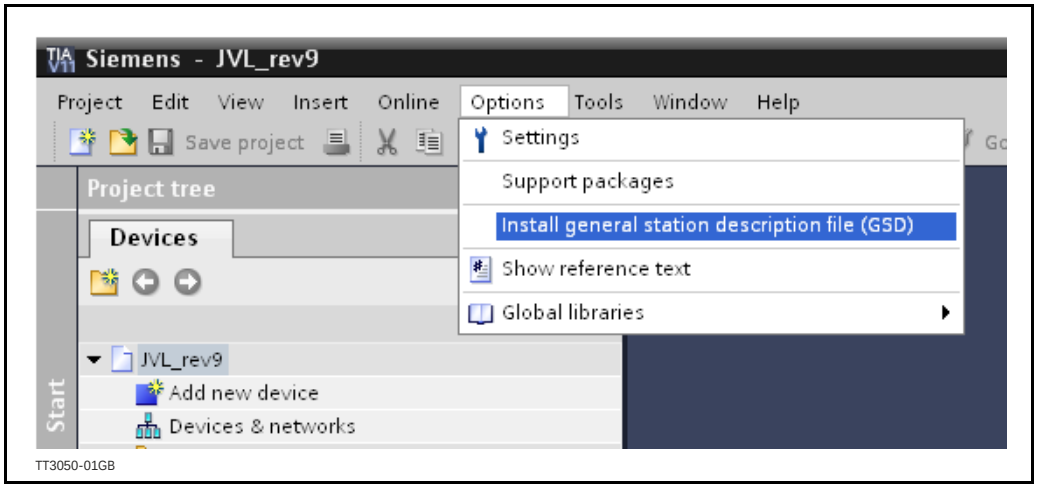
1. Connect the RS232 communication cable.
2. Apply power to the motor, and make sure the PWR LED is lit.
3. Open MacTalk and select the "MAC00-EP (PROFINET)" tab.
4. Change the IP address, to one suitable for the network.
5. Press "Apply and save".

Installation

6. Connect an Ethernet RJ45-M12 cable to one of the interfaces on the ET200S and to L/A IN or L/A OUT on MAC00-EPx.
7. Connect power to the ET200S, and Ethernet patch cable from the PC with Siemens TIA Portal v11 installed to the ET200S PLC.
8. Make sure power is applied to all devices.

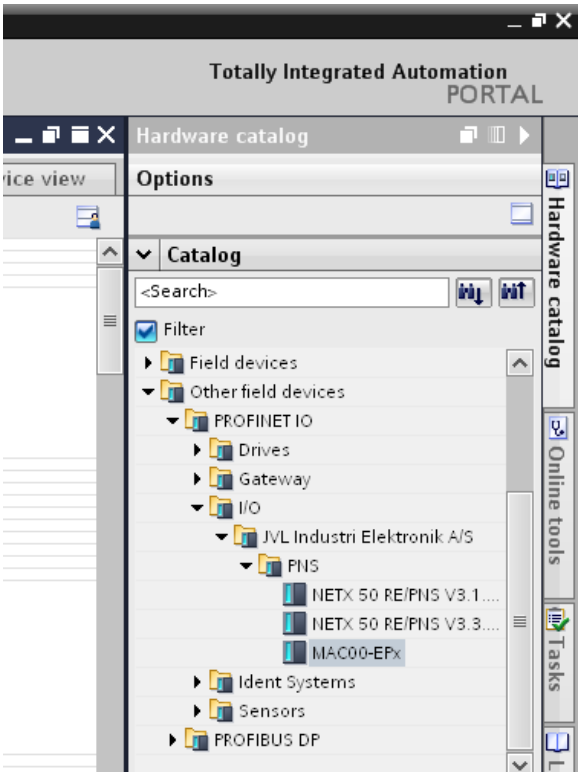
Add the GSD file (contains info on the capabilities of the device)

9. In the Options menu of TIA Portal V11, select Install general station description file (GSD).
10. In the "Install general station description file" window find and select the "GSDML-V2.2-JVL-MAC00-EPX-yyyymmdd.xml" file, and click Install.
11. Follow the on screen instructions.

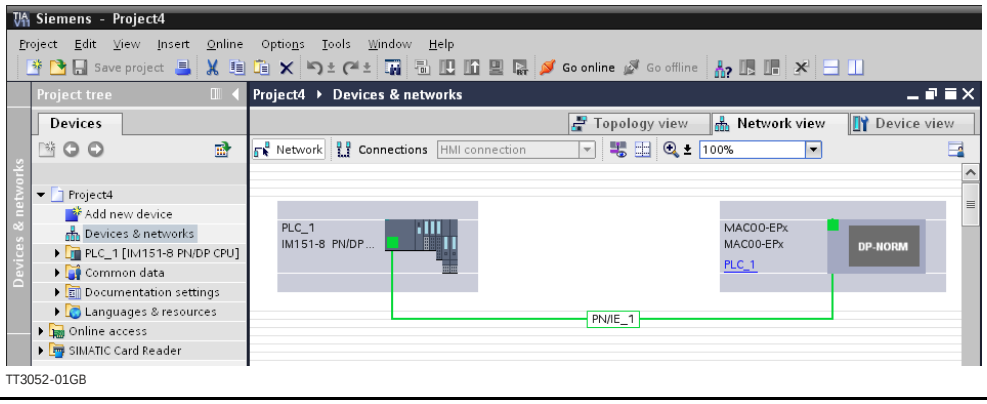


PLC configuration

12. Create a new project in TIA Portal v11 for your PLC, or open an existing project. See Siemens documentation for more information.
13. In the **Hardware catalog** under **Other field devices** / **PROFINET I/O** / **I/O** / **JVL Industri Elektronik A/S** / **PNS** should **MAC00-EPx** reside. See figure at next page.

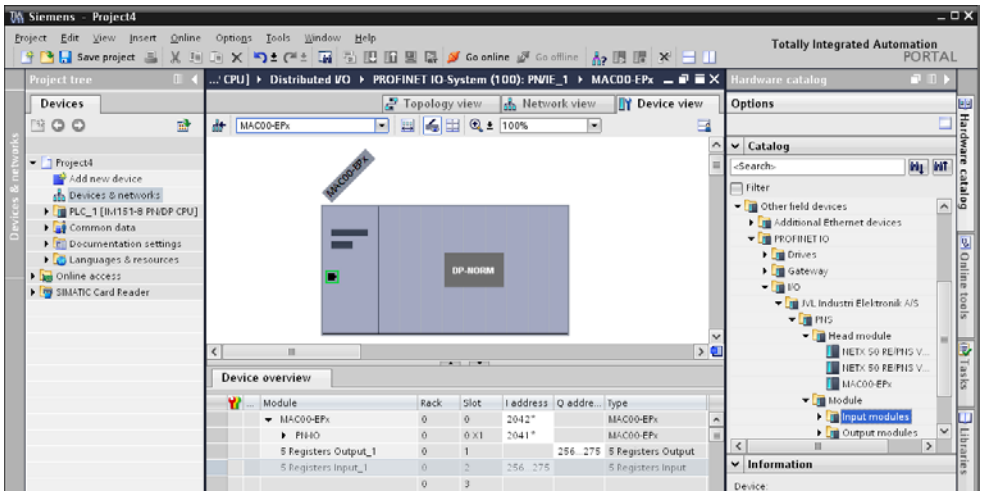


14. Drag and drop the MAC00-EPx to the Network view.
15. Also add your PLC to the Network view (see also Siemens documentation for further info).
16. If using external switches then these must be added too (see Siemens documentation for further info). If not continue to the next step.
17. Connect the two devices, by dragging a line between the small green boxes in each device, and it should now look like below.



Associating with the cyclic data

18. Drag the "**8 registers input**" and "**8 registers output**" from the **hardware catalog** under **Other field devices** / **PROFINET I/O** / **I/O** / **JVL Industri Elektronik A/S** / **PNS** / **Module** / **Input modules** and **Output modules**, and drop them in the **Device overview** of the MAC00-EPx. See illustration next page.



TT3053-01GB

19. It should now be possible to make a PLC application using cyclic communication to the 5 registers input and output (see section 6.3.1 for setting up those with Mactalk).

    There is also an example on the web page www.jvl.dk in the download section, named 'JVL_PN_ex1.zip' which can be downloaded and unzipped. This example is made for MAC140, but can easily be changed to work with MAC400-MAC3000.

6.3 PROFINET objects

6.3.1 Process data

Process Data (PZD) are used for cyclic transfer of time-critical process data between master and slaves, such as position, velocity, torque etc. Transmit PZD are used to transfer data from the slave to the master and receive PZD to transfer data from the master to the slave.
The JVL process data is fully user configurable. It is possibly to set up eight, 32 bit registers in each direction. The setup is done with MacTalk or via parameter object 0x11 sub-index 16-31. It requires a save in flash and a power cycle before the new configuration are used. If the configuration of the PZD, is not altered by the user, the JVL PROFINET module uses the default mapping shown in the tables below. It is mandatory to have the error/status register (register 35) as one of the slave to master registers. If not the motor will overrule the configuration and place register 35 anyway.

If module registers is placed in cyclic R/W, then the register number has to be calculated as follows:

Register number = 65536 x sub index.
Example: module command (sub-index 15) = 65536 x 15 = register **983040**

When module registers (register numbers above 65535) are chosen, they **have** to be placed **after** the motor registers in the list of cyclic registers.

NB! If an index is set to zero (No selection) then the following indexes is discarded. Thereby computing resources in the drive are released, which makes much faster cycle times possibly. Please see next paragraph.

Default registers in transmit PZD (Slave ' Master / Read words in MacTalk).

Object index	Motor Register no.	Motor register short	Motor register description
0	2	MODE_REG	Operating mode
1	10	P_IST	Actual position
2	12	V_IST	Actual velocity
3	169	VF_OUT	Actual torque
4	35	ERR_STAT	Status bits
5	-	-	-
6	-	-	-
7	-	-	-

Default registers in receive PZD (Master → Slave / Write words in MacTalk).

Object index	Motor Register no.	Motor register short	Motor register description
0	2	MODE_REG	Operating mode
1	3	P_SOLL	Target position
2	5	V_SOLL	Maximum velocity
3	7	T_SOLL	Maximum torque
4	-	-	-
5	-	-	-
6	-	-	-
7	-	-	-

Please notice: Even though all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

6.3.2 Minimum cycle time

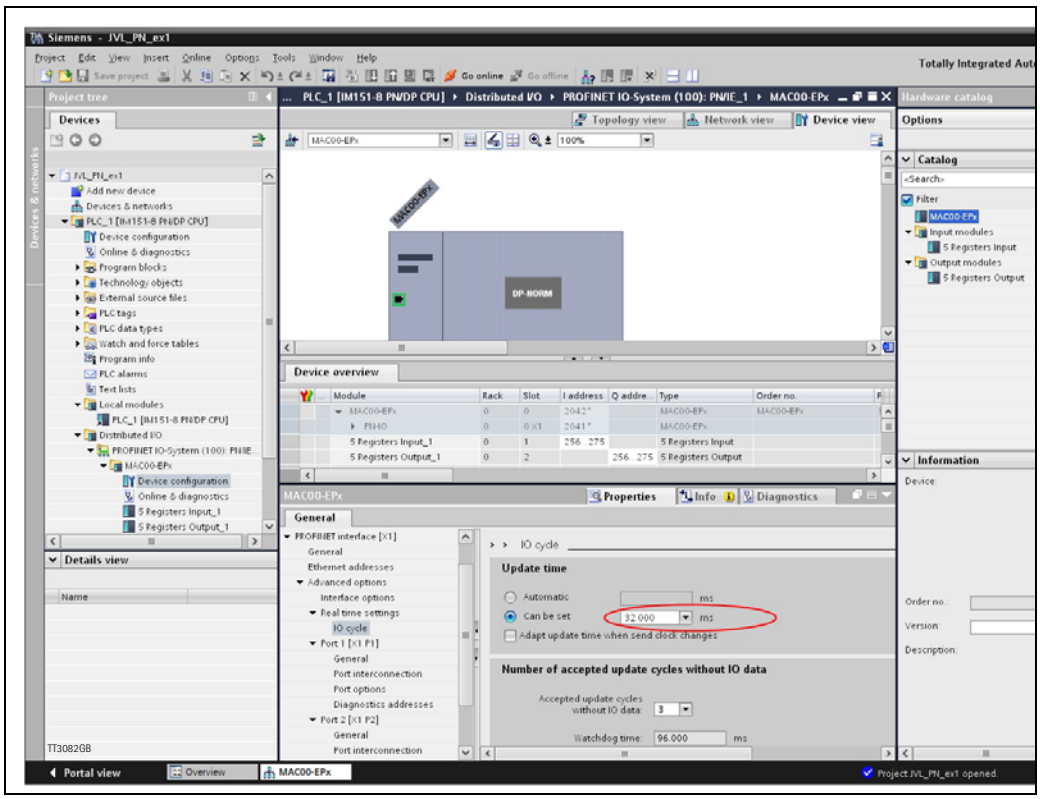
The minimum cycle time is the minimum amount of time between each cyclic request on the Ethernet. If operating with values lower than those listed, data loss will occur.

No. of motor registers transmitted in each direction	Motor series	
	MAC050 - MAC141	MAC400 and MAC3000
1/1	4ms *	1ms *
2/2	8ms *	1ms *
3/3	12ms *	1ms *
4/4	16ms *	1ms *
5/5	20ms *	1ms *
6/6	24ms *	1ms *
7/7	28ms *	1ms *
8/8	32ms *	1ms *

* The minimum cycle times, is only valid if not sending any acyclic requests while in any operating mode. MODULE registers can be appended as the last registers in the list, at no extra timing cost. Motor register 35 shall be in the cyclic read list, as it is also used internally.

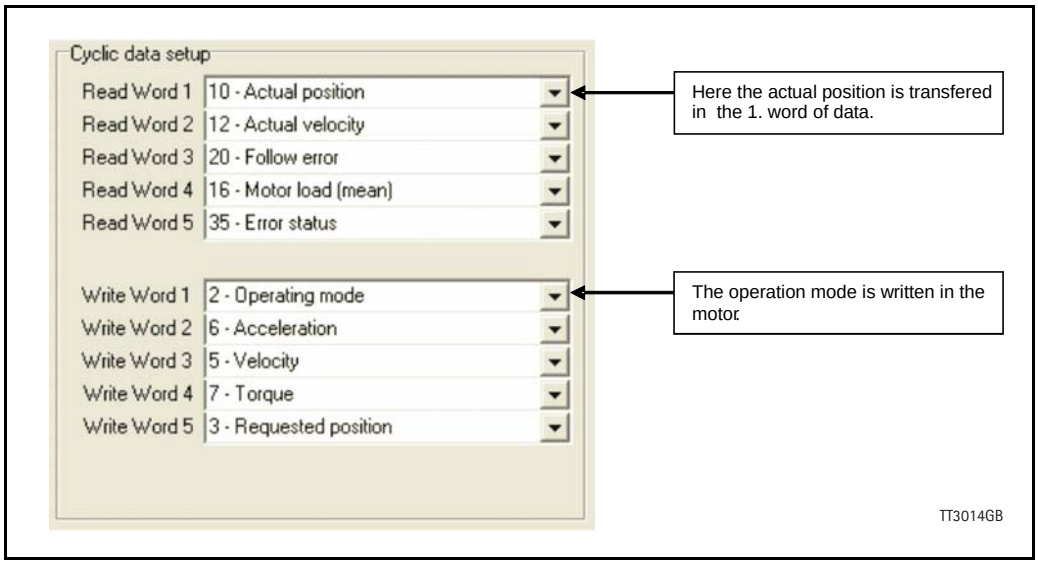
Changing cycle time in TIA Portal V11

In the TIA Portal V11 the cycle time is set up in the properties of each device, under Real time settings / IO Cycle. Please see the picture below. It is done in a similar way in Step7, but this is not shown.



Accessing process data

The PZD is done by setting up the motor registers you want to use with MacTalk or with acyclic parameter access to object 0x11 subindex 16-31. In MacTalk the process data is configured on the PROFINET tab, see below. After change of the registers, remember to press the Apply and save button.



6.3 PROFINET objects

6.3.3 Parameter objects.

The parameter objects provide access to all module registers, and all motor registers, as well as a module command object. The objects in the list can be accessed with acyclic services, *Accessing parameter objects, page 131*.

	Object (hex)	Sub Object	Type	Read only	Default	Description
Module command	0x10	0	UNSIGNED32			Module command object. See possible commands below.
Module parameters	0x11	1	UNSIGNED32	X		High 16 bit of MAC address (placed in low 16 bit of word)
		2	UNSIGNED32	X		Low 32 bit of MAC address
		3	UNSIGNED32		-	IP address / Node ID (The least significant 8 bits is node ID)
		4	UNSIGNED32	X	-	Net mask
		5	UNSIGNED32	X	-	Gateway
		6	UNSIGNED32		0x0	Setup bits
		7	UNSIGNED32		0	Digital outputs on module
		8-14	UNSIGNED32		-	Reserved for future use
		15	UNSIGNED32		-	Command register
		16	UNSIGNED32		2	Register no. to place in TxPDO 21, position 1.
		17	UNSIGNED32		10	Register no. to place in TxPDO 21, position 2.
		18	UNSIGNED32		12	Register no. to place in TxPDO 21, position 3.
		19	UNSIGNED32		169	Register no. to place in TxPDO 21, position 4.
		20	UNSIGNED32		35	Register no. to place in TxPDO 21, position 5.
		21	UNSIGNED32		-	Reserved for future use
		22	UNSIGNED32		-	Reserved for future use
		23	UNSIGNED32		-	Reserved for future use
		24	UNSIGNED32		2	Register no. to place in RxPDO 21, position 1.
		25	UNSIGNED32		3	Register no. to place in RxPDO 21, position 2.
		26	UNSIGNED32		5	Register no. to place in RxPDO 21, position 3.
		27	UNSIGNED32		7	Register no. to place in RxPDO 21, position 4.
		28	UNSIGNED32		0	Register no. to place in RxPDO 21, position 5.
		29	UNSIGNED32		-	Reserved for future use
		30	UNSIGNED32		-	Reserved for future use
		31	UNSIGNED32		-	Reserved for future use
		32	UNSIGNED32	X	-	Module serial no.
		33	UNSIGNED32	X	-	Module hardware version
		34	UNSIGNED32	X	-	Module software version
		35	UNSIGNED32	X	-	No. of internal motor communication timeouts
		36	UNSIGNED32	X	-	No. of retry frames to motor
		37	UNSIGNED32	X	-	No. of discarded frames to the motor
		38	UNSIGNED32	X	-	Total no. of frames to motor
		39-46	UNSIGNED32	X	-	Reserved for future use
		47	UNSIGNED32	X	-	Digital inputs on module
		48	UNSIGNED32	X	-	Status bits
		49-63				Reserved for future use
Motor parameters	0x12	0-255	Register dependant	X	254	Access to the motor parameter n (register)

Note: Module parameters are not automatically saved to permanent memory after a change. The parameters can be saved permanently by applying a "Save parameters to flash" command afterwards.

6.3 PROFINET objects

Object 0x10 - Subindex 0
This object is used for sending commands to the module and is write only. The possible commands are listed in the table below.

Command no.			Command description	
	Hex	**Dec**	**MAC050 - MAC141**	**MAC400 – MAC3000**
Module only commands				
	0x 0000 0000	0	No operation	< Same as
	0x 0000 0001	1	Reset the module	< Same as
	0x 0000 0010	16	Save module parameters to flash	< Same as
Synchronized commands				
	0x 0000 0101	257	Simultaneous reset of the motor and the module	< Same as
	0x 0000 0110	272	Save the motor parameters in flash memory, and do a re-sync. of internal communication afterwards.	< Same as
Motor only normal commands *(via module cmd register)*				
	0x 8000 0001	2147483649	Reset motor *(not recommended, use synchronized version instead)*.	< Same as
	0x 8000 0002	2147483650	Save motor parameters in flash and reset motor *(not recommended, use synchronized version instead)*.	< Same as
Motor only FastMac commands *(via module cmd register)*				
	0x8000 00E0	2147483872	No operation	< Same as
	0x8000 00E1	2147483873	Reset error *(Clear error bits in motor register 35)*	< Same as
	0x8000 00E2	2147483874	P_SOLL = 0	< Same as
	0x8000 00E3	2147483875	P_IST = 0	< Same as
	0x8000 00E4	2147483876	P_FNC = 0	< Same as
	0x8000 00E5	2147483877	V_SOLL = 0	< Same as
	0x8000 00E6	2147483878	T_SOLL = 0	< Same as
	0x8000 00E7	2147483879	Reset IN_POS, AC C,DEC	< Same as
	0x8000 00E8	2147483880	P_FNC = (FLWERR - P7) * 16	< Same as
	0x8000 00E9	2147483881	P_FNC = (FLWERR - P8) * 16	< Same as
	0x8000 00EA	2147483882	Reserved	< Same as
	0x8000 00EB	2147483883	Reserved	< Same as
	0x8000 00EC	2147483884	Activate P1,V1,A1,T1,L1,Z1	< Same as
	0x8000 00ED	2147483885	Activate P2,V2,A2,T2,L2,Z2	< Same as
	0x8000 00EE	2147483886	Activate P3,V3,A3,T3,L3,Z3	< Same as
	0x8000 00EF	2147483887	Activate P4,V4,A4,T4,L4,Z4	< Same as
	0x8000 00F0	2147483888	Start search zero	< Same as
	0x8000 00F1	2147483889	P_SOLL = P_IST + P7;	P_SOLL = P_IST + P7 – FLWERR;
	0x8000 00F2	2147483890	P_SOLL = P_IST + P8;	P_SOLL = P_IST + P8 – FLWERR;
	0x8000 00F3	2147483891	Reserved	< Same as
	0x8000 00F4	2147483892	Select absolute position mode	< Same as
	0x8000 00F5	2147483893	Select relative position mode using P_SOLL	< Same as
	0x8000 00F6	2147483894	Select relative position mode using P_FNC	< Same as
	0x8000 00F7	2147483895	Synchronize position manually using absolute new values. P_IST = P_NEW; P_SOLL = P_NEW; P_FUNC = P_NEW * 16;	Synchronize position manually using absolute new values. P_IST = P_NEW; P_SOLL = P_NEW; P_FNC = (P_NEW + FLWERR)*16;
	0x8000 00F8	2147483896	Synchronize position manually using relative new values. (basically offset the position range with the value of P_NEW). P_IST = P_IST + P_NEW; P_SOLL = P_SOLL + P_NEW; P_FUNC = P_FUNC + (P_NEW * 16);	< Same as
	0x8000 00F9	2147483897	No operation	< Same as
	0x8000 00FA	2147483898	No operation	< Same as
	0x8000 00FB	2147483899	No operation	< Same as
	0x8000 00FC	2147483900	No operation	< Same as
	0x8000 00FD	2147483901	Reserved	< Same as
	0x8000 00FE	2147483902	Reserved	< Same as
	0x8000 00FF	2147483903	Reserved	< Same as

Object 0x11
The module registers is mapped to object 0x11. The subindex 3-31 is R/W, the rest is read only.

Object 0x11 – Subindex 1 MAC address MSB.
The 2 most significant bytes of module MAC Address, placed.

Bit	16-31	0-15
Output	Reserved	16 Most significant bits of MAC address.

Object 0x11 – Subindex 2 MAC address LSB.
The 4 least significant bytes of module MAC Address.

Bit	0-31
Output	32 Least significant bits of MAC address.

Object 0x11 – Subindex 3 IP address.
This is the power on default IP address of the device. Please note that the actual used IP address can be changed by DCP "on the fly" without being reflected in this register.

Bit	0-31
Output	IP address.

Object 0x11 – Subindex 4 Netmask.
This is the netmask of the device. The netmask is fixed.

Bit	0-31
I/O	Subnet mask

Object 0x11 – Subindex 5 Gateway.
This is the gateway address of the device. The gateway address is also fixed.

Bit	0-31
I/O	Gateway

Object 0x11 – Subindex 6 Setup bits
This register is used to setup how the module should react on different events.

Bit	3-31	2	1	0
Output	Reserved	1: Erase "name of station" at power-up	1: Disable Ethernet handling.	0 : Ethernet error handling = motor set passive mode 1 : Ethernet error handling = motor set velocity to 0

Object 0x11 – Subindex 7 Digital outputs on module
With this object the digital outputs can be controlled.
The value written to this object is directly shown on the digital outputs.

Bit	2-31	1	0
Output	Reserved	OUT2* (O2)	OUT1* (O1)

* The availability of the inputs depends on the actual version of the module used. Example MAC00-EP4 only support Output 1 (OUT1).

Object 0x11 – Subindex 15 Command register
Analogue to writing to object 0x10. But this can be mapped in the PZD if desired.

Object 0x11 – Subindex 16-23 Register no. to place in Transmit PZD
These registers contain the numbers that define the registers which are in the transmit PZD. That is the register's, which is transmitted from slave to master cyclically. If some of these registers are changed, it is necessary to issue a "save in flash" command and to reboot the device before the changes take effect.

Object 0x11 – Subindex 24-31 Register no. to place in receive PZD
These registers contain the numbers that define the registers which are in the receive PZD. That is the register's, which is transmitted from master to slave cyclically. If some of these registers are changed, it is necessary to issue a "save in flash" command and to reboot the device before the changes take effect.

Object 0x11 – Subindex 32-38
These registers contain HW, SW and communication information of the module.

Object 0x11 – Subindex 47 Digital inputs on module
With this object the status of the 4 digital inputs can be read.

Bit	4-31	3	2	1	0
Input	Reserved	IN4*	IN3*	IN2*	IN1*

* The availability of the inputs depends on the actual version of the module used. Example MAC00-EP4 only support Input 1 (IN1).

Object 0x11 – Subindex 48 Status bits
This register is used for miscellaneous information about the module.

Bit	8-31	7	6	5	4	3	2	1	0
Output	Reserved	Reserved	Reserved	Reserved	No communication with motor	Temperature error	Voltage error	Current error	Generic error

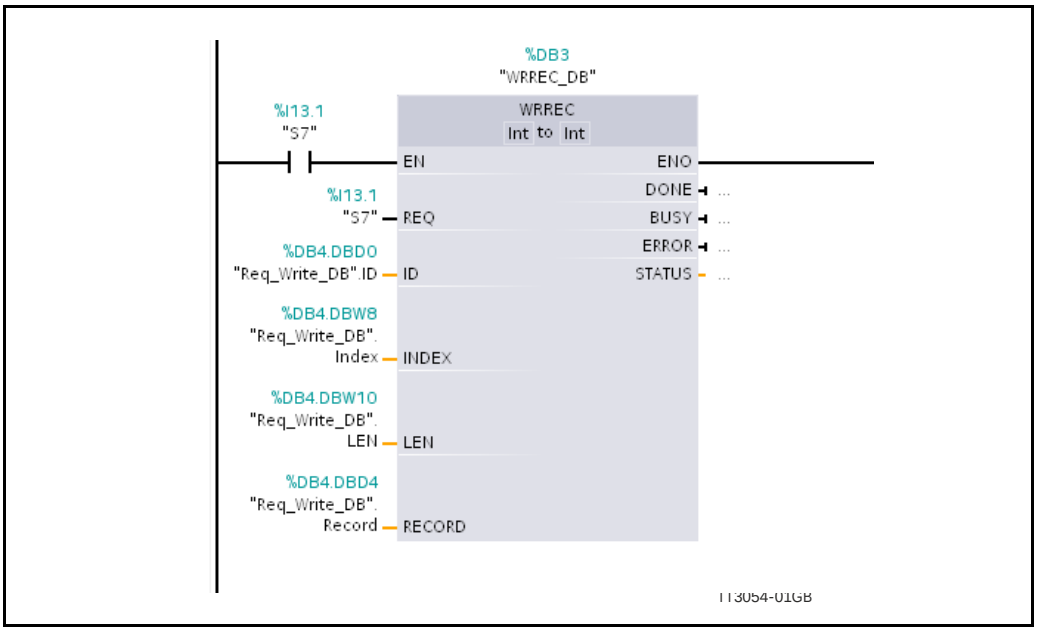
Object 0x12
Object 0x12 are for acyclic view or change of motor registers.
Please find a complete list of register descriptions in the appendex.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

6.3.4 Accessing parameter objects

Parameter objects are accessible by use of acyclic data. In Siemens Step 7, this is done with the Special Function Blocks SFB52 and SFB 53.

Write parameter

Write to parameters is done with the SFB53, as shown below.



The data block must be setup prior to use, in this example "Req_Write_DB".

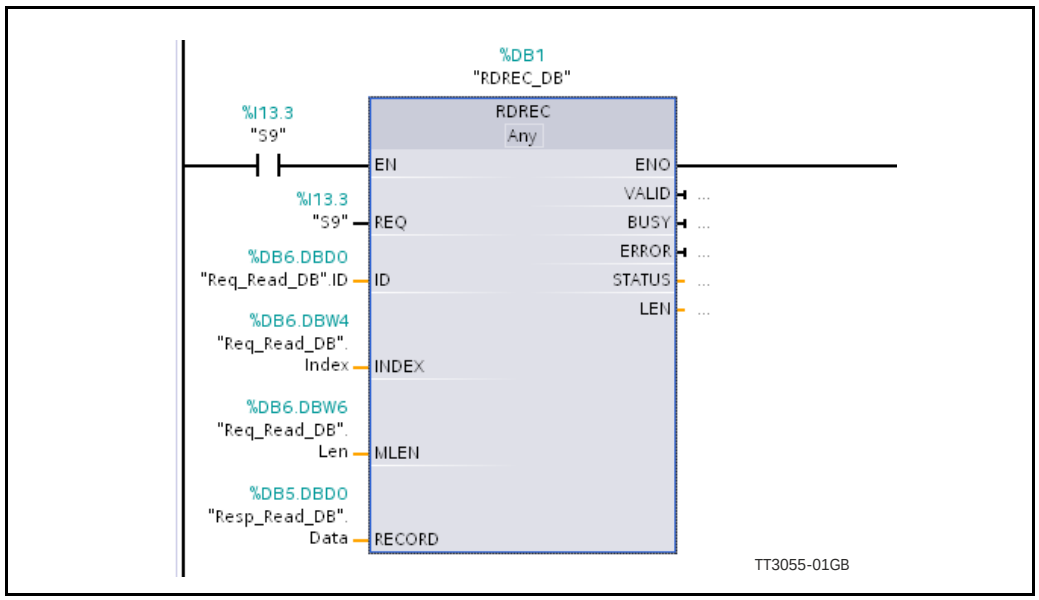
Name	Description	Example
ID	ID of device	2042
Index	Object and subobject to write to High byte = Object, Low byte = Subobject (parameter/register no.)	0x1231 (Object 0x12, parameter 0x31)
LEN	Length of data	4 (always 4 byte = 32 bit)
Record	32 bit data to write	0xFEDCBA98

Example:

Write 0xFEDCBA98 to object 0x12 subobject 0x31 (= motor register no. 49), in JVL device with ID 2042.

Please notice: Even though all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

Read parameter
Read of parameters is done with SFB52, as shown below.



The data block must be setup prior to use, in this example "Req_Read_DB", and the 32 bit result will be in "Resp_Read_DB.Data".

Name	Description	Example
ID	ID of device	2042
Index	Object and sub-object to read from High byte = Object, Low byte = Subobject (register no.)	0x1122
LEN	Length of data	4 (always 4 byte = 32 bit)

Example:
Read from object 0x11 subobject 0x22 (= module parameter no. 34), in JVL device with ID 2042.

Please notice: Even though all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

6.4 Ethernet switch

6.4.1 Selecting an Ethernet Switch

Depending on the network topology and size a suitable switch can be used. Also if multiple separated networks need to be connected a switch is used.
Depending on the actual size of the network different requirements needs to be met.

It is absolutely mandatory that every switch device or other device acting as a switch complies with the Profinet RT protocol (LLDP and PN_PTCP frames must be recognized).

Otherwise the net might very well get congested; because non-Profinet conforming switches will broadcast messages not intended for broadcast.

Besides the Port mirroring function for network analyzing and troubleshooting purposes, can be advantages. This feature makes it possible to route traffic out on a separate port connected to a network analyser for debugging purposes and general performance monitoring.

The JVL MAC00-EPx module has build in 2 port switch useful if a limited amount of motors is connected in a daisy chaining topology. This switch is Profinet-IRT capable and will of cause obey the above mentioned demands.

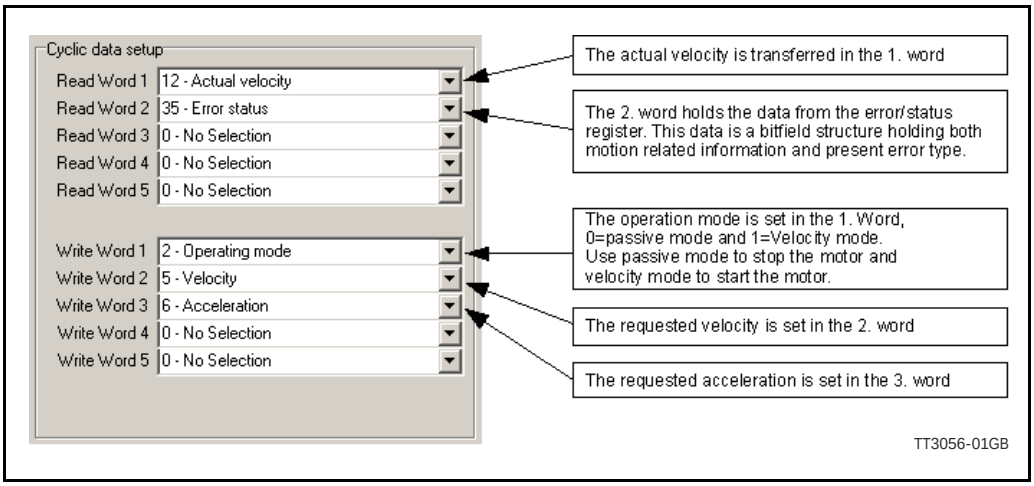
6.5 Examples

6.5.1 Running Velocity control

To use the JVL motor in velocity -mode the following registers basically is of interest.

1. "Mode" - mode register register 2
2. "V_SOLL" - velocity register 5
3. "A_SOLL" - acceleration register 6
4. "Error/Status" - register 35

So, to control these registers the assembly object needs to be configured.
From MacTalk the setup is configured as this.



With the settings illustrated above we initiate the velocity mode by writing 0x1 to the first word-value, this is velocity mode.

Since different PLC's have different methods of implementation the basic steps is described in the following.

1. **Set the needed velocity**.
   V_SOLL = V x 2.77 [rpm].
   Example: We need the motor to run with a constant speed of 1200 RPM.
   So, V_SOLL = 1200/2,77 = 433 cnt/smp
2. **Set the needed acceleration.**
   A_SOLL = A x 271 [RPM/s2].
   Example: We need the motor to accelerate with 100000 RPM/s2 so, A_SOLL = 100000/271 = 369 cnt/smp2
3. **Now set the motor in velocity mode and thereby activate the motor.**
   Example: The motor needs to be activated by setting it into velocity mode, so we need to set the mode register to the value 1. Mode = 1 which is velocity mode, now the motor will use the acceleration and the velocity just configured.

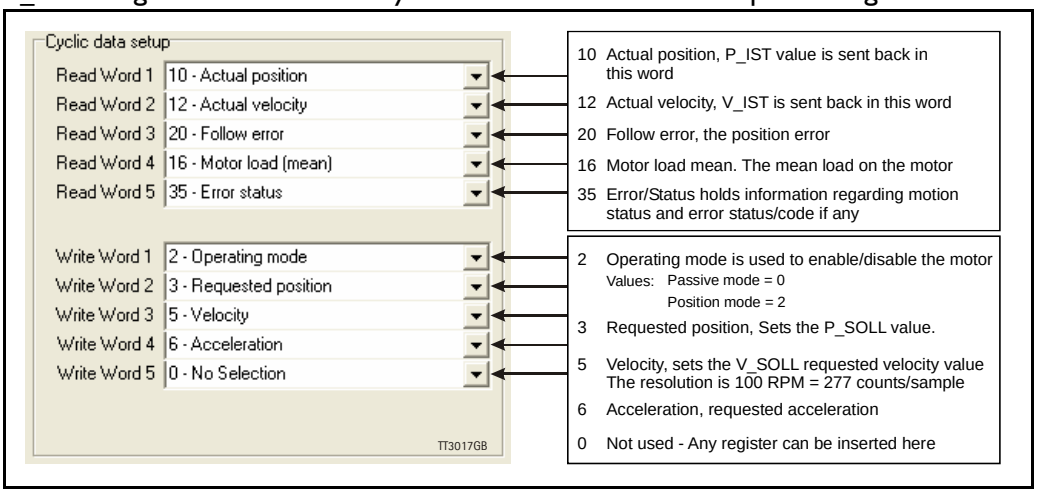
Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

6.5.2 Running Position control

Running the motor in position control requires that the mode register is set for position control. The following registers is of particular interest when position mode is used.

1. "Actual position" -P_IST, register 10
2. "Actual velocity" -V_IST, register 12
3. "Follow error" - The actual position error, register 20
4. "Motor load mean" - average motor load, register 16
5. "Error/Status" -register 35
6. "Requested position" -P_SOLL, register 3
7. "Requested velocity" -V_SOLL, register 5
8. "Requested acceleration" -A_SOLL, register 6

In this mode the position is controlled by applying a requested position to the "P_SOLL" -register and the actual position is monitored in the "P_IST" register. The V_SOLL and A_SOLL registers sets the velocity and acceleration used when positioning occurs.



6.5.3 General considerations

The register 35 in the motor holds information on the actual error/status. So it is crucial that this register is configured in the cyclic data and thereby obtained and monitored in the Master. In case of an error situation the motor will stop and the cause will be present in the register 35 and hence in the I/O -data.
This register also holds information on the motion status such as:

- In position, bit 4
- Accelerating, bit 5
- Decelerating, bit 6

Please find a complete list of register descriptions in the appendix.
Motor registers MAC050 - 141, page 171 and *Motor registers MAC400 - 3000, page 180*

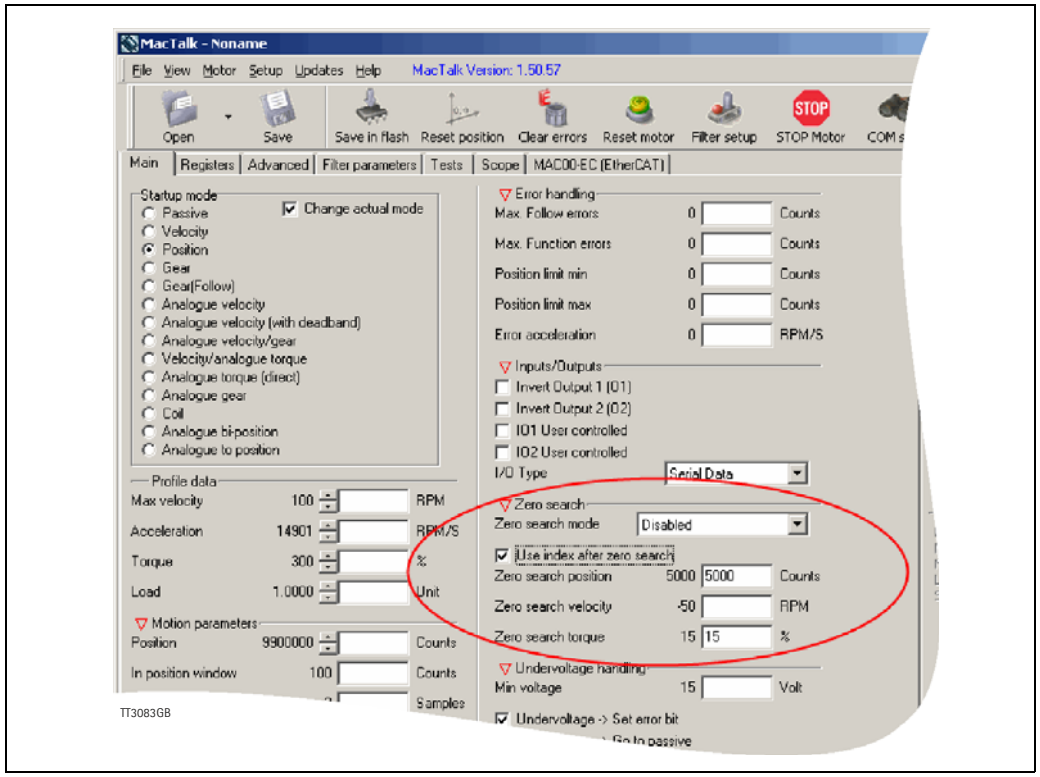
The JVL motor is basically put into a working mode and into a passive mode where the motor axle is de-energized, by setting register 2 into either 0 = "passive mode" or into one of the supported modes.

Example.
1= "Velocity mode" / 2= "Position mode" / etc.

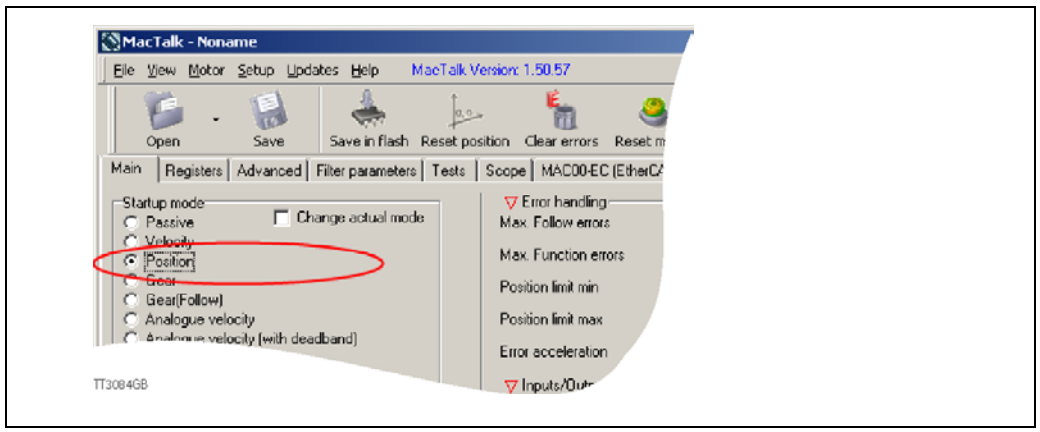
So in order to Stop or Start the motor this register can be supported in the I/O data or by sending an acyclic message.

6.5.4 Homing using only cyclic I/O (JVL profile).

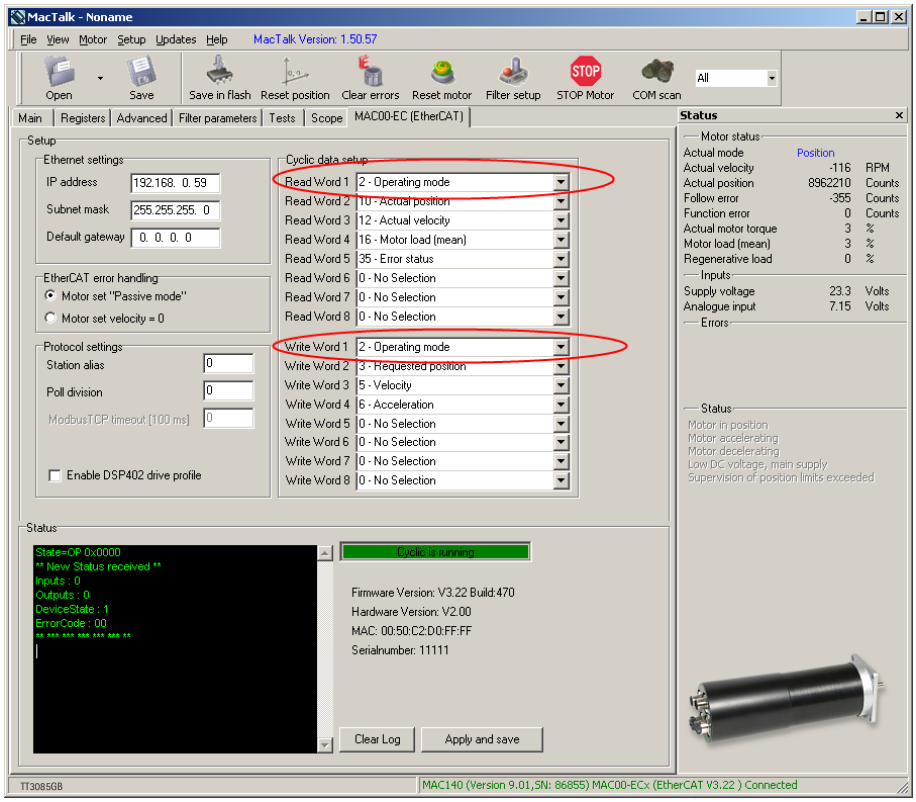
When doing a homing (Zero search), with only cyclic I/O, some preconditions have to be met:
Zero search position, zero search velocity and zero search torque (torque only for MAC motors) has to be set in MacTalk in the "Main" tab, and saved in flash in the motor once and for all.



Startup mode should be set to position, for the motor to stay in position after the homing sequence. And this setting should also be saved in flash.



Register 2 (Operating mode) has to be present in BOTH the cyclic read words and cyclic write words.



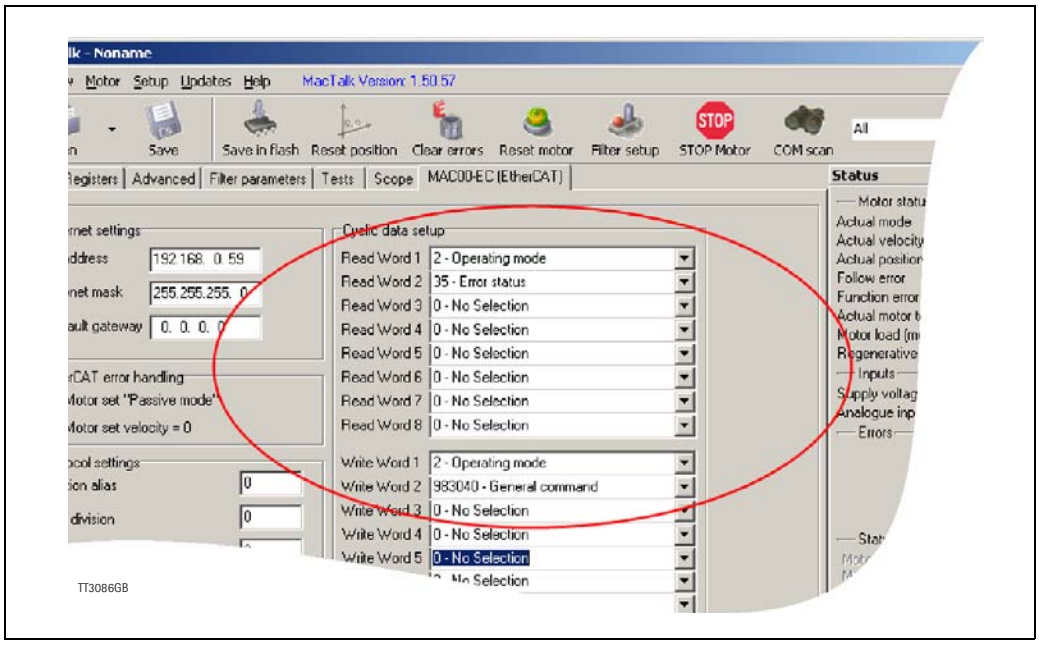
Procedure in the PLC:

- Treat the transmitted Register 2 as "Requested_Mode" and the received register 2 as "Actual_Mode".
- When homing is wanted, set the "Requested_Mode" to one of the values 12, 13 or 14 depending of the requested homing mode (12 = Torque based zero search mode (only MAC motors). 13 = Forward/only zero search mode. 14 = Forward+backward zero search mode (only MAC motors) .). For a comprehensive description of the homing modes, refer to the general MAC motor manual - LB0047-xxGB.
- Observe that the "Actual_Mode" is changing to the homing mode. Now the module is blocking cyclic writes TO the motor. Cyclic reads is still active.
- Wait for register 35 "Error status" bit 4 to be active =IN_POSITION. (Indicates that homing is finished).
- Then change "Requested_Mode" to whatever needed. The blocking of cyclic writes to the motor is then released by the module.

6.5.5 Relative positioning.

There are a number of ways to do relative positioning, but the one explained here is very simple, and can be used with a constant distance, or exchangeable distance, to move every time it is requested.

Preconditions:
Place the module command register (register 983040 in MacTalk) in the cyclic write list. The cyclic setup, could for example look like this:



Procedure in the PLC:
1. Set up register P7 in motor to requested relative offset.
2. Make sure one net cycle has passed, so P7 resides in the motor.
3. Issue command 0x800000F1 (0x80000071 if MIS34x) in module command register (register 983040 in MacTalk).
4. Make sure one net cycle has passed, so command is interpreted by the motor.
5. Set module command register to zero. This will prepare the Ethernet module for new commands.
6. If needed then monitor register 35 (Error status): When bit 4 is set (in position), then the move is finished.
7. When a new relative move is requested, go to step 3.

You may also have the P7 register in the cyclic write list, thereby enabling easy change of the relative distance to move.

7 MAC00-EM4 Modbus TCP/IP® module

7.1 Introduction to Modbus TCP/IP®

Expansion module MAC00-EM4 front plate

7.1.1 Introduction.

Modbus TCP/IP or Modbus TCP — is a Modbus variant used for communications over TCP/IP networks, connecting over port 502. It is basically a Modbus RTU without a checksum calculation as lower layers already provide checksum protection. It is protocol based on the standard TCP/IP protocols so it is applicable anywhere there is standard Ethernet available as it have no special requirements regarding the Ethernet hardware, opposite some of the other industrial Ethernet protocols. Further information about Modbus TCP is available from the Modbus Organization www.modbus.org.

7.1 Introduction to Modbus TCP/IP®

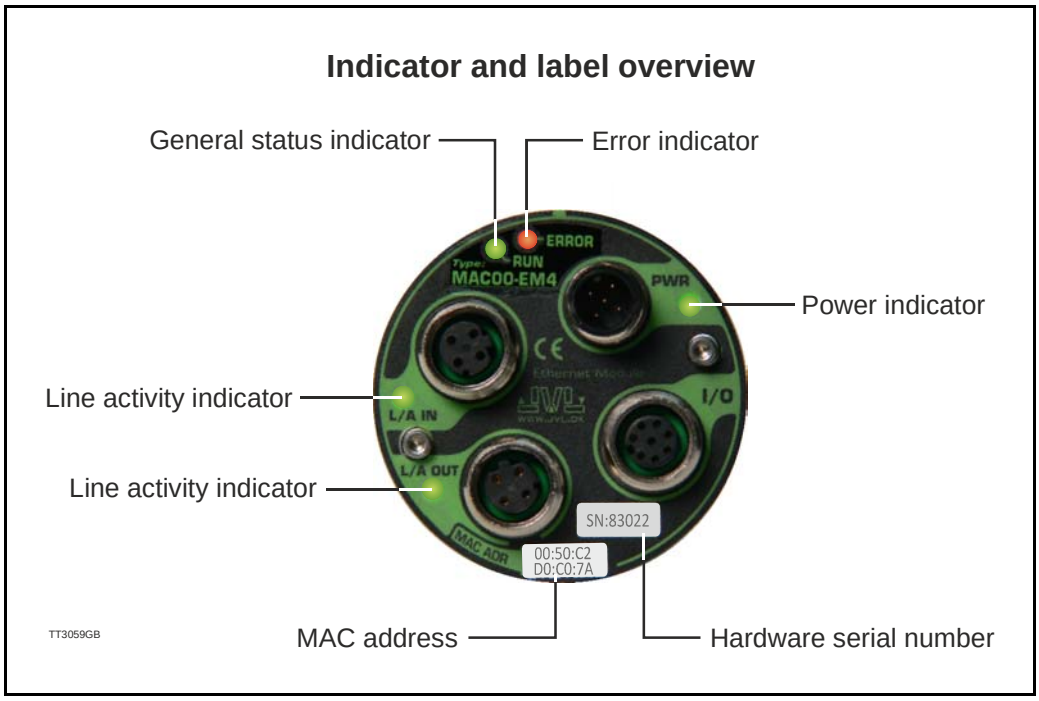
7.1.2 Abbreviations

The below general used terms are useful to know before reading the following chapters.

100Base-Tx	100 MBit Ethernet on twisted pairs
IP	Internet Protocol - IP address ~ the logical address of the device which is user configurable.
MAC	Media Access Controller - MAC address ~ the hardware address of the device.
MacTalk	A windows PC based program supplied from JVL. This is an overall program to install, adjust and monitor the function of the motor and a module installed in the motor.
TCP	Transfer Control Protocol (an IP based protocol used widely on the internet)
UDP	User Datagram (an IP based protocol used widely on the internet)

7.2.1 Indicator LED's - description.

The LED's are used for indicating states and faults of module. There is one power LED, two link/activity LED's (one for each Ethernet connector) and 2 status LED's.



LED indicator descriptions

LED Text	Off	Red	Orange	Green	Flickering Green
L/A IN	No valid Ethernet connection	-	-	Ethernet is connected	-
L/A OUT	No valid Ethernet connection	-	-	Ethernet is connected	-
RUN	-	Initializing or no valid Ethernet	TCP server open for connections	TCP client connected	-
ERROR	No Errors	Fatal error	-	-	-
PWR	Power is not applied	-	-	Power is applied to both motor and module.	Power is applied to module but no communication with motor.

Notes:
Flickering: Rapid flashing with a period of approx. 50ms(10Hz).

7.2 Commissioning

7.2.2 Mechanical installation

The network cables must be connected to the two M12 connectors (marked "L/A IN" and/or "L/A OUT") on the module. The cable from the IO CONTROLLER is connected to either of the two ports. In the line topology, if there are more slave devices in the same line, the next slave device is connected to the second port. Standard CAT 5 STP cables can be used. It is not recommended to use UTP cables in industrial environments, which is typically very noisy.

7.2.3 Network configuration

To enable communication through the Ethernet network, the module needs a valid IP address. This is done by MacTalk.

7.2.4 Communication description

Connect to Modbus TCP module by opening a TCP client connection to the module IP address on port 502. It's possibly to have only one open connection at a time.

The registers in the motor and in the module are all 32 bit. To comply with the clean 16-bit Modbus standard, a 32-bit register must be read or written as two consecutive 16-bit registers. The register address mapping follows the normal documented register numbers but the address field, must be multiplied by two, so to read or write register 3, P_SOLL, use the address 6. Thereby, enabling transfer of one 32 bit register, as two 16 bit registers, where the least significant 16 bit "register" is transmitted first (see examples).

It is possibly to access both motor registers and Modbus TCP module registers. Motor registers is accessed by addressing register 0x00 – 0x1FE (for motor register 0-255), and module registers is accessed by addressing 0x8000 – 0x807E (for module register 0-64).

The Modbus TCP extension includes 7 additional bytes to the original Modbus protocol which allows for transport over the TCP/IP layers – the MBAP header. So the frame format looks like this (excluding TCP/IP header):

| - MBAP Header - | - Function Code - | - Data - |

The **MBAP Header** (ModBus Application Protocol Header) consists of 7 bytes of information:

Transaction Identifier	2 bytes	Identification of Request/Response transaction – copied from request to response
Protocol Identifier	2 bytes	0 = Modbus protocol
Length	2 bytes	number of following bytes – includes the unit identifier
Unit Identifier	1 byte	identification of remote slave.

Function codes
The MAC00-EMx Modbus TCP module supports two function codes:

0x03	Read holding registers
0x10	Write multiple registers (only 2 x 16bit registers = 1 x 32 bit register)

If an error is detected in the received request an exception frame is returned.

| - MBAP Header - | - Function Code - | - Exception code - |

The returned function code, in case of an exception, is the transmitted function code with bit 7 set (that means that 0x03 ➔ 0x83, and 0x10 ➔ 0x90).

Exception codes

0x01	Function code not supported
0x02	Not allowed register no.
0x03	Too many registers or uneven no. of registers, as every register in motor/module is 32 bit and requires 2 x 16 bit modbus registers.

(0x03) Read Holding Registers
Read of registers. Max. 124 x 16bit registers at a time (=62 x 32bit registers). Only even no. of 16bit registers is supported. The response time is increased slightly for every register added. See *Minimum poll time, page 145* for minimum poll time.

Request:

7 bytes	1 byte	2 bytes	2 bytes
MBAP header	Modbus Cmd. (0x03)	Motor register no. x 2 or module register no. x 2 + 0x8000	Register count

Response:

7 bytes	1 byte	1 byte	2 bytes	2 bytes
MBAP header	Modbus Cmd. (0x03)	Data byte count	Register value low 16bit	Register value high 16bit

Example, read of **module** register 3 (= IP address = 192.168.100.1 = 0xC0.0xA8.0x64.0x01):
Request -
|0x00|0x01|0x00|0x00|0x00|0x06|0x01|0x03|0x80|0x06|0x00|0x02|
Response – (Note the byte order!)
|0x00|0x01|0x00|0x00|0x00|0x07|0x01|0x03|0x04|0x64|0x01|0xC0|0xA8|

Possibly exception responses: 0x02, 0x03.

For further documentation see "Modbus_Application_Protocol_V1_1b.pdf" and "Modbus_Messaging_Implementation_Guide_V1_0b.pdf" found on www.modbus.org.

Please notice: Even though all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

(0x10) Write Multiple registers
Write of registers. Only 2 x 16bit registers at a time (= 1 x 32bit registers).

Request:

7 bytes	1 byte	2 bytes	2 bytes	2 bytes	2 bytes
MBAP header	Modbus Cmd. (0x10)	Motor register no. x 2 or module register no. x 2 + 0x8000	Register count	Register value low 16bit	Register value high 16bit

Response:

7 bytes	1 byte	2 bytes	2 bytes
MBAP header	Modbus Cmd. (0x10)	Motor register no. x 2 or module register no. x 2 + 0x8000	Register count

Example, write of motor register 3 (= P_SOLL = 0x12345678):
Request – (Note the byte order!)

|0x00|0x02|0x00|0x00|0x00|0x0B|0x01|0x10|0x00|0x06|0x00|0x02|0x04|0x56|0x78|0x12|0x34|

Response -
|0x00|0x02|0x00|0x00|0x00|0x06|0x01|0x10|0x00|0x06|0x00|0x02|

Possibly exception responses: 0x02, 0x03.

For further documentation see "Modbus_Application_Protocol_V1_1b.pdf" and "Modbus_Messaging_Implementation_Guide_V1_0b.pdf" found on www.modbus.org.

Note! Even though all registers is to be transmitted as 32 bit some of them originally derive from 16 bit in the MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

7.2.5 Minimum poll time

The minimum poll time is the minimum amount of time between each poll request on the Ethernet. If operating with values lower than those listed, data loss will occur.

No. of polled motor registers (32bit)	Motor series	
	MAC050-MAC141	MAC400-MAC3000
1	2ms	2ms
5	10ms	3ms
10	20ms	4ms

The minimum poll times is only valid if not sending any requests while in any operating mode. MODULE registers can be appended at no extra timing cost. If motor register 35 is not polled it will be added internally anyway and has to be added to the minimum cycle time with 2ms if MAC050-MAC141.

7.2.6 Quick start guide

This section describes the steps to configure the **MAC00-EMx** module using the shareware program **Modbus poll**, which can be obtained from the website: http://www.modbustools.com/

Set IP address

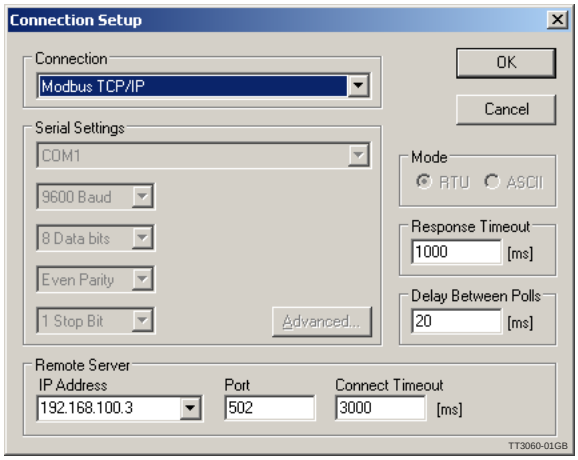
1. Connect the RS232 communication cable.
2. Apply power to the motor, and make sure the PWR LED is lit.
3. Open MacTalk and select the "MAC00-EM (Modbus TCP)" tab.
4. Change the IP address, to one suitable for the network.
5. Press "Apply and save".

Installation

6. Connect an Ethernet RJ45-M12 cable to the Ethernet interface of the PC with **Modbus Poll** installed and to L/A IN or L/A OUT on MAC00-EMx.
7. Make sure power is applied to all devices.

Connect to MAC00-EMx

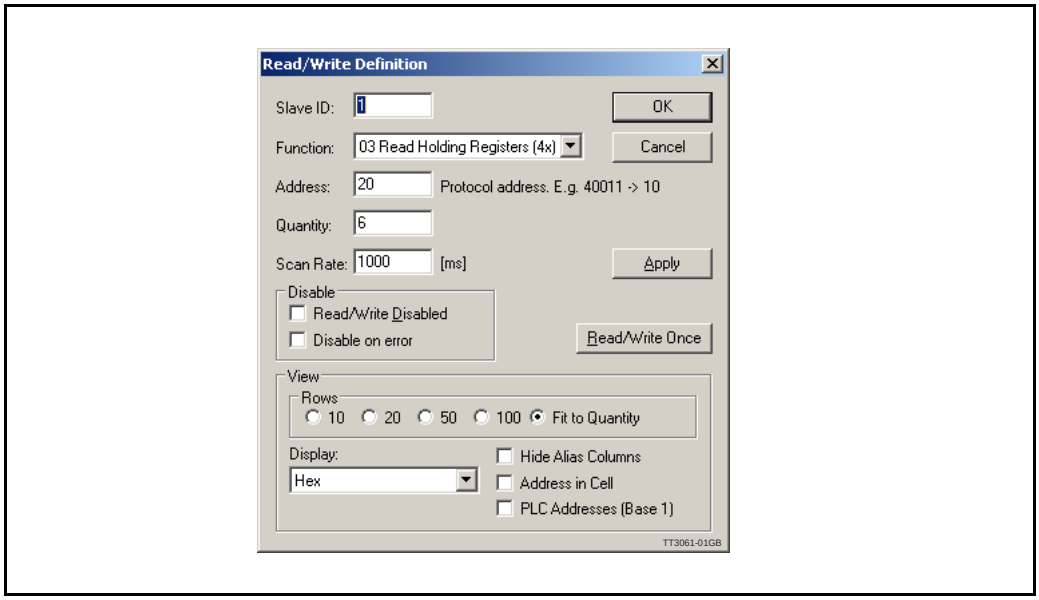
8. In the Connection menu of Modbus Poll select **Connect.**
9. The connection is made by choosing the "Modbus TCP/IP" protocol, the IP address of the motor, and port 502. As seen below.



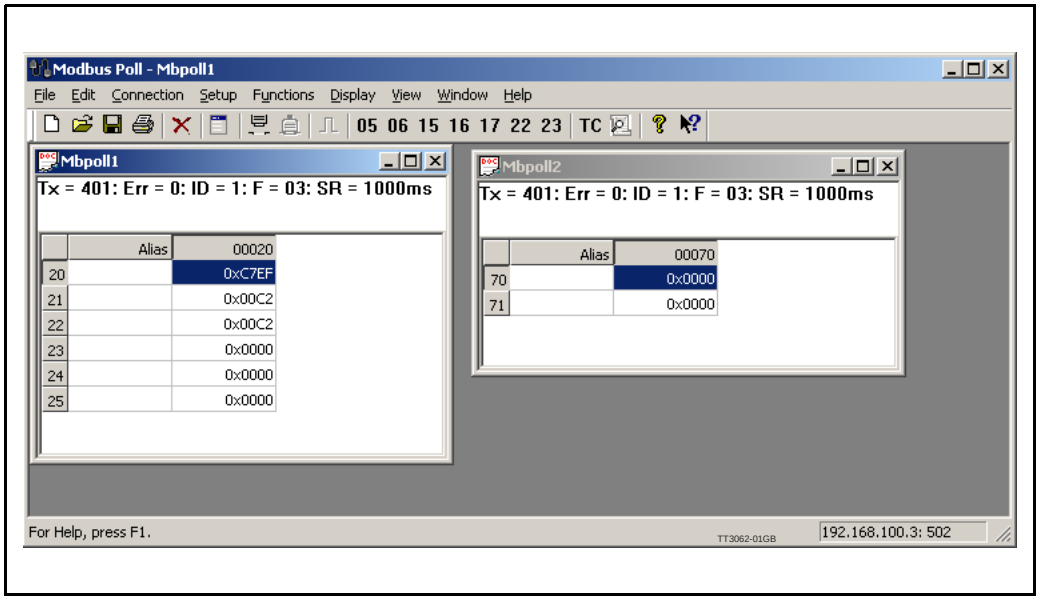
10. The "Run" led on the motor (which is red when powering up) should now change from orange to green when connected to the client (Modbus poll).

Setup poll registers

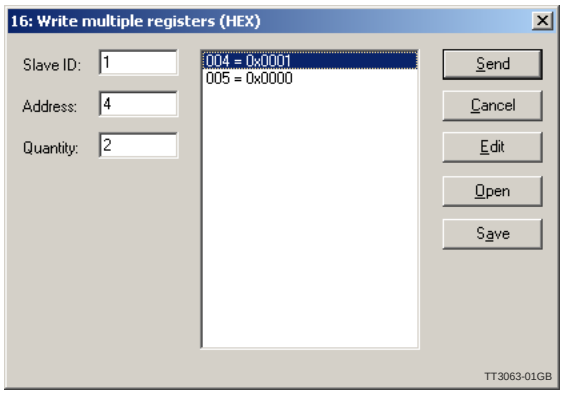
11. When connected it is possibly to change the polling of registers in the motor by right-clicking in the default "Mbpoll1" window and selecting "read/write definition". In the shown example below is chosen "Address:" 20 (= register 10), and "Quantity:" 6 (= 3 x 32bit registers). This means that register 10, 11 and 12 is polled.



12. By choosing **File** and **New** a second poll window is opened where "Address:" 70 and "Quantity:" 2 is chosen. Meaning that error register 35 is polled. Your screen should now look something like this:



Transmit data to motor
You can transmit data to the motor by chosing **Functions** and **16: Write registers**, and if choosing "Address:" 4, "Quantity:" 2, and data = 0x01 (in address 004 = least significant 16bit) as shown below (mode register = velocity) the motor should start turning. If not then try to also write velocity (reg. 5 = addr. 10), acceleration (reg. 6 = addr. 12) and/or Torque (reg. 7 = addr. 14) to some valid values. Please find a complete list of register descriptions in the appendix *Motor registers MAC050 - 141, page 171* and *Motor registers MAC400 - 3000, page 180*.



Note! Even though all registers is to be transmitted as 32 bit some of them originally derive from 16 bit in the MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

7.3 Register access

The registers in the motor and in the module are all 32 bit (at least they are when travelling through the module so special care must be taken with those registers in MAC050-141 which originally is 16bit). To comply with the clean 16-bit Modbus standard, a 32-bit register must be read or written as two consecutive 16-bit registers. The register address mapping follows the normal documented register numbers but the address field must be multiplied by two, so to read or write motor register 3, P_SOLL, use the address 6. Thereby enabling transfer of one 32 bit register, as two 16 bit registers, where the least significant 16 bit "register" is transmitted first (see examples in section 7.2.4).

Motor registers is accessed by addressing register 0x00 – 0x1FE (for logic motor register 0-255), and module registers is accessed by addressing 0x8000 – 0x807E (for logic module register 0-64). Please find a complete list of register descriptions in the appendix *Motor registers MAC050 - 141, page 171* and *Motor registers MAC400 - 3000, page 180*.

7.3.1 7.3.1 Module registers.

Logic register no.	Modbus address (hex)	Modbus address (dec)	Read only	Default	Description
0	0x8000	32768	X		Not used
1	0x8002	32770	X		High 16 bit of MAC address (placed in low 16 bit of word)
2	0x8004	32772	X		Low 32 bit of MAC address
3	0x8006	32774			IP address
4	0x8008	32776			Net mask
5	0x800A	32778			Gateway
6	0x800C	32780		0x00	Setup bits
7	0x800E	32782			Digital outputs on module
8	0x8010	32784			Reserved for other protocols
9	0x8012	32786			Reserved for other protocols
10	0x8014	32788			Modbus timeout. 0 = timeout function disabled
11-14	0x8016 – 0X801C				Reserved for future use
15	0X801E	32798			Command register
16 – 31	0x8020 – 0x803E				Reserved for other protocols
32	0x8040	32832	X		Module serial no.
33	0x8042	32834	X		Module hardware version
34	0x8044	32836	X		Module software version
35	0x8046	32838	X		No. of internal motor communication timeouts
36	0x8048	32840	X		No. of retry frames to motor
37	0x804A	32842	X		No. of discarded frames to motor
38	0x804C	32844	X		Total no. of frames to motor
39-46	0x804E – 0X805C		X		Reserved for future use
47	0x805E	32862	X		Digital inputs on module
48	0x8060	32864	X		Status bits
49-63	0x8062 – 0X807E		X		Reserved for future use

Note: Module parameters are not automatically saved to permanent memory after a change. The parameters can be saved permanently by applying a "Save parameters to flash" command afterwards.

Register 1 MAC address MSB.
The 2 most significant bytes of module MAC Address.

Bit	16-31	0-15
Output	Reserved	16 most significant bits of MAC address.

Register 2 MAC address LSB.
The 4 least significant bytes of module MAC Address.

Bit	0-31
Output	32 least significant bits of MAC address.

Register 3 IP address.
This is the IP address of the device.

Bit	0-31
I/O	IP address

Register 4 Netmask.
This is the netmask of the device. The netmask is fixed.

Bit	0-31
I/O	Subnet mask

Register 5 Gateway.
This is the gateway address of the device. The gateway address is also fixed.

Bit	0-31
I/O	Gateway

Register 6 Setup bits
This register is used to setup how the module reacts on different events.

Bit	6-31	5	2-4	1	0
Output	Reserved	1: Enable mirror of module registers at address 0x300	Reserved	0: Enable Ethernet error handling 1: Disable ethernet error handling	0: Ethernet error handling = motor set passive mode. 1: Ethernet error handling = motor set velocity to 0.

Register 7 Digital outputs on module
With this object the digital outputs can be controlled.
The value written to this object is directly shown on the digital outputs

Bit	2-31	1	0
Output	Reserved	OUT2 *	OUT1 *

* The availability of the outputs depends on the actual version of the module used. Example MAC00-EM4 only support Output 1 (OUT1).

Register 10 Modbus time-out
The Modbus TCP protocol does not have an implementation for timeout on application layer and this may be required when controlling a drive. A supervision method has been implemented for this purpose. If modbus timeout is set to zero, this feature is disabled. The unit of the parameter is 100ms (e.g. “35” will give 3.5 seconds).

Bit	16-31	0-15
Output	Reserved	Modbus timeout in units of 100ms.

7.3 Register access

Register 15 Command register
This register is used for sending commands to the module and is write only. The possible commands are listed in the table below.

Command no.			Command description	
	Hex	**Dec**	**MAC050 - MAC141**	**MAC400 – MAC3000**
Module only commands				
	0x 0000 0000	0	No operation	< Same as
	0x 0000 0001	1	Reset the module	< Same as
	0x 0000 0010	16	Save module parameters to flash	< Same as
Synchronized commands				
	0x 0000 0101	257	Simultaneous reset of the motor and the module	< Same as
	0x 0000 0110	272	Save the motor parameters in flash memory, and do a re-sync. of internal communication afterwards.	< Same as
Motor only normal commands *(via module cmd register)*				
	0x 8000 0001	2147483649	Reset motor *(not recommended, use synchronized version instead)*.	< Same as
	0x 8000 0002	2147483650	Save motor parameters in flash and reset motor *(not recommended, use synchronized version instead)*.	< Same as
Motor only FastMac commands *(via module cmd register)*				
	0x8000 00E0	2147483872	No operation	< Same as
	0x8000 00E1	2147483873	Reset error *(Clear error bits in motor register 35)*	< Same as
	0x8000 00E2	2147483874	P_SOLL = 0	< Same as
	0x8000 00E3	2147483875	P_IST = 0	< Same as
	0x8000 00E4	2147483876	P_FNC = 0	< Same as
	0x8000 00E5	2147483877	V_SOLL = 0	< Same as
	0x8000 00E6	2147483878	T_SOLL = 0	< Same as
	0x8000 00E7	2147483879	Reset IN_POS, AC C,DEC	< Same as
	0x8000 00E8	2147483880	P_FNC = (FLWERR - P7) * 16	< Same as
	0x8000 00E9	2147483881	P_FNC = (FLWERR - P8) * 16	< Same as
	0x8000 00EA	2147483882	Reserved	< Same as
	0x8000 00EB	2147483883	Reserved	< Same as
	0x8000 00EC	2147483884	Activate P1,V1,A1,T1,L1,Z1	< Same as
	0x8000 00ED	2147483885	Activate P2,V2,A2,T2,L2,Z2	< Same as
	0x8000 00EE	2147483886	Activate P3,V3,A3,T3,L3,Z3	< Same as
	0x8000 00EF	2147483887	Activate P4,V4,A4,T4,L4,Z4	< Same as
	0x8000 00F0	2147483888	Start search zero	< Same as
	0x8000 00F1	2147483889	P_SOLL = P_IST + P7;	P_SOLL = P_IST + P7 – FLWERR;
	0x8000 00F2	2147483890	P_SOLL = P_IST + P8;	P_SOLL = P_IST + P8 – FLWERR;
	0x8000 00F3	2147483891	Reserved	< Same as
	0x8000 00F4	2147483892	Select absolute position mode	< Same as
	0x8000 00F5	2147483893	Select relative position mode using P_SOLL	< Same as
	0x8000 00F6	2147483894	Select relative position mode using P_FNC	< Same as
	0x8000 00F7	2147483895	Synchronize position manually using absolute new values. P_IST = P_NEW; P_SOLL = P_NEW; P_FUNC = P_NEW * 16;	Synchronize position manually using absolute new values. P_IST = P_NEW; P_SOLL = P_NEW; P_FNC = (P_NEW + FLWERR)*16;
	0x8000 00F8	2147483896	Synchronize position manually using relative new values. (basically offset the position range with the value of P_NEW). P_IST = P_IST + P_NEW; P_SOLL = P_SOLL + P_NEW; P_FUNC = P_FUNC + (P_NEW * 16);	< Same as
	0x8000 00F9	2147483897	No operation	< Same as
	0x8000 00FA	2147483898	No operation	< Same as
	0x8000 00FB	2147483899	No operation	< Same as
	0x8000 00FC	2147483900	No operation	< Same as
	0x8000 00FD	2147483901	Reserved	< Same as
	0x8000 00FE	2147483902	Reserved	< Same as
	0x8000 00FF	2147483903	Reserved	< Same as

Register 32-38 (read only)
These registers contain HW, SW and communication information of the module.

Register 47 Digital inputs on module (read only)
With this object the status of the 4 digital inputs can be read.

Bit	4-31	3	2	1	0
Input	Reserved	IN4 *	IN3 *	IN2 *	IN1 *

* The availability of the inputs depends on the actual version of the module used. Example MAC00-EM4 only support Input 1 (IN1).

Register 48 Status bits (read only)
This register is used for miscellaneous information about the module and the motor. More detailed motor status is found in the motor error register (35).

Bit	1-31	7	6	5	4	3	2	1	0
Output	Reserved	Reserved	Reserved	Reserved	No communication with motor	Temperature error	Voltage error	Current error	Generic error

7.4 Examples

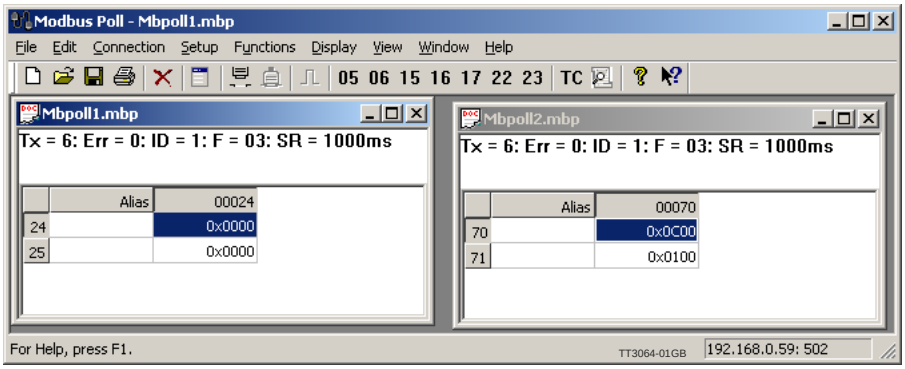
In this section is shown some examples of controlling the motor by Modbus TCP. As master is used the shareware program **Modbus poll** which can be obtained from the website: http://www.modbustools.com/
These examples assume you are already connected to the motor.
For connecting to the motor, please follow the *Quick start guide, page 146*.

7.4.1 Running Velocity control

To use the JVL motor in velocity mode the following motor registers is of interest.

1. "Mode" – mode, register 2
2. "V_SOLL" – velocity, register 5
3. "A_SOLL" – acceleration, register 6
4. "Error/Status" – register 35

So to control these registers setup polling of motor register 12 – actual velocity (modbus address 24), and motor register 35 (modbus address 70).
This could look like shown below.



Now we can monitor the motor errors and the motor velocity.

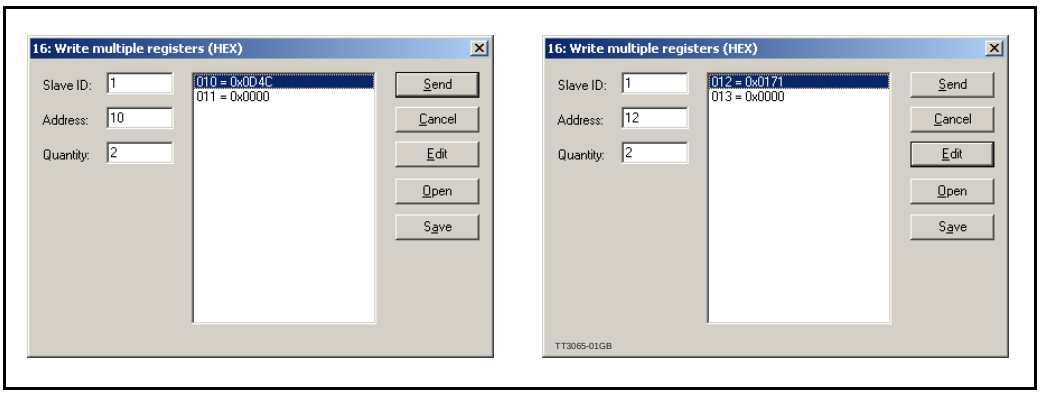
7.4 Examples

Calculate the values needed for velocity and acceleration (constant values valid for MAC400, MAC1500 and MAC3000).

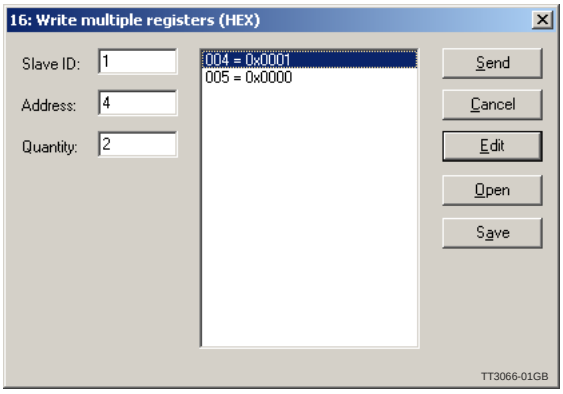
1. **Set the needed velocity**. V_SOLL = V x 2.8369 [rpm]
   Ex. *We need the motor to run with a constant speed of 1200 RPM. So,*
   V_SOLL = 1200 x 2,8369 = 3404 cnt/smp (= 0x0D4C)

2. **Set the needed acceleration**. A_SOLL = A / 271 [RPM/s^2]
   Ex. *We need the motor to accelerate with 100,000 RPM/s^2 so,*
   A_SOLL = 100,000/271 = 369 cnt/smp^2 (= 0x0171)

Insert the calculated values in send frames and send to motor as shown below (modbus address 10-11 = register 5, modbus address 12-13 = register 6). Remember to press the send button for every new value.



Now set the motor in velocity mode and thereby activate the motor.
Ex. *The motor needs to be activated by setting it into velocity mode, so we need to set the mode register to the value 1. Mode = 1 which is velocity mode, now the motor will use the acceleration and the velocity just configured.* (Modbus address 4-5 = register 2).



7.4.2 Running Position control

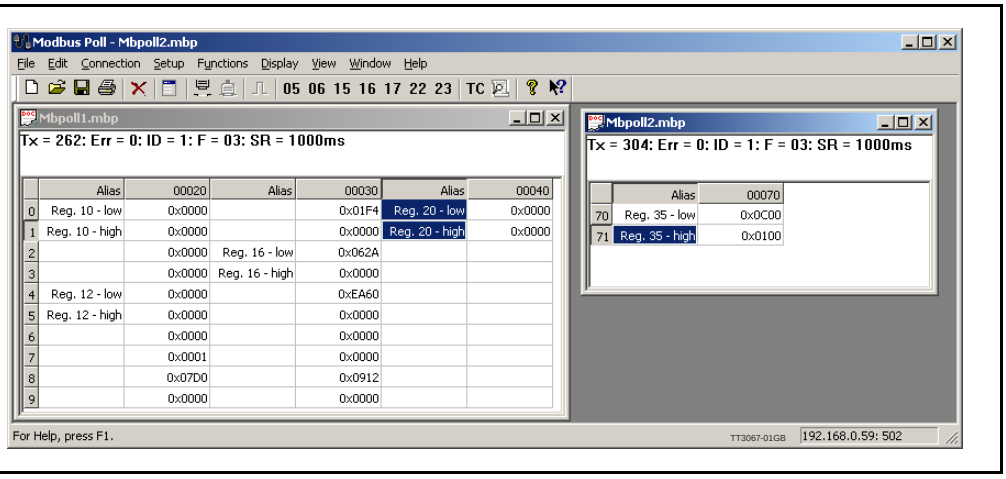
Running the motor in position control requires that the mode register is set for position control. The following registers is of particular interest when position mode is used.

- **Poll registers**
  - "Actual position" -P_IST, register 10
  - "Actual velocity" -V_IST, register 12
  - "Motor load mean" - average motor load, register 16
  - "Follow error" - The actual position error, register 20
  - "Error/Status" -register 35

- **Write registers**
  - "Mode" – mode, register 2
  - "Requested position" -P_SOLL, register 3
  - "Requested velocity" -V_SOLL, register 5
  - "Requested acceleration" -A_SOLL, register 6

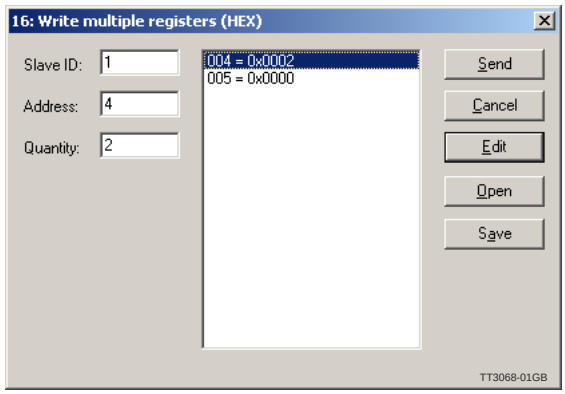
In this mode the position is controlled by applying a requested position to the "P_SOLL" -register and the actual position is monitored in the "P_IST" register. The V_SOLL and A_SOLL registers sets the velocity and acceleration used when positioning occurs.

For easy setup we can use a single poll setup for the registers 10,12,16 and 20, and another for register 35, see figure below but it also is possibly to setup one poll instance for every single register.

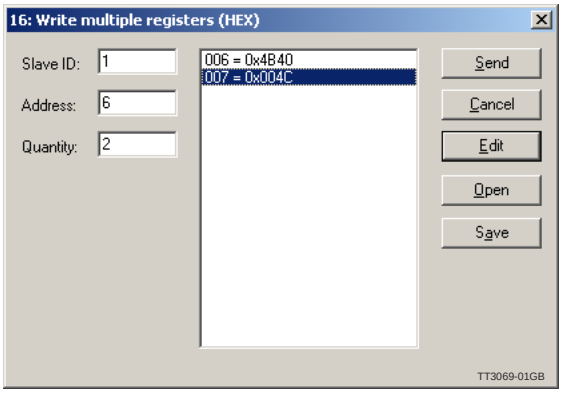


Calculate the values needed for velocity and acceleration and send to the motor, see previous example.

Now set the motor into position mode and thereby activate the motor.
Ex. *The motor needs to be activated by setting it into position mode so we need to set the mode register to the value 2. Mode = 2 which is position mode, now the motor will use the acceleration and the velocity just configured.* (Modbus address 4-5 = register 2).



Set a position in the motor by writing a position to register 3 (P_SOLL = Modbus address 6-7), in the example shown below is used position 5,000,000 (= 0x 004C 4B40), remark the order.



7.4.3 General considerations

The register 35 in the motor holds information on the actual error/status. So it is crucial that this register is configured in the polled data and thereby obtained and monitored in the Master. In case of an error situation the motor will stop and the cause will be present in the register 35.
This register also holds information on the motion status such as:

- In position, bit 4
- Accelerating, bit 5
- Decelerating, bit 6

Please find a complete list of register descriptions in the appendix *Motor registers MAC050 - 141, page 171* and *Motor registers MAC400 - 3000, page 180*.

The JVL motor is basically put into a working mode and into a passive mode where the motor axle is de-energized, by setting register 2 into either 0 = "passive mode" or into one of the supported modes.

Example.
1= "Velocity mode" / 2= "Position mode" / etc.

8 Using MacTalk over Ethernet

8.1 Using MacTalk over Ethernet

8.1.1 Introduction

The configuration software tool MacTalk is able to connect to a motor either using a serial connection or an Ethernet based TCP/IP connection.

Please notice that there are some limitations/precautions.

- Currently **only** the **MAC00-EPx (PROFINET IO)** and the **MAC00-EIx (EtherNetIP)** -modules are supported. PROFINET IO firmware version must be V.3.17 Build 425 or higher and EthernetIP V3.21 Build 425 or higher.

- Make sure the motor has the latest firmware installed, that is V2.05 for MAC400-3000 and V9.01 for MAC50-141. Ethernet connectivity is only supported in the MIS34x series of stepper motors. For the MIS34x please use firmware V1.12 or greater. All the firmwares required should be included in the install package for MacTalk or by using the internet update feature in MacTalk.

- Make sure that Mactalk is version 1.50.49 or newer.

- At the moment firmware update is still only possible using the standard serial connection.

The hardware required is the mandatory 24V supply for the motor and the Ethernet cable going either from an Ethernet switch or directly from the PC to the M12 connector on the MAC00-Exx module in the motor.
In order to establish the Ethernet connection from the PC where MacTalk is running, to the motor the PC and the motor needs to be configured to run on the same network. By default the motor is configured to run on the following IP-address: 192.168.0.XX at startup where XX refers to the last 2 digits in the MAC-ID which is printed on a label. So, if a MAC-ID has the value: 00 : 50 : C2 : D0 : C9 : 03, then the IP address is set to: 192.168.0.3 The PC from where MacTalk is used needs to be configured for this IP range.
The in depth PC – configuration is beyond the scope of this manual since this greatly depends on the networks equipment end network connected. However a brief description on how to configure the IP address manually is discussed. This method is necessary if the motor is connected directly to the Ethernet port in the PC or if the network isn't capable of assigning IP addresses to connected equipment automatically.

8.2 Setting up the Ethernet at the PC

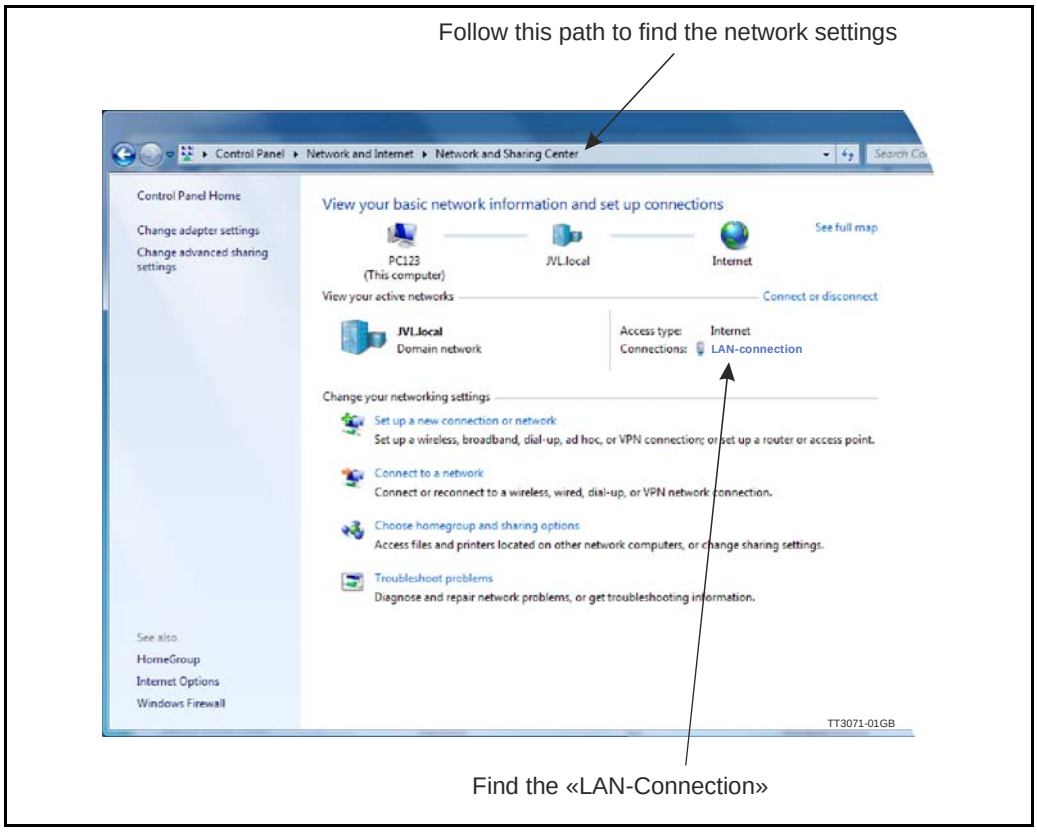
8.2.1 Setting up the Ethernet at the PC.

When a connection is made directly from the PC it is very important to observe the IP-settings of the PC, since the most common way is for the PC to receive the settings from a server such as a DHCP/server or similar.
Since the motor doesn't offer any DHCP service it is necessary to setup the IP-address in the PC manually.

Please note that this is taken from Windows 7, but the method is basically the same for other Windows version.

To reach the IP settings please follow this path:

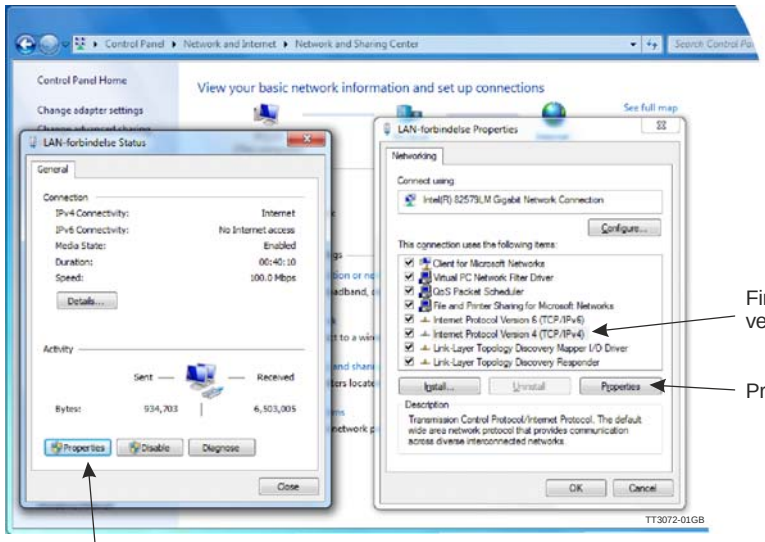
Step 1.
Press the LAN-Connection and press the "Properties".



Continued next page

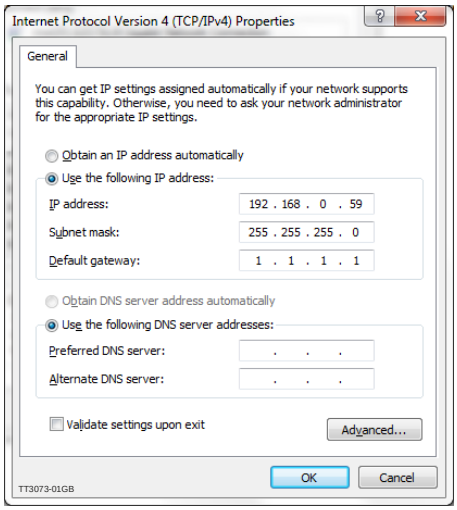
8.2 Setting up the Ethernet at the PC

Step 2.
Find the “Internet protocol version 4 (TCP/IPv4)” and press “Properties”.



Now the settings finally appears and we are able to change the IP address, subnet mask and gateway.

Step 3.
Select “Use the following” and enter a valid configuration similar to the one below.



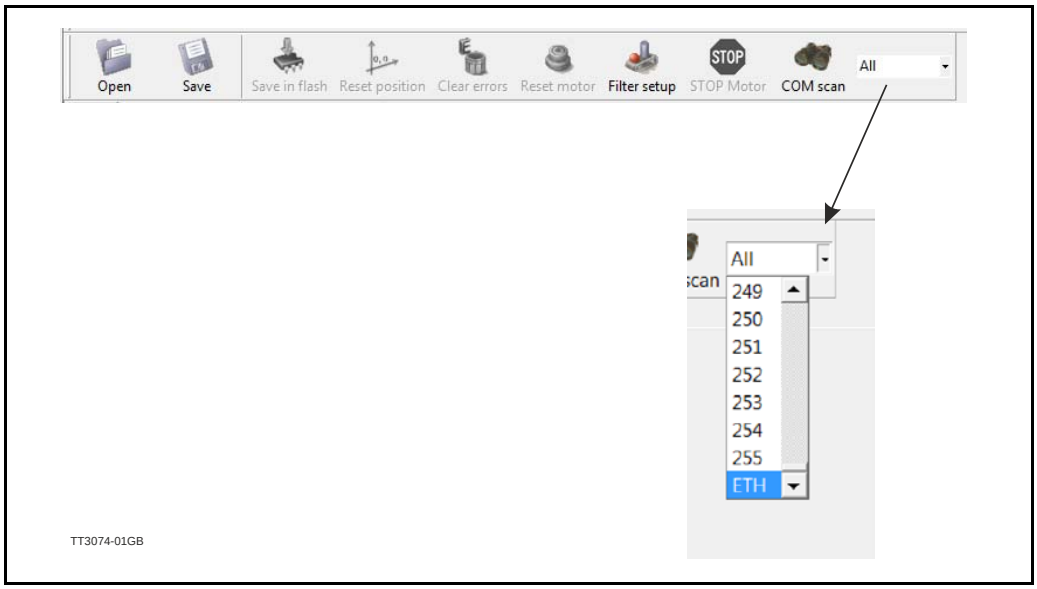
The above example is a basic settings that sets the IP address on the PC to 192.168.0.59, subnet mask to 255.255.255.0 and the gateway to 1.1.1.1.
Now the PC is configured for a fixed IP address and is ready to establish the connection to the motor.

8.3 Setting up MacTalk for Ethernet

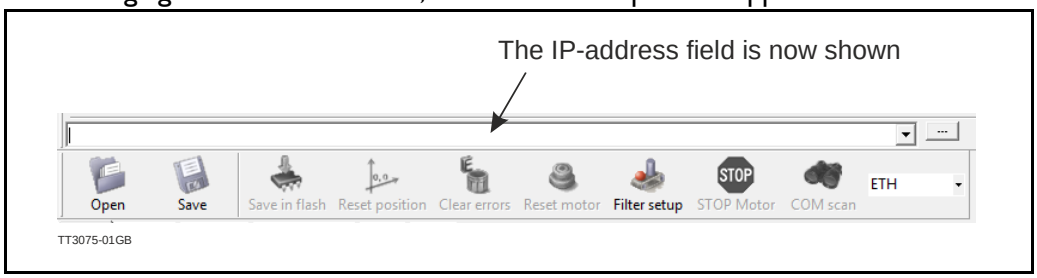
8.3.1 Setting up MacTalk for Ethernet communication

When MacTalk is opened the first time it is, by default configured for running serial RS232/RS485 connection. To change this please find the address box next to the "COM scan" in the upper tool bar and change it from "All" to "Eth".

Step 1.
Select "ETH" as shown below.



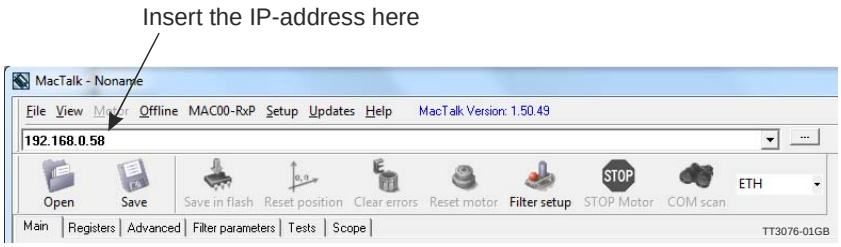
Step 2.
After changing the the Address box, the IP-address input field appears.



Step 3.
Now MacTalk is ready to connect to the motor and the next step is to enter the IP address of the motor to connect to.

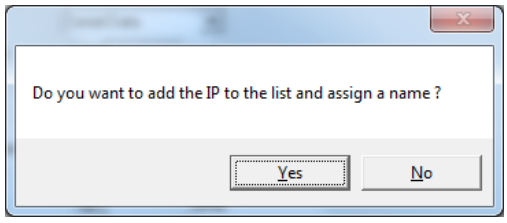
8.3 Setting up MacTalk for Ethernet

Step 4.
Lets assume that the motor with the IP address 192.168.0.58 is connected to the PC from where MacTalk is running or the same network that the PC is running, we enter the IP address.

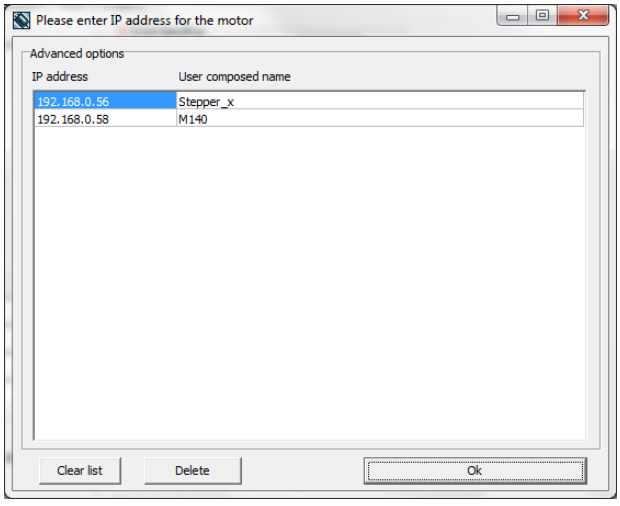


Step 5.
Since it is the first time the address is entered MacTalk offers the possibility to sign in the IP address and assign an alias name to this IP address which is stored and later be shown in the address field instead of remembering the IP address of the motor. This greatly helps managing multiple motors in a network instead of handling all the ”anonymous” IP addresses.

Step 6.
The following dialog appears when a new address is entered.



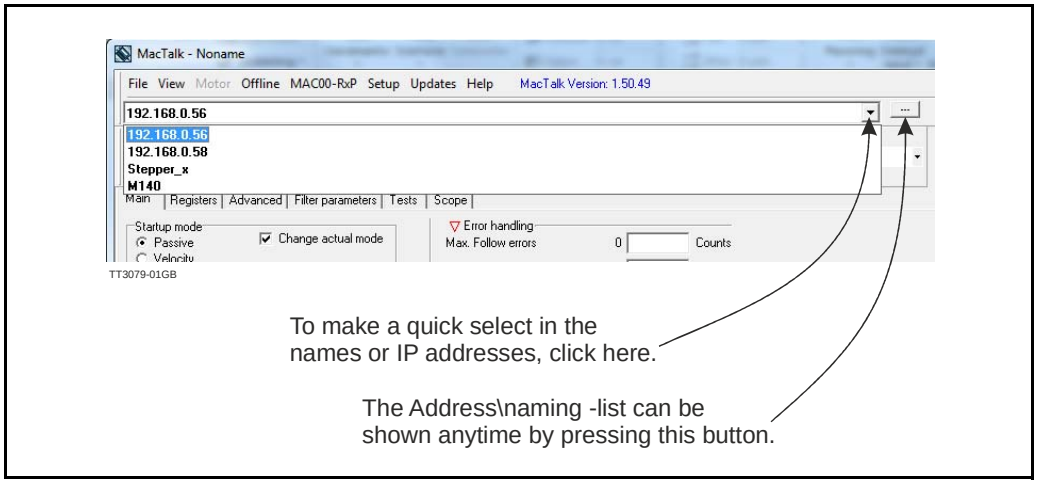
Step 7.
Pressing “Yes” will show the list of IP addresses and user composed names.



8.3 Setting up MacTalk for Ethernet

Step 8.
In the list presented we have added a motor with the IP/address 192.168.0.56. This motor is stepper motor so we name it “Stepper_x” to be easy recognizable. We also have a MAC140 motor in the network, for this motor we have assigned the name M140.
The list is added to the address bar which automatically suggest the motor when we type in the first letters of the name. The motor can also be selected directly in the list. Please note that both the IP address and the name is added to the list and saved. The list is loaded automatically when MacTalk is started.

Step 9.
Add a name to the list in the field next to the IP address and press “Ok”, Now the list is saved. The name entered can now be used to access the motor on the network. The complete list can be cleared by pressing “Clear list” or a single enty can be deleted by pressing “Delete”.
When MacTalk is started this list is read and added to the address bar selections, so that either the name or the IP address can be selected.



9 Appendix

9.1 Technical Data

9.1.1 MAC00-EI4 EthernetIP - Technical specifications

Galvanic isolated, 100MBit, full duplex, 100Base-Tx, no termination necessary.
Network topology: Line, Star, Tree, Ring.
Supported Protocols:
- DHCP - Dynamic Host Configuration Protocol
- BOOTP – Bootstrap Protocol
- ACD - Address Conflict Detection
- DLR - Device Level Ring (ring topology on device level)

Max. 100 m cable between slaves.
Connectors: "PWR" (power) M12 connector 5pin male
"I/O" M12 connector 8pin female
"L/A IN" and "L/A OUT" (Ethernet) M12 connector 4pin D-coded female.
Supply voltage (CV): 10-25V
Current rating (CV): typical 150mA, max. 250mA
User inputs:
Input impedance: 4.7k
Input current @24V: 5.1mA
Digital output current (HW rev. Up to 1.2): 10mA
Digital output current (HW rev. from 1.3): 15mA

9.1.2 MAC00-EC4 EtherCAT - Technical specifications

Galvanic isolated, 100MBit, full duplex, 100Base-Tx, no termination necessary.
Network topology: Line, Star, Tree, Ring (line recommended)
Pass through delay: $< 1\mu s$
Supported Protocols:
- SDO client and server side protocol
- CoE Emergency messages (CoE stack)

Max. 100 m cable between slaves.
Maximum number of slaves: 65535
Connectors: "PWR" (power) M12 connector 5pin male
"I/O" M12 connector 8pin female
"L/A IN" and "L/A OUT" (Ethernet) M12 connector 4pin D-coded female.
Supply voltage (CV): 10-25V
Current rating @ 24V DC (CV): typical 150mA, max. 250mA
User inputs:
Input impedance: 4.7k
Input current @24V: 5.1mA
Digital output current (HW rev. Up to 1.2): 10mA
Digital output current (HW rev. from 1.3): 15mA

9.1.3 MAC00-EL4 Powerlink - Technical specifications

Galvanic isolated, 100MBit, half duplex, 100Base-Tx, no termination necessary.
Network topology: Line and tree possibly (line recommended)
Pass through delay: <0.5µs.
Acyclic data tramsfer: SDO Upload/Download
Functions: SDO over ASND and UDP
Ethernet Powerlink version: V2
Max. 100 m cable between slaves.
Maximum number of slaves (CN's) per segment: 239
Connectors: "PWR" (power) M12 connector 5pin male
"I/O" M12 connector 8pin female
"L/A IN" and "L/A OUT" (Ethernet) M12 connector 4pin D-coded female.
Supply voltage (CV): 10-25V
Current rating @ 24V DC (CV): typical 150mA, max. 250mA
User inputs:
Input impedance: 4.7k
Input current @24V: 5.1mA
Digital output current (HW rev. Up to 1.2): 10mA
Digital output current (HW rev. from 1.3): 15mA

9.1.4 MAC00-EP4 PROFINET IO - Technical specifications

Galvanic isolated, 100MBit, full duplex, 100Base-Tx, no termination necessary.
Network topology: Line, ring, tree and star possibly.
Forwarding delay: 3.25µs.
Minimum cycle time: 1ms (with MAC400-3000).

Supported Protocols

- CL-RPC – Connection less Remote Procedure Call
- DCP – Discovery and Configuration Protocol
- LLDP – Link Layer Discovery Protocol
- RTA – Real time Acyclic Protocol
- RTC – Real time Cyclic Protocol, Class 1
- SNMP – Simple Network Management Protocol
- MRP – MRP Client is supported

Max. 100 m cable between slaves.

Connectors:

- "PWR" (power) M12 connector 5pin male
- "I/O" M12 connector 8pin female
- "L/A IN" and "L/A OUT" (Ethernet) M12 connector 4pin D-coded female.

Supply:
Supply voltage (CV): 10-24V
Current rating @ 24V DC (CV): typical 150mA, max. 250mA

User I/O:
Digital input impedance: 4.7k
Digital input current @24V: 5.1mA
Digital output current (HW rev. Up to 1.2): 10mA
Digital output current (HW rev. from 1.3): 15mA

9.1.5 MAC00-EM4 Modbus TCP/IP - Technical specifications

Galvanic isolated, 100MBit, full duplex, 100Base-Tx, no termination necessary.
Network topology: Line, ring, tree and star possibly.
Forwarding delay: 10-130μs.
Minimum cycle time: 2ms (with MAC400-3000).
Max. 100 m cable between slaves.

Protocol:

- Function codes supported: 3, 16.
- Max. 124 modbus read registers per frame (= 62, 32bit registers).
- Max. 2 modbus write registers per frame (= 1, 32bit register)
- 32bit support by 2x16bit registers. ➔ Only even no. of 16bit registers.
- I/O mode: Server, port 502.

Connectors:

- "PWR" (power) M12 connector 5pin male
- "I/O" M12 connector 8pin female
- "L/A IN" and "L/A OUT" (Ethernet) M12 connector 4pin D-coded female.

Supply:

Supply voltage (CV): 10-24V
Current rating @ 24V DC (CV): typical 150mA, max. 250mA

User I/O:

Digital input impedance: 4.7k
Digital input current @24V: 5.1mA
Digital output current (HW rev. Up to 1.2): 10mA
Digital output current (HW rev. from 1.3): 15mA

9.2 Motor registers MAC050 - 141

9.2.1 Register list for MAC050, 095, 140 and 141.

The following list is only valid for the MAC50, MAC95, MAC140 and MAC141 motors.

Please notice: At the Ethernet modules all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
0	N/A	N/A	N/A	N/A	N/A	Dummy register, do not use
1	PROG_VERSION	Displayed on bottom right status line.				Firmware version number.
2	MODE_REG	Startup mode / Change actual mode				The actual operating mode of the drive. In general, the motor will either be passive, attempt to reach a certain position, attempt to maintain a constant velocity or attempt to produce a constant torque. The various modes define the main type of operation as well as what determines the setpoint for that operation. The special cases 256..258 are used to perform a few special operations on the entire set of registers. Supported values are: 0 : **Passive mode**. The axis is not controlled by the drive, and can easily be moved by hand or external mechanics. 1 : **Velocity mode**. The drive will attempt to run the motor at a constant velocity selected by Reg5, V_SOLL, without violating the maximum torque or acceleration. 2 : **Position mode**. The drive will at all times attempt to move the actual motor position to the position selected by Reg3, P_SOLL, without violating the maximum velocity, torque or acceleration. 3 : **Gear Position mode.** 4 : **Analogue torque mode.** 5 : **Analogue velocity mode.** 6 : **Analog Velocity Gear mode.** 7 : **Manual current mode.** 8 : **Step response test mode.** 9 : **Internal test mode.** 10 : **Brake mode.** 11 : **Stop mode.** 12 : **Torque based zero search mode.** 13 : **Forward/only zero search mode.** 14 : **Forward+backward zero search mode.** 15 : **Safe mode.** 16 : **Analogue velocity with deadband mode.** 17 : **Velocity limited Analog Torque mode.** 18 : **Analogue gear mode.** 19 : **Coil mode.** 20 : **Analogue bi-position mode.** 21 : **Analogue to position mode.** 22 : **Internal test mode.** 23 : **Internal test mode.** 24 : **Gear follow mode.** 25 : **IHOME mode.** 256 257 258
3	P_SOLL, 32-bit	Position	-67M - +67M	32 bit R / W		The target position that the drive will attempt reach in position related modes.
4	(high word of P-SOLL)	-	-			-

9.2 Motor registers MAC050 - 141

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
5	V_SOLL	Max. Velocity				The maximum velocity the motor is allowed to use.
6	A_SOLL	Acceleration			Counts/ Sample	The maximum acceleration in counts/sample^2 the drive is allowed to use during normal operation. Also note Reg32, ACC_EMERG, used during emergency stops.
7	T_SOLL	Torque	0-1023		-	The maximum torque that the drive is allowed to use. The value 1023 corresponds to 300% of nominal load, and is the absolute maximum peak torque allowed. The value 341 gives 100% (nominal load).
8	P_FNC, 32-bit (Sometimes named P_SIM)				Counts	
9	(high word of P_FNC/P_SIM)					
10	P_IST, 32-bit	Actual position				The actual motor position measured by the internal encoder. Updated every 1.9ms. Note that this register is maintained incrementally, which means that the user can update it to offset the working range. When updating when the drive is not in Passive mode, P_IST and P_SOLL should be updated together in an atomic operation, using Reg163, P_NEW, or other special measures. Also note that the firmware will change this register after a zero search operation has completed.
11	(high word of P_IST)	-	-			-
12	V_IST					Actual velocity of the drive.
13	KVOUT	Load factor				Ratio of the total inertia driven by the drive to the inertia of the motors rotor itself.
14	GEARF1					Gear factor 1, Nominator
15	GEARF2					Gear factor 2, Denominator
16	I2T					Energy dissipated in the motor windings.
17	I2TLIM					Safety limit for I2T above. Motor will set an error bit if I2T gets above I2TLIMIT.
18	UIT					Energy dissipated in the internal power dump.
19	UITLIM					Limit for Reg18, UIT. Motor will set an error bit if UIT gets above UITLIM
20	FLWERR, 32-bit					A measure of how far the drive is from its ideal regulation goal. This value is calculated differently in the various modes, and can mean things like pulses from theoretical position or difference in actual velo city to V_SOLL. Contact JVL for more detailed information for specific modes.
21	(high word of FLWERR)					
22	FLWERRMAX, 32-bit					When Reg20, FLWERR, exceeds this limit, an error bit is set in Reg35, ERR_STAT, and the motor will stop if Reg22 is non-zero. Usually this value is set experimentally to detect situations where a movement is blocked or fails.
23	(high word of FLWERRMAX)					
24	FNCERR, 32-bit					Shows how much the motor is behind the ideal movement; precise operation depends on mode. When this accumulated value exceeds Reg26, FNCERRMAX, the FNC_ERR bit is set in Reg35, ERR_STAT and the motor will stop.

TT1521GB

9.2 Motor registers MAC050 - 141

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
25	(high word of FNCERR)					
26	FNCERRMAX, 32-bit					
27	(hi-word of FNCERRMAX)					
28	MIN_P_IST, 32-bit					
29	(hi-word of MIN_P_IST)					
30	MAX_P_IST, 32-bit					
31	(hi-word of MAX_P_IST)					
32	ACC_EMERG					
33	INPOSWIN					
34	INPOSCNT					
35	ERR_STAT					**Bit 0, I2T_ERR** Too much energy dissipated in the motor windings. Set when Reg16, I2T, exceeds Reg17, I2TLIM **Bit 1, FLW_ERR** The actual position is too far behind the ideal position. Set when FLWERRMAX is non-zero, and FLWERR exceeds FLWERRMAX. **Bit 2, FNC_ERR** The value of Reg24, FNCERR, exceeded the value of Reg26, FNCERRMAX. **Bit 3, UIT_ERR** The value of Reg18, UIT, exceeded the value of Reg19, UITLIM. **Bit 4, IN_POS** For position-related modes: The actual position was detected to be inside the InPosition window (Reg33, INPOSWIN) at least the number of times defined in Reg34, INPOSCNT. For other modes: Depends on mode; for velocity related modes, this bit means AtVelocity; for other more special modes, this bit is calculated differently, ask JVL for details. **Bit 5, ACC_FLAG** The drive is currently accelerating (the velocity is increasing). **Bit 6, DEC_FLAG** The drive is currently decelerating (the velocity is decreasing). **Bit 7, PLIM_ERR** One of the software position limits was exceeded,, drive will go into stop mode, then passive mode automatically. **Bit 8, FRAME_ERR_TX** A framing error was detected during the last reception on the FastMac protocol. Continued next page

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
35	ERR_STAT (cont. from last page)					**Bit 9, RELPOSPSOLL** **Bit 10, RELPOSPFNC** These two bits determine what will happen when one of the eight general purpose position registers, P1-P8 is activated through either a FastMac command (including activating s register group), through writing to Reg43, P_REG_P,on changes in bi-position mode or during manual resynchronization. If both are zero, the P register gets copied to the target register(s). If Bit 9 is set, the value of Reg3, P_SOLL, is added to the target register(s) to make a relative movement. If Bit 10 is set, the value of Reg8, P_FNC, is added to the target register(s) to make a relative movement. **Bit 11, IX_ERR** The current in at least one of the motor windings was measured to be too high, possibly because of bad current loop filter settings. Values for the current filter have been overwritten with default values. Specifically registers 106 through 111, 127 and 128. **Bit 12, UV_ERR** The motor power supply voltage (Reg151, U_SUPPLY) was measured to be below the value in Reg152, U_MIN_SUP and the drive was configured to set an error bit in case of undervoltage. **Bit 13, UV_DETECT** The motor power supply voltage (Reg151, U_SUPPLY) was measured to be below1.25 times the value in Reg152. This is a warning bit, not an error. **Bit 14, DIS_P_LIM** When this bit is set (during zero search or by the user), the drive will disable its position limits so it can move also outside the position limit range. This bit is cleared automatically when the actual position gets inside the position range again. **Bit 15, SSI_ERROR**

TT1523GB

9.2 Motor registers MAC050 - 141

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
36	CNTRL_BITS					Bit 0, USRINTF0 Bit 1, USRINTF1 Bit 2, PULSEDIR Bit 3, INPSIGN Bit 4, HICLK Bit 5, HALL_INT Bit 6, RECORDBIT Bit 7, REWINDBIT Bit 8, RECINNERBIT Bit 9, AUTO_RESYNC Bit 10, MAN_RESYNC Bit 11, INDEX_HOME Bit 12, REL_RESYNC Bit 13, HALL_C Bit 14, HALL_B Bit 15, HALL_A
37	STARTMODE					
38	P_HOME, 32-bit					
39	(hi-word of P_HOME)					
40	V_HOME					Velocity used during Zero Search/Homing
41	T_HOME					Negative => home on falling edge of AN_INP
42	HOMEMODE					Used by FastMac commands
43	P_REG_P					
44	V_REG_P					
45	A_REG_P					
46	T_REG_P					
47	L_REG_P					
48	Z_REG_P					
49	POS0 / P1, 32-bit					
50	(hi-word of P1)					
51	POS1 / P2, 32-bit					
52	(hi-word of P2)					
53	POS2 / P3, 32-bit					
54	(hi-word of P3)					
55	POS3 / P4, 32-bit					
56	(hi-word of P4)					
57	POS4 / P5, 32-bit					
58	(hi-word of P5)					
59	POS5 / P6, 32-bit					Bit 0, COIL_START_DIR Bit 1, COIL_POS_CMD Bit 2, COIL_PWR_CMD Bit 3, COIL_POS_ACCEPT Bit 4, COIL_PWR_FLASH
60	(hi-word of P6)					
61	POS6 / P7, 32-bit					
62	(hi-word of P7)					
63	POS7 / P8, 32-bit					
64	(hi-word of P8)					
65	VEL0 / V1					
66	VEL1 / V2					
67	VEL2 / V3					
68	VEL3 / V4					
69	VEL4 / V5					
70	VEL5 / V6					
71	VEL6 / V7					
72	VEL7 / V8					
73	ACC0 / A1					
74	ACC1 / A2					
75	ACC2 / A3					
76	ACC3 / A4					
77	TQ0 / T1					
78	TQ1 / T2					
79	TQ2 / T3					
80	TQ3 / T4					

TT1524GB

9.2 Motor registers MAC050 - 141

Reg. Nr.	Firmware / MacRegIo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
81	LOAD0 / L1					
82	LOAD1 / L2					
83	LOAD2 / L3					
84	LOAD3 / L4					
85	ZERO0 / Z1					
86	ZERO1 / Z2					
87	ZERO2 / Z3					
88	ZERO3 / Z4					
89	KFF3					
90	KFF2					
91	KFF1					
92	KFF0					
93	KVFX4					
94	KVFX3					
95	KVFX2					
96	KVFX1					
97	KVFY3					
98	KVFY2					
99	KVFY1					
100	KVFY0					
101	GEARB					
102	KVB3					
103	KVB2					
104	KVB1					
105	KVB0					
106	KIFX2					
107	KIFX1					
108	KIFY1					
109	KIFY0					
110	KIB1					
111	KIB0					
112	SAMPLE1					
113	SAMPLE2					
114	SAMPLE3					
115	SAMPLE4					
116	REC_CNT					
117	FNC_OUT					
118	FF_OUT					
119	VB_OUT					
120	V_EXT					Velocity of external encoder (Pulse In) in counts per sample
121	VF_OUT					
122	ANINP					
123	ANINP_OFFSET					
124	ELDEGN_OFFSET					
125	ELDEGP_OFFSET					
126	PHASE_COMP					
127	AMPLITUDE					
128	MAN_I_NOM					
129	MAN_ALPHA					
130	UMEAS					
131	I_NOM					
132	PHI_SOLL					
133	IA_SOLL					
134	IB_SOLL					
135	IC_SOLL					
136	IX_SELECT					
137	IA_IST					
138	IB_IST					
139	IC_IST					
140	IA_OFFSET					
141	IB_OFFSET					
142	IC_OFFSET					

TT1525GB

9.2 Motor registers MAC050 - 141

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
143	ELDEG_IST					
144	V_ELDEG					
145	UA_VAL					
146	UB_VAL					
147	UC_VAL					
148	KIA					
149	KIB					
150	KIC					
151	U_SUPPLY					
152	MIN_U_SUP					
153	MOTORTYPE					
154	SERIALNUMBER, 32-bit					
155	(hi-word of SERIALNUMBER)					
156	MYADDR					
157	HWVERSION					
158	CHECKSUM, 32-bit					
159	(hi-word of CHECKSUM)					
160	UV_HANDLE					Bit 0, SET_UV_ERR Bit 1, UV_GO_PASSIVE Bit 2, unused Bit 3, UV_VSOLL0
161	INV_OUTPUT					Bit 0, INV_INPOSOUT Bit 1, INV_ERROROUT Bit 2, INVROTDIR Bit 3, O1USERCTRL Bit 4, O2USERCTRL
162	INDEX_OFFSET					
163	P_NEW, 32-bit					
164	(hi-word of P_NEW)					
165	FILTERID, 32-bit					
166	(hi-word of FILTERID)					
167	HARDWARELIM					Bit 0, HW_PLIM_NEG Bit 1, HW_PLIM_POS Bit 2, HW_PLIM_IN1 Bit 3, HW_PLIM_IN2 Bit 4, HW_PLIM_IN3 Bit 5, HW_PLIM_IN4 Bit 6, HW_PLIM_IN5 Bit 7, HW_PLIM_IN6 Bit 8, HW_PLIM_ANINP
168	HOMING_DONE					Bit-0 set every time a zero search has completed. Not cleared by firmware, except after reset.

TT1526GB

9.2 Motor registers MAC050 - 141

Reg. Nr.	Firmware / MacRegIo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
169	GROUP_ID					
170	GROUP_SEQ					
171	MONITOR_CMP					
172	MONITOR_REG1					
173	MONITOR_REG2					
174	MONITOR_ACT					
175	MONITOR_SRC					
176	MONITOR_DST					
177	MONITOR_SAV					
178	SSI_BITS1					Bit 0, SSI_ENABLE Bit 1, SSI_DIR Bit 2, SSI_POS_SYNC Bit 3, SSI_RESET Bit 4, SSI_NOCHECK Bit 15, SSI_ERROR_CNTL
179	OUTPUT_CTRL					Bit 0, OUTPUT_O1 Bit 1, OUTPUT_O2
180	SETUP_BITS					Bit 0, POWERSAVE_ENABLED
181	V_IST_MAX					
182	UART1_SETUP		0, 1, 2			Selects what protocol to run on the RS422 lines that can be used for Pulse In, Pulse Out or Serial Data. The selection in this register is used only if the lowest two bits in Reg36, CNTRL_BITS are set to Serial Data. Values of Reg182, UART1_SETUP: 0: Autodetect incoming 1 Megabit Modbus telegrams for a few seconds after startup. Stay in Modbus if any valid Modbus telegrams detected, else switch to 19200 baud FastMac and stay in Fastmac. 1: Run the FastMac protocol at 19200 baud from the beginning and stay in FastMac. 2-65535: Run 1 Megabit/s Modbus from the beginning and stay in Modbus.
183	STATUS_BITS					
184	MODE0 / M1					
185	MODE1 / M2					
186	MODE2 / M3					
187	MODE3 / M4					
188	HWI0, 32-bit					
189	(hi-word of HWI0)					
190	HWI1, 32-bit					
191	(hi-word of HWI1)					
192	HWI2, 32-bit					
193	(hi-word of HWI2)					
194	HWI3, 32-bit					
195	(hi-word of HWI3)					
196	HWI4, 32-bit					
197	(hi-word of HWI4)					

TT1527GB

9.2 Motor registers MAC050 - 141

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
198	HWI5, 32-bit					
199	(hi-word of HWI5)					
200	HWI6, 32-bit					
201	(hi-word of HWI6)					
202	HWI7, 32-bit					
203	(hi-word of HWI7)					
204	-					Reserved for future purposes
205	-					Reserved for future purposes
206	-					Reserved for future purposes
207	-					Reserved for future purposes
208	-					Reserved for future purposes
209	-					Reserved for future purposes
210	-					Reserved for future purposes
211	COMMAND					
212	FIELDBUS_ADDR					
213	FIELDBUS_SPEED					
214	-					Reserved for future purposes
215	-					Reserved for future purposes
216	-					Reserved for future purposes
217	-					Reserved for future purposes
218	-					Reserved for future purposes
219	-					Reserved for future purposes
220	-					Reserved for future purposes
221	-					Reserved for future purposes
222	-					Reserved for future purposes
223	-					Reserved for future purposes
224	-					Reserved for future purposes
225	-					Reserved for future purposes
226	-					Reserved for future purposes
227	-					Reserved for future purposes
228	-					Reserved for future purposes
229	-					Reserved for future purposes
230	-					Reserved for future purposes
231	-					Reserved for future purposes
232	-					Reserved for future purposes
233	-					Reserved for future purposes
234	-					Reserved for future purposes
235	-					Reserved for future purposes
236	-					Reserved for future purposes
237	-					Reserved for future purposes
238	-					Reserved for future purposes
239	-					Reserved for future purposes
240	-					Reserved for future purposes
241	-					Reserved for future purposes
242	-					Reserved for future purposes
243	-					Reserved for future purposes
244	-					Reserved for future purposes
245	-					Reserved for future purposes
246	-					Reserved for future purposes
247	-					Reserved for future purposes
248	-					Reserved for future purposes
249	-					Reserved for future purposes
250	-					Reserved for future purposes
251	-					Reserved for future purposes
252	-					Reserved for future purposes
253	-					Reserved for future purposes
254	-					Reserved for future purposes

TT1528GB

9.3 Motor registers MAC400 - 3000

9.3.1 Register list for MAC400, 800, 1500 and 3000

The following list is only valid for the MAC400 to MAC3000 motors.

Please notice: At the Ethernet modules all registers is transmitted as 32 bit, some of them originally derive from 16 bit in the case of MAC050-141. In those situations it is necessary to interpret them as 16 bit to get the sign correct.

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range / Default	Size / Access	Unit	Description
0	N/A	N/A	N/A	N/A	N/A	Dummy register, do not use.
1	PROG_VERSION	Displayed on bottom right status line.	-	- / R	-	Firmware version
2	MODE_REG	Startup mode / Change actual mode	0..25, 256, 257, 258 / 0 (passive)	Word / RW	-	The actual operating mode of the drive. In general, the motor will either be passive, attempt to reach a certain position, attempt to maintain a constant velocity or attempt to produce a constant torque. The various modes define the main type of operation as well as what determines the setpoint for that operation. The special cases 256..258 are used to perform a few special operations on the entire set of registers. Supported values are: 0 = **Passive mode**. The axis is not controlled by the drive, and can easily be moved by hand or external mechanics. 1 = **Velocity mode**. The drive will attempt to run the motor at a constant velocity selected by Reg5, V_SOLL, without violating the maximum torque or acceleration. 2 = **Position mode**. The drive will at all times attempt to move the actual motor position to the position selected by Reg3, P_SOLL, without violating the maximum velocity, torque or acceleration. 3 = Gear Position mode. 4 = Analogue torque mode. 5 = Analogue velocity mode. 6 = Analog Velocity Gear mode. 7 = Manual current mode. 8 = Step response test mode. 9 = Internal test mode. 10 = Brake mode. 11 = Stop mode. 12 = Torque based zero search mode. 13 = Forward/only zero search mode. 14 = Forward+backward zero search mode. 15 = Safe mode. 16 = Analogue velocity with deadband mode. 17 = Velocity limited Analog Torque mode. 18 = Analogue gear mode. 19 = Coil mode. 20 = Analogue bi-position mode. 21 = Analogue to position mode. 22 = Internal test mode. 23 = Internal test mode. 24 = Gear follow mode. 25 = IHOME mode. 256: 257: 258:

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
3	P_SOLL	Max Velocity		Word / RW	Encoder counts	The target position that the drive will attempt to reach in position related modes.
4	P_NEW	*(not present)*	±2^31 / 0	Word / RW	Encoder counts	Used to update both P_IST and P_SOLL in a single atomic operation to prevent motor movements during the change. P_NEW holds either an absolute position or a relative position. After writing a value to P_NEW, update both bits 8 and 6 in Reg36, CNTRL_BITS. Bit 8, SYNCPOSREL, will select a relative position update when set or an absolute update when cleared. Setting bit 6, SYNCPOSMAN, executes the P_IST+P_SOLL update, that is, either both are set equal to P_NEW, or P_NEW is added to both, using signed addition. P_FUNC is updated accordingly. The undocumented FastMac commands 23 and 24 can also be used to set these bits and perform the same absolute and relative updates. This is useful for expanding the logical position range beyond +/- 2^31.
5	V_SOLL	Max Velocity	Na / 277(100RPM)	Word / RW		Desired velocity 1 RPM=2.77056 counts/sample. Example: To obtain 100 RPM, V_SOLL must be set to 277.
6	A_SOLL	Acceleration	na / 18	Word / RW	Cnt's/ Sample²	The desired nominal acceleration. 1000 RPM/s = 3.598133 counts/Sample² Example: To obtain 100000 RPM/s, A_SOLL must be set to 360.
7	T_SOLL	Torque	0-1023 / 1023(300%)	Word / RW	-	The maximum torque that the drive is allowed to use. The value 1023 corresponds to 300% of nominal load, and is the absolute maximum peak torque allowed. The value 341 gives 100% (nominal load).
8	P_FNC			Word / RW	Encoder counts	
9	INDEX_OFFSET	*(not present)*		Word / RW	Encoder counts	Updated after a Zero Search to show at what single-turn encoder position the zero point was detected. This is used by MacTalk on the Test tab to show if the zero search resulted in a valid zero position.
10	P_IST	Actual Position	±2^31 / 0	Word / RW	Encoder counts	The actual motor position measured by the internal encoder. Updated every 1.3ms (or every 2.6 ms with Reg157, OUTLOPDIV=2) Note that this register is maintained incrementally, which means that the user can update it to offset the working range. When updating when the drive is not in Passive mode, P_IST and P_SOLL should be updated together in an atomic operation, using Reg4, P_NEW, or other special measures. Also note that the firmware will change this register after a zero search operation has completed.
11	V_IST_16	Actual Velocity	Na / 0	Word / R	Enc.cnt's/ Sample/16	V_IST (actual velocity) measured over 16 samples. Same unit as V_SOLL (register 5).
12	V_IST	*(not present)*	Na / 0	Word / R	Enc.cnt's/ Sample	Actual velocity. 1RPM=0.17316 counts/sample.
13	KVOUT	Load	Na / 65536(1.0)	Fixed16 / RW	-	Must be set to the ratio between the total inertia driven by the motor relative to the motors own rotor inertia. So for at motor shaft that is not mechanically connected to anything, this value should be 1.0. The load factor is perhaps the single most important value of the filter setup. Always try to set this right before experimenting with filter setups.

TT1501GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
14	GEARF1	Gear factor Input	Na / 2000	Word / RW	-	The nominator used to scale / gear pulses from an external encoder/source. Used in gear modes.
15	GEARF2	Gear Output	Na / 500	Word / RW	-	The denominator used to scale / gear pulses from an external encoder/source. Used in gear modes.
16	I2T	Motor Load (mean)	Na / 0	Word / R	-	The calculated power dissipated in the motor, and thus an approximated value for the rise in temperature inside the physical motor. See also I2TLIM (Reg 17). MacTalk value is calculated as **[%]=I2T/I2TLIM x 100**
17	I2TLIM	*(not present)*	Na / 100000	Word / R	-	The limit for the value of Reg16, I2T, where bit 0, I2T_ERR, in Reg35, ERR_STAT will be set and the motor will change into passive mode.
18	UIT	Regen-Rative Load	Na / 0	Word / R		The calculated power dissipated in the internal power dump/brake resistors, and thus a way to estimate their rise in temperature. See also UITLIM (Reg 19) MacTalk value is calculated as **[%]=UIT/UITLIM x 100**
19	UITLIM	*(not present)*	Na / 2322	Word / R		The limit for the value of Reg18, UIT, where bit 3, UIT_ERR, in Reg35, ERR_STAT will be set and the motor will change into passive mode.
20	FLWERR		Na / 0	Word / RW	Encoder counts	A measure of how far the drive is from its ideal regulation goal. This value is calculated differently in the various modes, and can mean things like 'pulses from theoretical position' or 'difference in actual velocity to V_SOLL'. Contact JVL for more detailed information for specific modes.
21	U_24V		Na / 0	Word / R		The internal control voltage measured.
22	FLWERRMAX		Na / 0	Word / RW	Encoder counts	When Reg20, FLWERR, exceeds this limit, bit 1, FLW_ERR, in Reg35, ERR_STAT, is set and the motor will stop if Reg22 is non-zero. Usually this value is set experimentally to detect situations where a movement is blocked or fails.
23	UV_HANDLE	- Set error bit - Go to passive - Set velocity to 0	Na / 0	Word / RW		Bits to determine what will happen when the main supply voltage to the motor is below the threshold for motor operation. Any combination of the following bits can be set. Bit 0: Set bit 9, UV_ERR, in Reg35, ERR_STAT. Bit 1: Perform a controlled stop, then go passive. Bit 2: Set V_SOLL to zero, do not go passive.
24	FNCERR	*(not present)*	Na / 0	Word / RW	Encoder counts	Shows how much the motor is behind the ideal movement; precise operation depends on mode. When this accumulated value exceeds Reg26, FNCERRMAX, the FNC_ERR bit is set in Reg35, ERR_STAT and the motor will stop.
25	P_IST_TURNTAB	*(not present)*	Na / 0	Word / R	-	Displays the actual position, like P_IST, but is offset by N times the rotary table working range so P_IST_TURNTAB is always between MIN_P_IST and MAX_P_IST. Used mainly with the Rotary table option.
26	FNCERRMAX	*(not present)*	Na / 0	Word / RW	Encoder counts	The limit used with Reg24, FNCERR.
27	TURNTAB_COUNT	*(not present)*	Na / 0	Word / RW	-	Holds a count of the number of times the value of Reg25, P_IST, wraps around one of its limits, MIN_P_IST or MAX_P_IST. Used only with the Rotary table option. Counts up or down depending on the direction of the wrap around.
28	MIN_P_IST	*(not present)*	Na / 0	Word / RW	Encoder counts	Used to define and enable the minimum software position limit, so the motor will stop (and enter passive mode) if the value of P_IST (the actual position) gets below this value. If MIN_P_IST is zero, the low position limit will not be enabled.
29	DEGC	*(not present)*	Na / 0	Word / R	-	The temperature measured inside the drive.

TT1502GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
30	MAX_P_IST	*(not present)*	Na / 0	Word / RW	Encoder counts	Used to define and enable the maximum software position limit, so the motor will stop (and enter passive mode) if the value of P_IST (the actual position) gets above this value. If MAX_P_IST is zero, the high position limit will not be enabled.
31	DEGCMAX	*(not present)*	Na / 690(84'C)	Word / R	-	The maximum value of Reg29, DEGC, before the motor will set the Temperature error bit in ERR_STAT and change into Passive mode.
32	ACC_EMERG	*(not present)*	Na / 0	Word / RW	-	Acceleration to use during emergency stops.
33	INPOSWIN	*(not present)*	Na / 100	Word / RW	Encoder counts	The value of this parameter depends on the operating mode. In all cases it helps to define when the motor is InPosition and thus will set the InPosition bit in the ERR_STAT register. For normal Position related modes, the motor is considered to be in position when the actual position is less than INPOSWIN encoder counts away from its target position P_SOLL and have been detected to be so at least INPOSCNT times. For Velocity related modes, the concept of InPosition will instead mean AtVelocity and work in a similar way that the actual velocity V_IST must have been measured INPOSCNT consecutive times to be within INPOSWIN counts/sample before the InPosition bit is set in Reg35, ERR_STAT.
34	INPOSCNT	*(not present)*	Na / 3	Word / RW	-	The number of consecutive times the In Position condition must have been met before the InPosition bit is set in ERR_STAT. See description above for INPOSWIN.
35	ERR_STAT	*(not present)*	Na / 0	Word / RW	-	Bit 0, I2T_ERR Bit 1, FLW_ERR Bit 2, FNC_ERR Bit 3, UIT_ERR Bit 4, IN_POS Bit 5, ACC_FLAG Bit 6, DEC_FLAG Bit 7 PLIM_ERR Bit 8, DEGC_ERR Bit 9, UV_ERR Bit 10, UV_DETECT Bit 11, OV_ERR Bit 12, IPEAK_ERR Bit 13, SPEED_ERR Bit 14, DIS_P_LIM Bit 15, INDEX_ERR Bit 16, OLDFILTERR Bit 17, U24V_ERR Bit 18, SHORT_CIRC Bit 19, VAC_ON Bit 20, PWM_LOCKED A critical error has occurred that makes further motor operation too unsafe to continue. The motor must be reset to clear this error. The cause of this error is one or more of bits IPEAK_ERR, INDEX_ERR, OLDFILTER, U24V_ERR. At least one of these bits will be set when PWM_LOCKED is set. Bit 21, COMM_ERR Communications error (master or slave timeout with Modbus-Gear mode). Bit 22, CURLOOP_ERR Less than 2 mA was detected on the 4-20 mA input on the MAC00-P4/P5 module for more than 100 ms Bit 23, SLAVE_ERR One or more error bits were set in an ERR_STAT reading from the Modbus slave or COMM_ERR Bit 24, ANY_ERR single bit = (ERR_STAT and ALL_ERROR_BITS) != 0)

TT1503GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
36	CNTRL_BITS	*(not present)*	Na / 32	Word / RW	-	Bit 0, RECORDBIT Bit 1, REWINDBIT Bit 2, RECINNERBIT Bit 3, RELPOSPSOLL Bit 4, RELPOSPFNC Bit 5, SYNCPOSAUTO Bit 6, SYNCPOSMAN Bit 7, MAN_NO_BRAKE Bit 8, SYNCPOSREL Bit 9, INDEX_HOME Bit10, FWTRIGBITS When set, use the advanced sampling with firmware trigger conditions - when 0, use backwards compatible sampling Bit 11, SAMPLING_BIT Set when sampling is active after trigger has been detected Bit 12, TRIGGER_ARMED_BIT Set when sampling is active but trigger has not been detected yet Bit 13, ADVSAMPLE_BIT If set, enables div-shift, min/max/avg + bitfield sampling.
37	START_MODE	*(not present but is preset as function of the mode register)*	Na / 0	Word / RW	-	Determines in what mode the motor should start after power on and after a Zero Search. This register works closely together with Reg2, MODE_REG. Bits [31:16] are reserved. Bits [15:8] are used to select the type of zero search to perform when the FastMac command (16 + 96) is received. This should be one of 12, 13, 14, or zero. Bits [7:0] select the value to transfer to Reg2, MODE_REG at motor power up and after a zero search has completed. If bits [15:8] are non-zero the motor will remain in Passive mode at power up regardless of the value in bits [7:0]. The intention is then to wait for a FastMac command 16 + 96. It is also possible to simply write a new value to Reg2, MODE_REG to change mode.
38	P_HOME	Zero search position	Na / -10000	Word / RW	Encoder counts	The offset value to use to adjust P_IST at the end of a Zero Search.
39	HW_SETUP	*(not present)*	Na / 9	Word / RW	-	Bit 0, DIRAWR Bit 1, DIRBWR Bit 2, PULSEOUT Bit 3, XSEL1 Bit 4, XPRINP Pulse/Direction or Quadrature input type. Bit 5, NOFILT Disable lowpass filtering of external encoder pulses. Bit 6, INVXDIR Bit 7, INVROTDIR Bit 8, USER_INPOS Bit 9, USER_ERROR Error output pin is controlled by the user via RegXX Bit 10, INV_INPOS_OUT Bit 11, INV_ERROR_OUT Bit 12, CMP_ERROR_OUT If set, OUT2_PIN is controlled by (P_IST > CMP_POS0) (continued next page)

TT1504GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
39	HW_SETUP (continued from last page)	*(not present)*	Na / 9	Word / RW	-	Bit 13, PULSE_8000 If set, rescale the 8192 encoder pulses to 8000 for MAC800 compatibility and better Vel-filter performance Bits 14..15: reserved Bit 16, DIRCDWR Direction signal for the MultiFuncIo2 A channel (or both A and B?) Bit 17, SELINDEX Not used - prepared to select between encoder A or Index signal -> MultF. Bit 18, ALWAYS_COOL Bit 19, POSITION_CAPTURE_UP Used to enable SW position capture based on analogue input rising edge Bit 20, POSITION_CAPTURE_DN Used to enable SW position capture based on analogue input falling edge Bit 21, PULSE_8000 If set, rescale the 8192 encoder pulses to 8000 for MAC800 compatibility and better Vel-filter performance Bit 22, ENC_SCALING Reserved for freely selectable encoder scaling. Bit 23, SBUF_2048 Set to use a sample buffer length of 2048. Use 512 if not set (backwards compatible).
40	V_HOME	*(not present)*	Na / -138	Word / RW	-	Velocity to use during a zero search operation (Homing operation). After the operation has completed, the drive will go back to using the regular V_SOLL.
41	T_HOME	*(not present)*	Na / 341	Word / RW	-	Torque to use during a zero search operation (Homing operation). After the operation has completed, the drive will go back to using the regular T_SOLL.
42	HOME_MODE	*(not present)*	Na / 0	Word / RW	-	Defines if the motor should start a zero search immediately after start up, as well as the type of zero search to perform when a FastMac command is received. Bits 7..0 define the zero search mode the motor should start up in. If this value is zero, the motor will not perform a zero search at startup, but will start up in the mode selected by Reg37, START_MODE. See bits 15..8 below for an exception! Bits 15..8 define what mode the motor will set when it receives a FastMac command (96+16). NOTE that if all these bits are non-zero the motor will start up in passive mode instead of starting in START_MODE! Bit 16 is set after a zero search has completed, and can thus be used to test if the motor has performed a zero search at least once after +24V was last turned on. After a zero search has completed, the motor will always change into the mode defined by Reg17, START_MODE (unless an error occurs that will stop the motor and set ERR_STAT bit(s)).
43	P_REG_P	*(not present)*	0-8 / 0	Word / RW	-	When set to 1..8, copies one of POS0..POS7 to P_SOLL, then resets to 0
44	V_REG_P	*(not present)*	0-8 / 0	Word / RW	-	When set to 1..8, copies one of VEL0..VEL7 to V_SOLL, then resets to 0
45	A_REG_P	*(not present)*	0-4 / 0	Word / RW	-	When set to 1..4, copies one of ACC0..ACC3 to A_SOLL, then resets to 0
46	T_REG_P	*(not present)*	0-4 / 0	Word / RW	-	When set to 1..4, copies one of TQ0..TQ3 to T_SOLL, then resets to 0
47	L_REG_P	*(not present)*	0-4 / 0	Word / RW	-	When set to 1..4, copies one of LOAD0..LOAD3 to KVOUT then resets to 0
48	Z_REG_P	*(not present)*	0-4 / 0	Word / RW	-	When set to 1..4, copies one of ZERO0..ZERO3 to INPOSWIN, then resets to 0

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
49	POS0	Position1 (P1)	Na / 0	Word / RW	-	
50	CAPCOM0	*(not present)*	Na / 0	Word / RW	-	
51	POS1	Position2 (P2)	Na / 0	Word / RW	-	
52	CAPCOM1	*(not present)*	Na / 0	Word / RW	-	
53	POS2	Position3 (P3)	Na / 0	Word / RW	-	
54	CAPCOM2	*(not present)*	Na / 0	Word / RW	-	
55	POS3	Position4 (P4)	Na / 0	Word / RW	-	
56	CAPCOM3	*(not present)*	Na / 0	Word / RW	-	
57	POS4	Position5 (P5)	Na / 0	Word / RW	-	
58	CAPCOM4	*(not present)*	Na / 0	Word / RW	-	
59	POS5	Position6 (P6)	Na / 0	Word / RW	-	
60	CAPCOM5	*(not present)*	Na / 0	Word / RW	-	
61	POS6	Position7 (P7)	Na / 0	Word / RW	-	
62	CAPCOM6	*(not present)*	Na / 0	Word / RW	-	
63	POS7	Position8 (P8)	Na / 0	Word / RW	-	
64	CAPCOM7	*(not present)*	Na / 0	Word / RW	-	
65	VEL0	Velocity 1 (V1)	Na / 277(100RPM)	Word / RW	-	Velocity register V1. Used with the fastmac protocol or by the MAC00-R1/3/4 nanoPLC module. See also V_SOLL (register 5) which have the same scaling.
66	VEL1	Velocity 2 (V2)	Na / 277(100RPM)	Word / RW	-	Velocity register V8 - see also register 65.
67	VEL2	Velocity 3 (V3)	Na / 277(100RPM)	Word / RW	-	Velocity register V8 - see also register 65.
68	VEL3	Velocity 4 (V4)	Na / 277(100RPM)	Word / RW	-	Velocity register V8 - see also register 65.
69	VEL4	Velocity 5 (V5)	Na / 277(100RPM)	Word / RW	-	Velocity register V8 - see also register 65.
70	VEL5	Velocity 6 (V6)	Na / 277(100RPM)	Word / RW	-	Velocity register V8 - see also register 65.
71	VEL6	Velocity 7 (V7)	Na / 277(100RPM)	Word / RW	-	Velocity register V8 - see also register 65.
72	VEL7	Velocity 8 (V8)	Na / 277(100RPM)	Word / RW	-	Velocity register V8 - see also register 65.

TT1506GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
73	ACC0	*(not present)*	Na / 18(5003RPM/s²)	Word / RW	Enc.cnt's Per sample²	
74	ACC1	*(not present)*	Na / 18(5003RPM/s²)	Word / RW	Enc.cnt's Per sample²	
75	ACC2	*(not present)*	Na / 18(5003RPM/s²)	Word / RW	Enc.cnt's Per sample²	
76	ACC3	*(not present)*	Na / 18(5003RPM/s²)	Word / RW	Enc.cnt's Per sample²	
77	TQ0	Torque 1 (T1)	Na / 1023(300%)	Word / RW		Torque register T1. Used with the fastmac protocol or by the MAC00-R1/3/4 nanoPLC module. See also T_SOLL (register 7)
78	TQ1	Torque 2 (T2)	Na / 1023(300%)	Word / RW	-	Torque register T2 - see also register 77.
79	TQ2	Torque 3 (T3)	Na / 1023(300%)	Word / RW	-	Torque register T2 - see also register 77.
80	TQ3	Torque 4 (T4)	Na / 1023(300%)	Word / RW	-	Torque register T2 - see also register 77.
81	LOAD0	Load 1 (L1)	Na / 0	Word / RW	-	
82	LOAD1	Load 2 (L2)	Na / 0	Word / RW	-	
83	LOAD2	Load 3 (L3)	Na / 0	Word / RW	-	
84	LOAD3	Load 4 (L4)	Na / 0	Word / RW	-	
85	ZERO0	*(not present)*	Na / 0	Word / RW	-	
86	ZERO1	*(not present)*	Na / 0	Word / RW	-	
87	ZERO2	*(not present)*	Na / 0	Word / RW	-	
88	ZERO3	*(not present)*	Na / 0	Word / RW	-	
89	MODE0	*(not present)*	Na / 0	Word / RW	-	
90	MODE1	*(not present)*	Na / 0	Word / RW	-	
91	MODE2	*(not present)*	Na / 0	Word / RW	-	
92	MODE3	*(not present)*	Na / 0	Word / RW	-	
93	HWI0	*(not present)*	Na / 0	Word / RW	-	**H**ard**W**are **I**nputs Regs 93-104, HWI0-11, allow the digital inputs from Reg106 to control the values of other motor registers. The most common use is to copy one of two values to a target register. This can be used to switch between two velocities, positions or modes. For instance to switch between two target positions, set Reg49, POS0 to 1000 and Reg51, POS1 to 2000 and set the motor into position mode. Then P_SOLL can be set to receive either the value 1000 or 2000 depending on the voltage on the digital input (the Input State) The copying is executed every 1.3 ms. The digital inputs can thus be considered level-triggered rather than edge-triggered. (Contrinued next page)

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
93	HWI0 (Continued from last page)	*(not present)*	Na / 0	Word / RW	-	Each of the HW!0-11 registers have the following bit fields: Bits [31:24]: Destination register used (only) when bits [3:0] equals 7. Bits [23:16]: Source register number 0..254 for DI=1 Bits [15:8]: Source register number 0..254 for DI=0 Bits [7:4]: Select digital input bit number in Reg106. Bits [3:0]: Target register selection. 0=None, 1=MODE_REG, 2=V_SOLL, 3=P_SOLL, 4=A_SOLL, 5=T_SOLL, 6=INPOSWIN, 7=Register number from bits [31:24]. When the value of bits [3:0] are one of 1..6, the two source registers are implicitly fixed to the corresponding group of register, and the value of bits [23:16] and bits [15:8] are used as an index into that group of registers. For instance if bits [3:0] equals 3, the values of bits [23:16] and bits [15:8] must be in the range 1..8 to select POS1 ..POS8 for source registers to copy into P_SOLL. When the value of bits [3:0] equals 7, the values of bits [23:16] and [15:8] hold the full register numbers in the range 1-254. For more advanced use, any of the source register or index values can be set to zero, which means DoNothing. This effectively means that in one of the Input States a source register will be copied to the target register, while in the other Input State no copying will happen so the target register will not be modified by the digital input. The 12 HWI functions are executed every 1.3 ms in the order from HWI0 to HWI11. NO other operations happen in between regardless of communications and other parallel operations. It is therefore safe to rely on stable register values and consistent digital input values during the execution of the 12 HWI functions. This implies that HWI function with higher numbers have higher priority because they are executed later, and that it is safe to change the same target register several times during the HWI evaluation. Note that each of the HWI function can use any of the digital inputs, and that more than one HWI function can use the same digital input. A typical HWI application is Jogging, where two pushbuttons connected to two separate digital inputs are used to move the motor position manually. This can be realized with a HWI setup like: HWI0 uses Digital Input 1: ON => MODE_REG=1 (velocity mode) OFF => MODE_REG=3 (gear mode) HWI1 also uses Digital Input 1: ON => V_SOLL=**+100**RPM OFF => V_SOLL = 3000 RPM HWI2 uses Digital Input 2: ON => MODE_REG=1 (velocity mode) OFF => MODE_REG=3 (gear mode) HWI3 also uses Digital Input 2: ON => V_SOLL=**-100**RPM OFF => V_SOLL = 3000 RPM This will keep the motor in Gear mode with a maximum velocity of 3000 RM when none of the pushbuttons are activated, and change to Velocity mode wit either +100 or -100 RPM as long as one of the pushbuttons are held active. In this setup Digital Input 2 will have higher priority than Digital Input 1, because it is evaluated later and overwrites V_SOLL in case both buttons are held down.

TT1508GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
94	HWI1	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
95	HWI2	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
96	HWI3	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
97	HWI4	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
98	HWI5	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
99	HWI6	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
100	HWI7	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
101	HWI8	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
102	HWI9	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
103	HWI10	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
104	HWI11	*(not present)*	Na / 0	Word / RW	-	See Reg93, HWI0, for description
105	MAC00_TYPE	*(not present)*	Na / 0	Word / RW	-	Identifies the Generation-2 module type autodetected at startup. 0 = No Gen2 module found, 1=MAC00-B41, 2=MAC00-P4 or MAC00-P5 found.
106	MAC00_1 / Digital Inputs	I/O management	Na / 0	Word / RW	-	The registers from 106 to 120 are used to support different interface modules with the Generation-2 connectors. The function of these registers will be different depending on which module is mounted in the motor. The Gen.2 module type is detected automatically by the motor at start up. Reg106, Digital inputs, is a bitmapped value where bits [15:8] show the status of hardware signals in the basic motor as described below, while bits [7:0] show the status of the digital inputs from the MAC00-B41 module. Be aware that bits [15:0] in Reg215, IO_POLARITY, can be set to invert the value of the corresponding bits [15:0] in this register. Bits [15:12] show the values of the four RS-422 signals. These are intended mostly for serial communications to some modules or to use Modbus RS485, but they can be used as digital inputs provided that the input voltage is kept within -7 to +12 volts. These are differential signals, so to use them as single-ended inputs, one of the differential lines must be kept at a constant voltage in between the high and low thresholds for the single-ended line. At the time of this writing, bits [15:12] are supported on MAC400, but not yet on MAC800. Bit 15: Multifunction 1, channel B Bit 14: Multifunction 1, channel A Bit 13: Multifunction 2, channel B Bit 12: Multifunction 2, channel A Bits [10:8] show the status of the analogue inputs ANINP2, ANINP1 and ANINP. Status will be high (logic 1) when the value of the analogue line is above 5.0 volts. This threshold can be adjusted by modifying the corresponding ANINPx_OFFSET registers. This way it is possible to use the analogue inputs as digital inputs with adjustable thresholds in the range -10V to +10V. Bit 10: ANINP2 (not signal conditioned) Bit 9: ANINP1 (not signal conditioned) Bit 8: ANINP (signal conditioned) To use ANINP3 (availab le on the MAC00-P4 and MAC00-P5 modules as analogue current loop 4-20 mA) use Reg222, IOSETUP to make ANINP reflect the (signal conditioned) value of this input, so the digital status will be shown in Bit 8. To use ANINP2 as a signal conditioned input, use a similar trick so IOSETUP is set to make ANINP reflect the signal conditioned value of ANINP2 in bit 8. Bits 6, 7, and 11 are unused.

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
107	MAC00_2	*(not present)*	Na / 0	Word / RW	-	Shows various status bits for the currently mounted Gen2 module. For the MAC00-B41: Bit 0: Digital Output overload. This shows the staus of the output driver chip that controls the six digital outputs. The overload status can be set if either an overcurrent condition or a too high temperature is detected. This status bit is cleared when these conditions are no longer present. Bit 1: CVO voltage detected. This bit reflects if the voltage at the CVO terminal is above a hardwired default value. CVO is the supply voltage for the digital outputs.
108	MAC00_3	*(not present)*	Na / 0	Word / RW	-	N/U
109	MAC00_4	*(not present)*	Na / 0	Word / RW	-	N/U
110	MAC00_5	*(not present)*	Na / 0	Word / RW	-	N/U
111	MAC00_6	*(not present)*	Na / 0	Word / RW	-	N/U
112	MAC00_7	*(not present)*	Na / 0	Word / RW	-	N/U
113	MAC00_8 / B41_DO / Digital outputs	I/O management	Na / 0	Word / RW	-	Bits [5:0] of this register controls the digital outputs O6..O1 on the MAC00-B41 module. Each bit that is set here will enable the corresponding PNP output. It is possible to overwrite these bits by using Registers 115-120, see below. Also Reg215, IO_POLARITY, will invert the value of these bits before there are written to the hardware.
114	MAC00_9 / B41_DOSTATUS	I/O management	Na / 0	Word / RW	-	Shows the status of each of the six digital outputs actually written to the hardware. This value will be Reg113, possibly modified by Regs115 - 120 and finally possibly having some bits inverted by Reg215.
115	MAC00_10 / B41_CONF0	*(not present)*	Na / 0	Word / RW	-	Controls **IO1** on MAC00-B41 (bit 0 in B41_DO). Each of the B41_CONF5..CONF0 registers can be used to modify the corresponding digital outputs by effectively overwriting bits [5:0] in Reg113, B41_DO. They can be set to replace the corresponding bit in B41_DO with any bit from any motor register in the range 1..254, typically status bits from Reg35, ERR_STAT, for instance bits INPOS or ANY_ERR. Bits [31:24]: reserved Bits [23:16]: Source register number, 1..254. Bits [15:5]: Reserved Bits [4:0]: Bit number in source register to use. Reg215, IO_POLARITY, will be applied after these registers to allow general inversion of each digital output bit.
116	MAC00_11 / B41_CONF1	*(not present)*	Na / 0	Word / RW	-	Controls **IO2** on MAC00-B41 (bit 1 in B41_DO). See Reg115, B41_CONF0 for description.
117	MAC00_12 / B41_CONF2	*(not present)*	Na / 0	Word / RW	-	Controls **IO3** on MAC00-B41 (bit 2 in B41_DO). See Reg115, B41_CONF0 for description.
118	MAC00_13 / B41_CONF3	*(not present)*	Na / 0	Word / RW	-	Controls **IO4** on MAC00-B41 (bit 3 in B41_DO). See Reg115, B41_CONF0 for description.
119	MAC00_14 / B41_CONF4	*(not present)*	Na / 0	Word / RW	-	Controls **IO5** on MAC00-B41 (bit 4 in B41_DO). See Reg115, B41_CONF0 for description.
120	MAC00_15 / B41_CONF5	*(not present)*	Na / 0	Word / RW	-	Controls **IO6** on MAC00-B41 (bit 5 in B41_DO). See Reg115, B41_CONF0 for description.

TT1510GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
121	KFF5	KFF5	Na / 0	Word / RW	-	Filter coefficients used by the velocity and position regulator loops. These values should be loaded only from MacTalk, and not modified by the user, since this can have dangerous effects.
122	KFF4	KFF4	Na / 0	Word / RW	-	
123	KFF3	KFF3	Na / 0	Word / RW	-	
124	KFF2	KFF2	Na / 0	Word / RW	-	
125	KFF1	KFF1	Na / 0	Word / RW	-	
126	KFF0	KFF0	Na / 0	Word / RW	-	
127	KVFX6	*(not present)*	Na / 0	Word / RW	-	
128	KVFX5	*(not present)*	Na / 0	Word / RW	-	
129	KVFX4	*(not present)*	Na / 0	Word / RW	-	
130	KVFX3	*(not present)*	Na / 0	Word / RW	-	
131	KVFX2	*(not present)*	Na / 0	Word / RW	-	
132	KVFX1	*(not present)*	Na / 0	Word / RW	-	
133	KVFY5	*(not present)*	Na / 0	Word / RW	-	
134	KVFY4	*(not present)*	Na / 0	Word / RW	-	
135	KVFY3	*(not present)*	Na / 0	Word / RW	-	
136	KVFY2	*(not present)*	Na / 0	Word / RW	-	
137	KVFY1	*(not present)*	Na / 0	Word / RW	-	
138	KVFY	*(not present)*	Na / 0	Word / RW	-	
139	KVB4	*(not present)*	Na / 0	Word / RW	-	
140	KVB3	*(not present)*	Na / 0	Word / RW	-	
141	KVB2	*(not present)*	Na / 0	Word / RW	-	
142	KVB1	*(not present)*	Na / 0	Word / RW	-	
143	KVB0	*(not present)*	Na / 0	Word / RW	-	
144	KIFX2	*(not present)*	Na / 0	Word / R	-	Filter coefficients used by the current loop for low-level control f the phase currents. These values are fixed and should not be modified by the user.
145	KIFX1	*(not present)*	Na / 0	Word / R	-	
146	KIFY1	*(not present)*	Na / 0	Word / R	-	
147	KIFY0	*(not present)*	Na / 0	Word / R	-	
148	KIB1	*(not present)*	Na / 0	Word / R	-	
149	KIB0	*(not present)*	Na / 0	Word / R	-	

TT1511GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
150	<reserved>	*(not present)*	-			
151	<reserved>	*(not present)*	-			
152	<reserved>	*(not present)*	-			
153	<reserved>	*(not present)*	-			
154	<reserved>	*(not present)*	-			
155	ID_RESERVED	*(not present)*	-			<reserved>
156	S_ORDER	*(not present)*	Na / 0	Word / RW	-	An S-profile can be used to modify/smooth the acceleration at the beginning and end of a change in velocity. This is useful to prevent overshoot. The value of zero disables the S-profile so the normal A_SOLL is used. Values 1..8 can be used to select a progressively smoother S-profile, with 8 being the smoothest (and slowest). The value of S_ORDER may not be changed unless the motor is in Passive mode (MODE_REG=0).
157	OUTLOOPDIV	*(not present)*	Na / 0	Word / RW	-	Divider value for the velocity loop. With the standard value of 1, the velocity loop is recalculated every 1.3 ms. With a value of 2, the loop is recalculated every 2.6 ms, which can give better performance for slow movements and/or large inertia. It is absolutely necessary to use a different set of filters in Regs121-142 when changing this value. To change this value from MacTalk, and gain access to the extended filters, open the Filter Setup window, then hold down both the Control and Shift keys and double-click on the text 'More' to the left of the 'Stability' slider (at the green end). After entering the correct password, Sample Frequency can be selected and MacTalk will use the appropriate filter set. Note that the units of all velocity-related register, measured in counts/sample will now be doubled, and all acceleration-related registers, measured in Counts/sample^2, will be four times larger.

TT1512GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
158	SAMPLE1	*(not present)*	Na / 0	Word / RW	-	SAMPLE1..4 controls the scope/sample function. Register number, bit field and min/max/average sample type for the first value in each sample.
159	SAMPLE2	*(not present)*	Na / 0	Word / RW	-	Register number, bit field and min/max/average sample type for the second value in each sample.
160	SAMPLE3	*(not present)*	Na / 0	Word / RW	-	Register number, bit field and min/max/average sample type for the third value in each sample.
161	SAMPLE4	*(not present)*	Na / 0	Word / RW	-	Register number, bit field and min/max/average sample type for the fourth value in each sample.
162	REC_CNT	*(not present)*	0-511 or 0..2047 / 0	Word / RW	-	Index into the sample buffer used for scope functionality. The length of the sample buffer, and thus the range of this parameter if determined by bit 23, SBUF_2048, in Reg39, HW_SETUP. See document/section “YY” for further information on the sample system.
163	V_EXT	*(not present)*	Na / 0	Word / R	-	Unscaled/Raw velocity of external encoder input in pulses per 1.3ms.
164	GV_EXT	*(not present)*	Na / 0	Word / R	-	Velocity of external encoder input V_EXT, after being scaled by the ratio GEARF1/GEARF2
165	G_FNC	*(not present)*	Na / 0	Word / R	-	
166	FNC_OUT	*(not present)*	Na / 0	Word / R	-	
167	FF_OUT	*(not present)*	Na / 0	Word / R	-	
168	VB_OUT	*(not present)*	Na / 0	Word / R	-	
169	VF_OUT	Actual torque	Na / 0	Word / RW	-	
170	ANINP	*(not present)*	Na / 0	Word / RW	-	
171	ANINP_OFFSET	*(not present)*	Na / 0	Word / RW	-	
172	ELDEG_OFFSET	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
173	PHASE_COMP	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
174	AMPLITUDE	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
175	MAN_I_NOM	*(not present)*	Na / 0	Word / RW	-	<used with motor current loop>
176	MAN_ALPHA	*(not present)*	Na / 0	Word / RW	-	<used with motor current loop>
177	UMEAS	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
178	I_NOM	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
179	PHI_SOLL	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
180	IA_SOLL	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
181	IB_SOLL	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
182	IC_SOLL	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>

TT1513GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
183	IA_IST	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
184	IB_IST	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
185	IC_IST	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
186	IA_OFFSET	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
187	IB_OFFSET	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
188	KIA	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
189	KIB	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
190	ELDEG_IST	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
191	V_ELDEG	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
192	UA_VAL	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
193	UB_VAL	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
194	UC_VAL	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
195	EMK_A	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
196	EMK_B	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
197	EMK_C	*(not present)*	Na / 0	Word / R	-	<used with motor current loop>
198	U_BUS	Bus voltage	Na / 0	Word / R	-	The actual voltage of the internal DC bus, updated every 100 us. One count corresponds to ~0.888V.
199	U_BUS_OFFSET	*(not present)*	-	Word / R	-	Factory offset used to calibrate the measurement of Reg198, U_BUS.
200	TC0_CV1	*(not present)*	Na / 0	Word / R	-	<used by JVL only to monitor internal timing´>
201	TC0_CV2	*(not present)*	Na / 0	Word / R	-	<used by JVL only to monitor internal timing´>

TT1514GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
202	MY_ADDR	*(not present)*	Na / 0	Word / RW	-	The motor address used for the MacTalk protocol. The motor will respond to telegrams with this address or the broadcast address 255. MY_ADDR can also be used for the Modbus protocol if selected in Reg213, UART1_SETUP: Further, MY_ADDR can be read and used by the fieldbus modules for CANopen, DeviceNet and Profibus to define their address on the fieldbus, if not selected by DIP-switches on the MAC00-xx module.
203	MOTOR_TYPE	*(not present)*	Na / 0	Word / R	-	Value read from factory flash memory to identify the type of motor: 12=MAC400, 13=MAC400B, 14=MAC800, 15=MAC800B.
204	SERIAL_NUMBER	*(not present)*	Na / 0	Word / R	-	Value read from factory flash memory to show the JVL serial number of the motor.
205	HW_VERSION	*(not present)*	Na / 0	Word / R	-	Bits [23:20]: Value read from factory flash memory to identify the Main version of the bootloader. Bits [19:16]: Value read from factory flash memory to identify the Minor version of the bootloader. Bits [7:4]: Value read from factory flash memory to identify the Main version of the PCB controller board hardware. Bits [3:0]: Value read from factory flash memory to identify the Minor version of the PCB controller board hardware. The remaining bits are reserved.
206	CHKSUM	*(not present)*	Na / 0	Word / R	-	Value read from factory flash memory to show the checksums of the firmware and the bootloader.
207	USEROUTVAL	*(not present)*	Na / 0	Word / RW	-	The values of bits [1:0] are output to the standard InPosition and ErrorOut hardware signals if the corresponding bits [9:8], USER_INPOS and USER_ERROR, in Reg39, HW_SETUP are set.
208	COMM_ERRS	*(not present)*	Na / 0	Word / RW	-	Counts the number of communication errors that have occurred on the MacTalk serial interface. Errors can be framing errors and protocol data errors.
209	INDEX_IST	*(not present)*	0..8191 or 0..7999	Word / R	-	Actual single-turn position of the internal encoder, valid for both incremental and absolute encoders.
210	HW_PLIM	*(not present)*	Na / 0	Word / RW	-	Hardware position limits – used by the MAC00-FSx module.
211	COMMAND_REG	*(not present)*	Na / 0	Word / RW	-	1=Reset, 2=Save to flash and reset, 128..255 = Execute FastMac commands.
212	UART0_SETUP	MacTalk Baudrate	Na / 0	Word / RW	-	0=9600, 1=19200, 2=38400, 3=57600, 4=115200, 5=230400 baud.
213	UART1_SETUP	Serial data	Na / 0	Word / RW	-	This register selects the type of protocol to use on the Serial Data interface. See section “XX”.
214	EXTENC_BITS	*(not present)*	Na / 0	Word / RW	-	Supports setup of signals used for label dispenser functionality with the MAC00-B41 module.
215	INPUT_LEVELS	*(not present)*	Na / 0	Word / RW	-	
216	ANINP1	*(not present)*	Na / 0	Word / RW	-	
217	ANINP1_OFFSET	*(not present)*	Na / 0	Word / RW	-	
218	ANINP2	*(not present)*	Na / 0	Word / RW	-	
219	ANINP2_OFFSET	*(not present)*	Na / 0	Word / RW	-	
220	ANINP3	*(not present)*	Na / 0	Word / RW	-	
221	ANINP3_OFFSET	*(not present)*	Na / 0	Word / RW	-	

TT1515GB

9.3 Motor registers MAC400 - 3000

Reg. Nr.	Firmware / MacReglo Name	MacTalk Name	Range/ Default	Size / Access	Unit	Description
222	IOSETUP	*(not present)*	Na / 0	Word / RW	-	Selects what hardware analogue input signal that goes to the main ANINP register and controls some filtering/signal conditioning.
223	ANOUT1	*(not present)*	Na / 0	Word / RW	-	The value written here by the user, or by the firmware, will be output to the 4-20 mA hardware output on the MAC00-P5/P4 modules.
224	ANOUT1_OFFSET	*(not present)*	Na / 0	Word / RW	-	Offset that is added to ANOUT1 before writing to hardware.
225	P_OFFSET	*(not present)*	Na / 0	Word / RW	-	Used to adjust the zero position for absolute multi-turn encoders.
226	P_MULTITURN	*(not present)*	Na / 0	Word / RW	-	The full multi-turn position read directly from the absolute encoder, if mounted.
227	AIFILT_MAXSLOPE	*(not present)*	Na / 0	Word / RW	-	
228	AIFILT_FILTFACT	*(not present)*	Na / 0	Word / RW	-	
229	P_QUICK	N/A	Na / 0	Word / RW	-	The actual position of the internal encoder. Much like P_IST, but updated every 100us. P_IST is updated only once every 1.3ms (or 2.6 ms for OUTLOOPDIV=2).
230	XREG_ADDR	*(not present)*	Na / 0	Word / RW	-	Address of extended registers, XREGs. A positive value will write the contents of Reg231, XREG_DATA, to that register. A negative value will cause the value of that XREG to be writen to XREG_DATA. After the reading or writing operation has completed, XREG_ADDR will be set to zero. The first NN XREGs are used for configuration of the switchboard for hardware signals that can be routed in several ways through the FPGA in MAC800 HW 1.8 and later or MAC400 HW1.? And later.
231	XREG_DATA	*(not present)*	Na / 0	Word / RW	-	Data to or from extended registers. See XREG_ADDR for description

TT1516GB

8 Index

8 Index