eNod4-T Ethernet

Digital Process Transmitter

Software user manual

1. ENOD4 PRODUCT RANGE

1.1. General presentation

eNod4 is a high speed digital process transmitter with programmable functions and powerful signal processing capabilities. ***eNod4*** offers operating modes for advanced process control both static and dynamic.

Quick and accurate:

- Analog to digital conversion rate up to 1920 meas/s with maximum scaled resolution of ±500 000 points.
- Digital filtering and measurement scaling.
- Measurement transmission up to 1 000 meas/s.

Easy to integrate into automated system:

- **USB**, **RS485** and **CAN** communication interfaces supporting ***ModBus RTU***, ***CANopen®*** and ***PROFIBUS-DPV1*** (depending on version) communication protocols.
- Digital Inputs/Outputs for process control.
- Setting of node number by rotary switches and communication baud rate by dip switches.
- Integrated selectable network termination resistors.
- Wiring by plug-in terminal blocs.

1.2. Versions and options

1.2.1. Versions

- Strain gauges load-cell conditioner with ***CANopen®*** and ***ModBus RTU*** communication.
- Strain gauges load-cell conditioner with ***Profibus DP-V1*** and ***ModBus RTU*** communication.
- Strain gauges load-cell conditioner with ***Modbus TCP*** and ***ModBus RTU*** communication.
- Strain gauges load-cell conditioner with ***Ethernet/IP*** and ***ModBus RTU*** communication.
- Strain gauges load-cell conditioner with ***Profinet IO*** and ***ModBus RTU*** communication.

EDS, GSD and GSDML configuration file for ***CANopen®*** can be downloaded from our web site: http://www.scaime.com

1.2.2. Options

With appropriate option the strain gauges load-cell can be exchanged with:

- 4/20mA analog signal.
- 0/10V analog signal.

1.3. Versions and options

So as to configure ***eNod4***, SCAIME provides ***eNodView*** software tool. ***eNodView*** is the software dedicated to ***eNod*** devices and digital load cell configuration from a PC. Its simple graphical interface allows accessing the whole functionalities of ***eNod4*** for a complete setting according to the application.

eNodView features and functions :

- eNod4 control from a PC
- Calibration system
- Modification/record of all parameters
- Measure acquisition with graphical display
- Numerical filters simulation
- Frequential analysis FFT
- Process control
- Network parameters

eNodView software is available in English and French version and can be downloaded from our web site: http://www.scaime.com or ordered to our sales department on a CD-ROM support.

2. COMMUNICATION AND FUNCTIONING MODES

Name	Modbus address	Ethernet/IP Class/ Attribute (hex/dec)	Profinet Record Index	Profinet cyclic Req Code	Type	Access
Functioning mode / Serial protocol	0x003E	/	/	/	Uint	RW

2.1. Communication protocols Modbus RTU and SCMBus

Modbus RTU, SCMBus, and fast SCMBus communication protocols are accessible through AUX, USB or DB9 connection (depend on version).

The protocol can be changed via the « Functioning mode/ serial protocol » register (see below).

bits b9b8	Protocol
00	SCMBus
01	Modbus RTU
11	Fast SCMBus

Note: To be applied, any modification of this setting must be followed by an EEPROM back up and device reboots (hardware or software).

2.2. Functioning mode

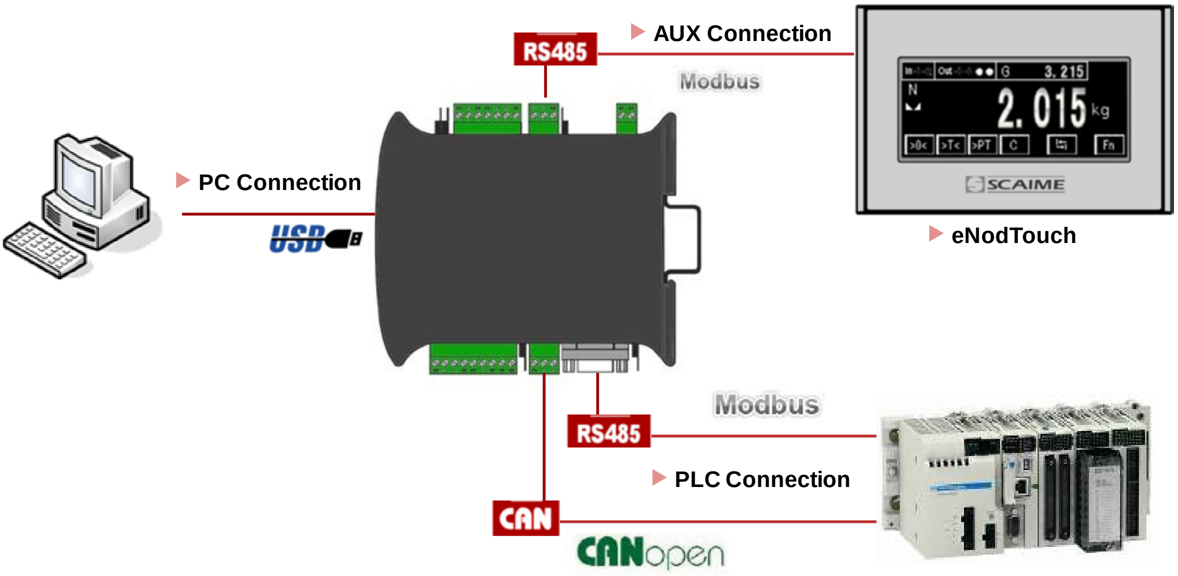
The « Functioning mode/ serial protocol » register offers the possibility to change the eNod4 application according to the following list:

bits b_1b_0	Functioning mode		
	eNod4-T	eNod4-C	eNod4-D
00	Transmitter	Transmitter	Transmitter
01	/	checkweigher transmitter on request	Dosing by filling
10	/	/	Dosing by unfilling

Note: To be applied, any modification of this setting must be followed by an EEPROM back up and device reboots (hardware or software).

2.3. Simultaneous functioning of communications

2.3.1. Standard version

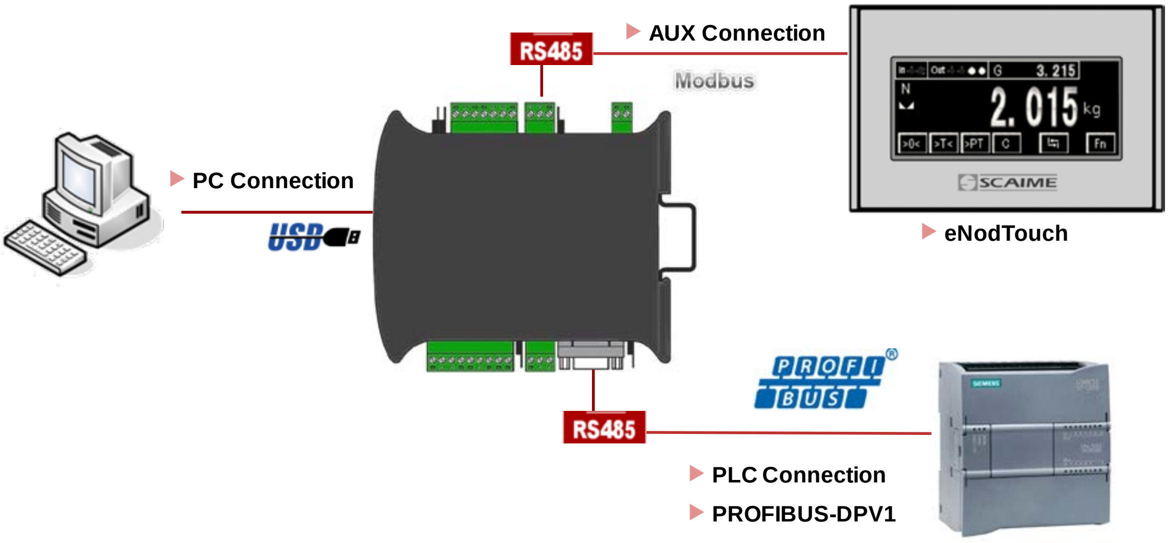


Simultaneous Communication	*RS485 PLC*	*RS485 AUX*	*CAN*
USB	*Yes**	*No*	*Yes**
RS485 PLC		*Yes*	*No*
RS485 AUX			*Yes***

(*)Simultaneous use of CAN or RS485 communication with USB port can reduce performance of this interface.

(**)In this configuration, we recommend a typical speed on AUX output of 9600 bps (Max 19200 bps)

2.3.2. Profibus version

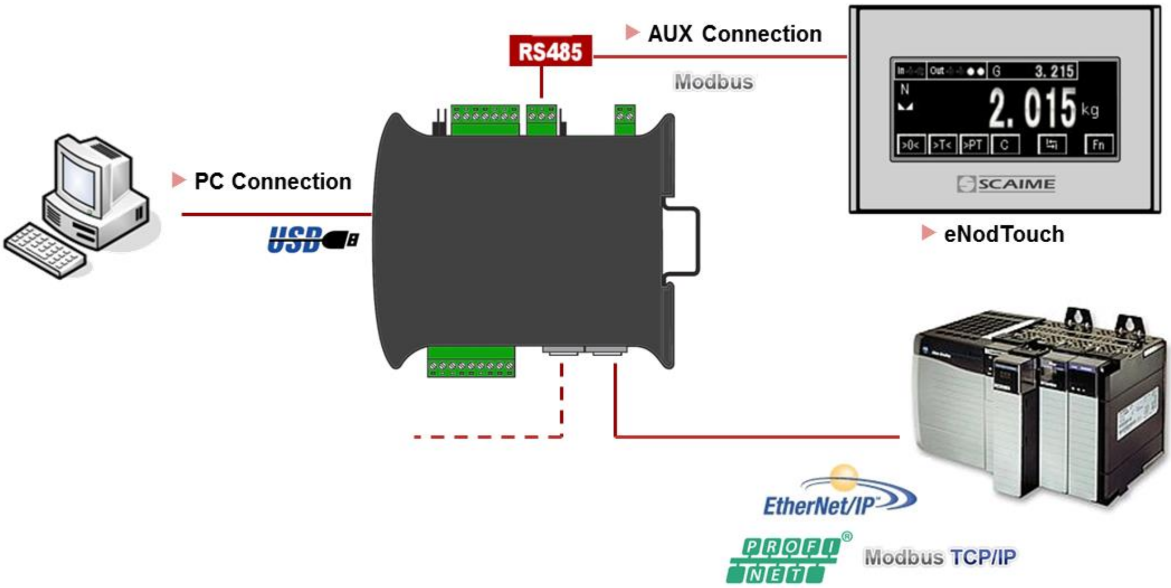


Simultaneous Communication	Profibus	RS485 AUX
USB	Yes*	No
Profibus		Yes**

(*)Simultaneous use of Profibus with USB port can reduce performance of this interface.

(**)In this configuration, we recommend a typical speed on AUX output of 9600 bps (Max 19200 bps)

2.3.3. Ethernet versions



Simultaneous Communication	Ethernet	RS485 AUX
USB	Yes*	No
Ethernet		Yes**

(*)Simultaneous use of Ethernet with USB port can reduce performance of this interface.

(**)In this configuration, we recommend a typical speed on AUX output of 9600 bps (Max 19200 bps)

3. MODBUS RTU

3.1. Physical interfaces

Modbus RTU communication protocol can be used either through ***eNod4*** USB port, AUX port or DB9 port (depend on version).

USB port behaves as a full duplex interface whereas the DB9 and AUX ports support half-duplex RS485 communication. Supported baud rates are 9600, 19200, 38400, 57600, and 115200.

For a complete description of the recommendations about ***eNod4*** RS485 connection, please refer to the user manual "characteristics and functioning" of the ***eNod4***.

Note: using ***eNod4*** through USB requires installing first the necessary USB drivers available on the website http://www.scaime.com.

3.2. Byte format

Data transmitted to ***eNod4*** thanks to Modbus RTU communication protocol must respect following format:

- 1 start bit
- 8 data bits
- no parity
- 2 stop bits

Every Modbus RTU frame is ended by a CRC-16 2-bytes code whose polynomial generator is

$$G(x) = x^{16} + x^{15} + x^{2} + 1$$

(cf. CRC-16 calculation algorithm).

3.3. Modbus RTU supported functions

As a Modbus RTU slave, ***eNod4*** supports following Modbus RTU functions:

Function	*Code*
read N registers*	*03_H / 04_H*
write 1 register*	*06_H*
write N registers*	*10_H*

** 1 register = 2 bytes, maximum admitted value for N is 30.*

Note: Broadcast addressing is not allowed by ***eNod4***.

3.4. Frames structure

During a read or write transaction, the two bytes of a register are transmitted MSB first then LSB.

If a data is coded on **4 bytes** (that means it requires two registers), **the two LSB are stored in the low address register and the two MSB are stored in the high address register.**

3.4.1. Function (03_H/04_H) – read N input registers (N = 30 max)

- request command sent to the slave :

slave address	*03_H or 04_H*	*starting register offset*	*N registers*	*CRC16*
1 byte	*1 byte*	*2 bytes*	*2 bytes*	*2 bytes*

- slave response :

slave address	*03_H or 04_H*	*NB **	*data 1*	*…*	*CRC16*
1 byte	*1 byte*	*1 byte*	*2 bytes*	*2 bytes*	*2 bytes*

* NB: number of read bytes (= N*2)

3.4.2. Function (06_H) – write single register

- request command sent to the slave :

slave address	*06_H*	*register offset*	*data*	*CRC16*
1 byte	*1 byte*	*2 bytes*	*2 bytes*	*2 bytes*

- slave response :

slave address	*06_H*	*register offset*	*data*	*CRC16*
1 byte	*1 byte*	*2 bytes*	*2 bytes*	*2 bytes*

3.4.3. Function (10_H) – preset multiple registers (N = 30 max)

- request command sent to the slave :

slave address	*10_H*	*starting register offset*	*N registers*	*NB*	*Data 1*	*…*	*CRC16*
1 byte	*1 byte*	*2 bytes*	*2 bytes*	*1 byte*	*2 bytes*	*2 bytes*	*2 bytes*

- slave response :

slave address	*10_H*	*starting register offset*	*N registers*	*CRC16*
1 byte	*1 byte*	*2 bytes*	*2 bytes*	*2 bytes*

3.4.4. Error frames

- frame format in case of a transaction error :

slave address	*function code + 80_H*	*error code*	*CRC16*
1 byte	*1 byte*	*1 byte*	*2 bytes*

- Error codes meaning :

Error code	Meaning	description
01_H	*illegal function*	*Modbus-RTU function not supported by eNod4*
02_H	*illegal data address*	*register address requested out of eNod4 register table*
03_H	*illegal data value*	*forbidden data values for the requested register*
04_H	*eNod4 not ready*	*eNod4 is not ready to answer (for example measurement request during a taring operation)*

3.5. Address and Baud rate

Address Modbus RTU	Meaning	Access	Type
0x0001	*Address and Baud rate*	*RO*	*Uint*

Reads the address and baud rate selected on the front panel via the rotary switches and dipswitches.

3.6. Product identification

Software and product versions of the ***eNod4*** are accessible via Modbus RTU.

Address Modbus RTU	Meaning	Access	Type
0x0000	*SW and product version*	*RO*	*Uint*

The 12 LSB bits define the software version (073_H = 115) and the 4 MSB bits define the product version (6_H for the ***eNod4***).

3.7. Measurement transmission

As a master/slave protocol, measurement transmission in Modbus protocol is only done on master request.

3.8. EEPROM error management

Functioning and calibration parameters are stored in EEPROM. After every reset the entireness of parameters stored in EEPROM is checked. If a default appears, measurements are set to 0xFFFF and default is pointed out in measurement status.

4. SCMBUS / FAST SCMBUS

4.1. Physical interfaces

Modbus RTU communication protocol can be used either through ***eNod4*** USB port, AUX port or DB9 port (depend on version).

USB port behaves as a full duplex interface whereas the DB9 and AUX ports support half-duplex RS485 communication. Supported baud rates are 9600, 19200, 38400, 57600, and 115200.

For a complete description of the recommendations about ***eNod4*** RS485 connexion, please refer to the user manual "characteristics and functioning" of the ***eNod4***.

Note : using ***eNod4*** through USB requires installing first the necessary USB drivers available on the website http://www.scaime.com.

4.2. SCMBus and fast SCMBus features

SCMBus and its variant fast SCMBus can be imbricate into ModBus RTU protocol if the setting *'communication protocol'* is set to SCMBus or fast SCMBus. That means that ***eNod4*** continues answering Modbus RTU frames but it also allows the device to send frames coded according to SCMBus/fast SCMBus format.

Each protocol has its advantages:

- in SCMBus measurements are transmitted as ASCII with the decimal point and the unit integrated to the frame
- fast SCMBus is dedicated to fast measurement transmission as the frames are the most compact as possible
- both protocols allow to communicate without any master request (continuous transmission or sampling triggered by a logical input)

4.3. Byte format

Data transmitted to ***eNod4*** thanks to SCMBus or fast SCMBus communication protocol must respect following format:

- 1 start bit
- 8 data bits
- no parity
- 2 stop bits

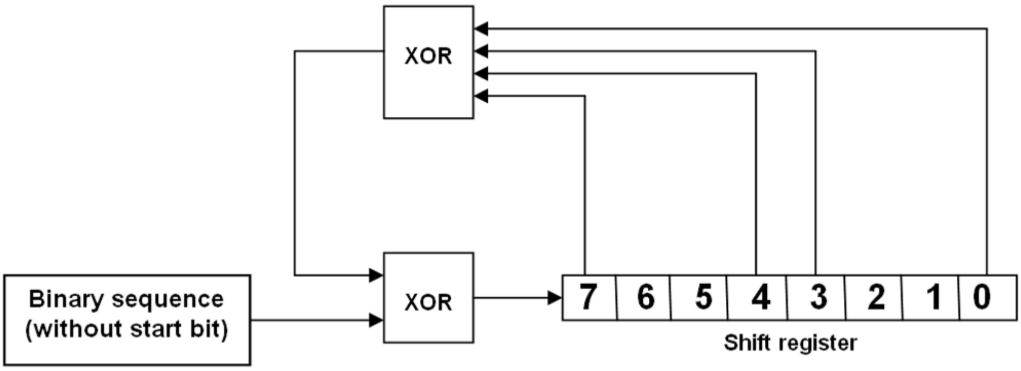
in SCMBus protocol, data is encoded as ASCII numeral characters (30_H ..... 39_H) and ASCII hexadecimal characters ($3A_H$ ..... $3F_H$).

in fast SCMBus protocol, data is encoded as signed hexadecimal (see frame structure paragraph) below.

SCMBus CRC-8 byte is generated by the following polynomial:

$$G(x) = x^{8+} x^7 + x^4 + x^3 + 1$$

The CRC-8 polynomial result can be determined by programming the algorithm corresponding to the following diagram:



Note: The frame error detection can be ignored. Value **0xFF** of the CRC-8 always is admitted by ***eNod4*** and a received frame which is ended by such CRC-8 is considered as a frame without any error.

- Fast SCMBus checksum byte is obtained by summing all the frame previous bytes then setting b7 bit to 1.

4.4. Frames structure

4.4.1. Transmission organization

- frame : ***eNod4*** address first
- byte : lsb first
- multi-bytes data : MSB first

4.4.2. Reading request

- request

Address	*Command*	*CR*	*CRC*
*1 **Hex** byte*	*1 **Hex** byte (command)*	*1 **ASCII** byte (0D$_H$)*	*1 **Hex** byte*

- SCMBus response

Address	*Status*	*Value*	*CR*	*CRC*
*1 **Hex** byte*	*2 **Hex** bytes*	*N **ASCII** Hex bytes*	*1 **ASCII** byte (0D$_H$)*	*1 **Hex** byte*

If the *'decimal point position'* and the *'unit'* settings are assigned to a non-null value, the response frame when transmitting measurement contains the decimal point character (2E$_H$) and the unit that is separated from the measurement value by a space ASCII character (20$_H$).

- Fast SCMBus response

STX	*Status word*	*Value*	*Cks*	*ETX*
02$_H$	*2 Hex bytes*	*3 signed Hex bytes (2's complement)*	*Σ of previous bytes and b7 bit set to 1*	*03$_H$*

Note: Because values are encoded in signed hexadecimal bytes format (2's complement) some data bytes can be equal to **STX (02$_H$) or ETX (03$_H$) or DLE (10$_H$)** so before those specific bytes values a **DLE (10$_H$)** byte is inserted. The ***eNod4*** address is not transmitted in the frame.

4.4.3. Functional command request (tare, zero...)

- request :

Address	*Command*	*CR*	*CRC*
*1 **Hex** byte*	*1 **Hex** byte (command)*	*1 **ASCII** byte (0D$_H$)*	*1 **Hex** byte*

- response (SCMBus and fast SCMBus) :

Address	Command	CR	CRC
*1 **Hex** byte*	*1 **Hex** byte (command)*	*1 **ASCII** byte (0D$_H$)*	*1 **Hex** byte*

If the command execution is successful, ***eNod4*** sends back the request frame that has been received as an acknowledgement.

4.4.4. Error frame

In case of an error upon reception of a request, ***eNod4*** sends back an error frame that contains an error code:

- response (SCMBus and fast SCMBus) :

Address	Error code	CR	CRC
*1 **Hex** byte*	*1 **Hex** byte (command)*	*1 **ASCII** byte (0D$_H$)*	*1 **Hex** byte*

- The error codes are listed below:

Error code	Meaning	Description
FE$_H$	*unknown command*	*requested command is not supported by **eNod4***
FF$_H$	*error during command execution*	*ex. : tare when gross meas.<0*

4.5. Address and Baud rate

Address and baud rate identical to Modbus RTU (See § Modbus RTU)

4.6. Product identification

Product identification identical to Modbus RTU (See § Modbus RTU)

4.7. Measurement transmission

Measurement transmission can be triggered by a master request but it might also be triggered and used through the following options:

- transmission triggered by a rising or falling edge on a logical input
- transmission at a configurable period (defined in ms) while a logical input is maintained at a given logical level
- continuous transmission at a configurable period (defined in ms) after a master request. The transmission is then stopped by another master instruction, be careful not to use this mode in half-duplex at a too high rate.

4.8. Continuous transmission

SCMBus and fast SCMBus communication protocols allow ***eNod4*** to transmit measurements at a user-defined rate without the need for successive master queries. To perform this measurement acquisition mode, it is necessary to set first the *'sampling period'* (in ms):

Address SCMBus	Description	Accès	Type
0x003F	*SCMBus Measurement transmission period*	*RW*	*Uint*

A value of 0 implies that measurement transmission is synchronized on the A/N conversion rate. The continuous transmission is triggered and stopped by reception of the following commands:

SCMBus/fast SCMBus functional command	*Command code*
start net measurement transmission	*$E0_H$*
start factory calibrated points transmission	*$E1_H$*
start brut measurement transmission	*$E2_H$*
stop continuous transmission	*$E3_H$*

Note 1: the measurement transmission rate also depends on the baud rate. So, to achieve the fastest transmission, it is necessary to use the highest baud rate.

Note 2: as RS485 is a half-duplex communication medium, it can be a little hard to transmit the '*stop continuous transmission'* query if the bandwidth is saturated. Therefore, prefer USB communication channel to reach the highest measurement transmission rate.

4.9. EEPROM error management

EEPROM management identical to Modbus RTU (See § Modbus RTU)

5. MODBUS TCP

When a configuration change occurs (change of Ethernet parameters, set default params via eNodView or eNodTouch) eNod4 Modbus-TCP absolutely must not be reset or power cycled within 10 seconds after send of the change. This could permanently damage the eNod. MS LED blinks green or red cyclically when in this "damaged" state.

5.1. Physical interface

eNod4 is fitted with an Ethernet interface on RJ45 connectors and is galvanically isolated.

The Auto-Crossover function is supported. Due to this fact the signals RX and TX may be switched on ETH1 and ETH2 interfaces.

Because Modbus TCP (or Modbus TCP/IP) shares the same physical and data link layers of traditional IEEE 802.3 Ethernet, physical interface remains fully compatible with the already installed Ethernet infrastructure of cables, connectors, network interface cards, hubs, and switches.

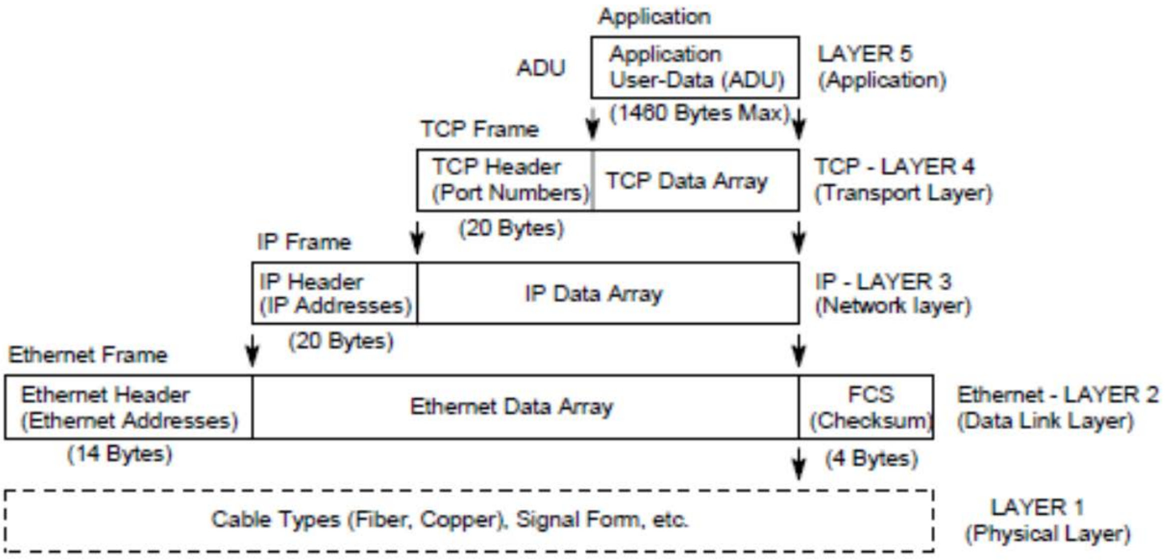
Only tree, line or star network topologies are allowed.

Every ***eNod4*** drives two Ethernet ports and has an internal switch and hub functions, respectively the different circuits which are related to the special features of some Real-Time-Ethernet systems to build up a line structure.

5.2. General information

eNod4 is fitted with an Ethernet communication interface that supports protocols TCP (Transmission Control Protocol) and IP (Internet Protocol). These protocols are used together and are the main transport protocol for the internet. When Modbus information is sent using these protocols, the data is encapsulated by TCP where additional information is attached and given to IP. IP then places the data in a packet (or datagram) and transmits it on Ethernet network.

Construction of a Modbus TCP data packet and simplified OSI model communication layers representation:


Application
ADU
Application User-Data (ADU)
(1460 Bytes Max)
LAYER 5 (Application)
TCP Frame
TCP Header (Port Numbers)
TCP Data Array
(20 Bytes)
TCP - LAYER 4 (Transport Layer)
IP Frame
IP Header (IP Addresses)
IP Data Array
(20 Bytes)
IP - LAYER 3 (Network layer)
Ethernet Frame
Ethernet Header (Ethernet Addresses)
Ethernet Data Array
FCS (Checksum)
(14 Bytes)
(4 Bytes)
Ethernet - LAYER 2 (Data Link Layer)
Cable Types (Fiber, Copper), Signal Form, etc.
LAYER 1 (Physical Layer)

TCP must establish a connection before transferring data, since it is a connection-based protocol.

The Master (or Client in Modbus TCP) establishes a connection with the Slave (or Server) ***eNod4***. The Server ***eNod4*** waits for an incoming connection for the Client. Once a connection is established, the Server ***eNod4*** then responds to the queries from the Client until the Client closes the connection.

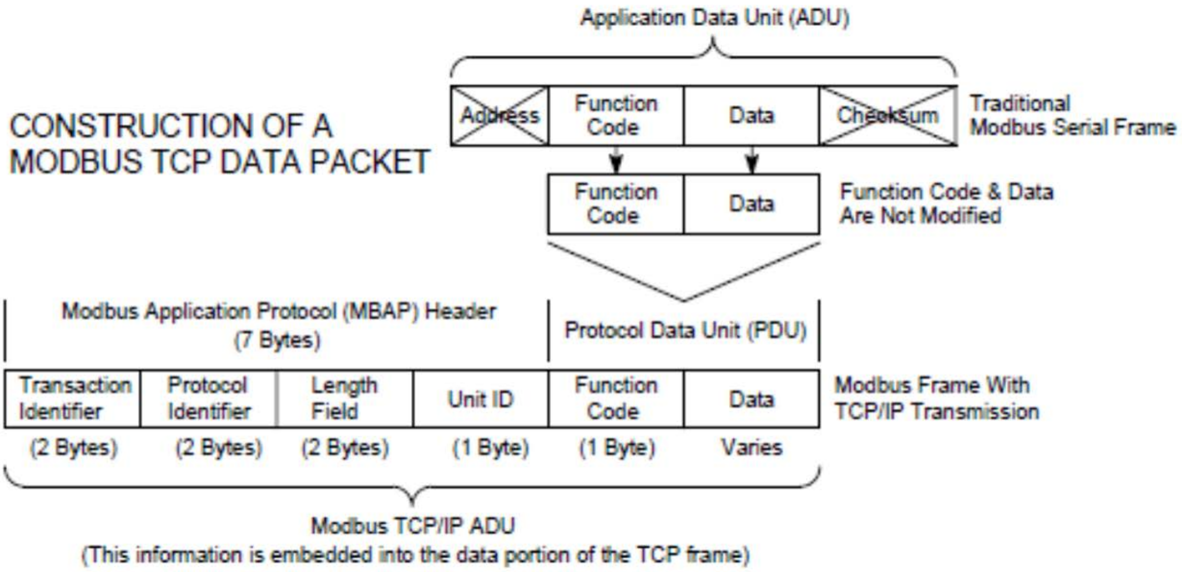
Modbus TCP/IP uses well-known specific port 502 to listen and receive Modbus messages over Ethernet.

Note: ***eNod4*** does not support **Modbus RTU over TCP** protocol (simply put, this is a Modbus RTU message transmitted with a TCP/IP wrapper and sent over a network instead of serial lines).

eNod4 supports **Modbus TCP (or Modbus TCP/IP)** protocol: a document ***Modbus Messaging on TCP/IP implementation guide*** provided by Schneider Automation outlines a modified protocol specifically for use over TCP/IP. The official Modbus specification can be found at ***Modbus organization*** (www.modbus.org).

ADU (Application Data Unit) and PDU (Protocol Data Unit): aside from the main differences between serial and network connections stated above, there are few differences in the message content between Modbus TCP and Modbus RTU.

Starting with Modbus RTU frame (**ADU**), the checksum disappears. From now on data integrity is granted by Ethernet Data Link layer. Slave ID address is suppressed and supplanted by an identifier (Unit ID) that is a part of a complementary data header called **MBAP** (Modbus Application Protocol) header. The MBAP header is 7 bytes long.


Application Data Unit (ADU)
CONSTRUCTION OF A MODBUS TCP DATA PACKET
Address
Function Code
Data
Checksum
Traditional Modbus Serial Frame
Function Code
Data
Function Code & Data Are Not Modified
Modbus Application Protocol (MBAP) Header (7 Bytes)
Protocol Data Unit (PDU)
Transaction Identifier
Protocol Identifier
Length Field
Unit ID
Function Code
Data
Modbus Frame With TCP/IP Transmission
(2 Bytes)
(2 Bytes)
(2 Bytes)
(1 Byte)
(1 Byte)
Varies
Modbus TCP/IP ADU
(This information is embedded into the data portion of the TCP frame)

MBAP header: fields are defined below:

fields	*Length (bytes)*	*Description*	*Client (Master)*	*Server (Slave)*
Transaction Identifier	*2*	*Transaction pairing (request / response Modbus)*	*Initiated by the Client*	*Echoed back by the Server*
Protocol Identifier	*2*	*0 = MODBUS Protocol*	*Initiated by the Client*	*Echoed back by the Server*
Length	*2*	*byte count of the remaining fields (Unit ID + Function Code + Data)*	*Initiated by the Client (request)*	*Initiated by the Server (response)*
Unit Identifier	*1*	*Idendification of a remote server (non TCP/IP or other buses), 0x00 or 0xFF otherwise*	*Initiated by the Client*	*Echoed back by the Server*

Supported functions: identical to Modbus RTU ones.

- Read multiple registers* : **03_H / 04_H**
- Write single register* **06_H**
- Write multiple registers* **10_H**

*1 register = 2 bytes

Maximal number of registers = 123

5.3. Frames structure

- By default and as in Modbus RTU, during a read or write transaction, the two bytes of a register are swapped. The MSB is transmitted first and then the LSB. However it may be possible using ***eNodView*** software to invert the swapping of data in a register.
- if a data is coded on 4 bytes (that means it requires two registers) , the two LSB are stored in the low address register and the two MSB are stored in the high address register Modbus RTU request command example sent to the slave in hexadecimal:

Slave address	*03_H or 04_H*	*First register address*	*N registers*	*CRC16*
1 byte	1 byte	2 bytes	2 bytes	2 bytes
11	03	00 7D	00 03	97 43

- Equivalent request in Modbus TCP:

Transaction Identifier	*Protocol Identifier*	*Message length*	*Unit Identifier*	*03_H or 04_H*	*First register address*	*N registers*
2 bytes	2 bytes	2 bytes	1 byte	1 byte	2 bytes	2 bytes
00 01	00 00	00 06	FF	03	00 7D	00 03

Modbus exception codes: like in Modbus RTU a server ***eNod4*** may generate an exception response to a client request.

- Exception codes table:

Error code	*Exception*	*Description*
01	Illegal Function	The function code received by ***eNod4*** in the query is not allowed or invalid.
02	Illegal Data Address	The data address received in the query is not an allowable address for ***eNod4*** or is invalid.
03	Illegal Data Value	A value contained in the query data field is not an allowable value or out of the limits
06	***eNod4*** Device Busy	***eNod4*** is not ready to answer (for example measurement request during a taring operation).

5.4. Network configuration

Every ***eNod4*** is identified on the network by an IP address, a subnet mask and a default gateway address. Network configuration can only be set using ***eNodView*** software at minimum version V.

IP address: the IP address is comprised of two parts: the network address or Net ID (first part), and the host address or Host ID (last part). This last part refers to a specific machine on the given sub-network identified by the first part. The numbers of bytes of the total four that belong to the network address depend on the Class definition (Class A, B, or C) and this refers to the size of the network.

Class C subnets share the first 3 octets of an IP address, giving 254 possible IP addresses for ***eNod4*** device. Recall that the first 00_H and last FF_H IP addresses are always used as a network number and broadcast address respectively.

eNod4 default local IP* address is **192.168.0.100**

******if IP static configuration set*

Subnet mask: a Subnet Mask is used to subdivide the host portion of the IP address into two or more subnets. The subnet mask will flag the bits of the IP address that belong to the network address, and the remaining bits correspond to the host portion of the address.

The unique subnet to which an ***eNod4*** IP address refers to is recovered by performing a bitwise AND operation between the IP address and the mask itself, with the result being the sub-network address.

eNod4 subnet mask default value is the default Class C subnet mask **255.255.255.0**

Gateway address: a gateway is being used to bridge Ethernet to other networks like a serial sub-network of Modbus RTU devices in order to provide communication compatibility.

The IP address of the default gateway has to be on the same subnet as the local IP address. The value 0.0.0.0 is forbidden. If no gateway is to be defined then this value is to be set to the local IP address of the ***eNod4*** device.

Default gateway address has been set to **192.168.0.254**

DHCP functionality (Dynamic Host Configuration Protocol):

It's a protocol that automates network-parameter assignment and allows an ***eNod4*** device to dynamically configure (without any particular action) an IP address and other information that is needed for network communication. ***eNod4*** device needs imperatively to be connected on the sub-network to a DHCP server that allocates IP address and also DHCP functionality has to be activated in ***eNod4*** device.

A label affixed on every ***eNod4*** contains 6 bytes of its MAC address (Media Access Control Address) which is a unique identifier assigned to network interfaces for communications on any physical network segment.

In DHCP when the Master of the sub-network attributes an IP address to a Slave (***eNod4*** device), it associates its unique MAC address to the IP address. So the MAC address is the only way for a Master to identify an ***eNod4*** device on the sub-network.

DHCP functionality is not activated by default (set to **static IP configuration**).

5.5. Modbus TCP LED

State of the **NS (Network Status)** bicolor LED is described in the table below:

Color	*State*	*Meaning*
Green	*Blinking 1Hz*	*Device **READY** but not **CONFIGURED** yet*
	Blinking 5Hz	*Device **WAITING** for communication*
	Always on	***CONNECTED** (at least one TCP connection is established)*
Red	*Blinking 2Hz (On/Off rate 25%)*	*Internal Fault detect (like TCP connection lost)*
	Always on	*Communication fatal error*
-	*Always off*	*Device not powered or defective*

State of the **MS (Module Status)** bicolor LED is described in the table below:

Color	State	Meaning
Green	*Blinking*	*Device **WAITING FOR CONFIGURATION***
	Always on	*Device is **OPERATING** correctly*
Red	*Blinking*	*Communication error detected*
	Always on	*Fatal error detected*
Red / Green	*Blinking*	*Autotest at power on*
-	*Always off*	*Device not powered or defective*

State of the **ACT / LINK** ETH1 and ETH2 network RJ45 connector LED:

Color	State	Meaning
LINK *(Eth1 & Eth2)* *Green*	*Always on*	*A physical connection to the Ethernet exist*
	Always off	*Device not connected to the Ethernet*
ACT *(Eth1 & Eth2)* *Yellow*	*On*	*The device sends/receives Ethernet frames*
	Always off	*No traffic on the Ethernet*

5.6. I/O scanning

The exchange of application data at a high refreshment rate is only possible in a specific range of Modbus addresses. Specified 28 Input registers that are exchanged in I/O scanning are defined in the table below:

Register address (Hex)	*Size in bytes (n)*	*Type*	*Name*	*Access*
007D	*2*	*Uint*	*measurement status*	*RO*
007E	*4*	*long*	*gross measurement*	*RO*
0080	*4*	*long*	*tare value*	*RO*
0082	*4*	*long*	*net measurement*	*RO*
0084	*4*	*long*	*factory calibrated points*	*RO*
0086	*20*		*reserved*	
0090	*2*	*Uint*	*command register*	*R/W***
0091	*2*	*Uint*	*response register*	*RO*
0092	*4*	*long*	*delta zero*	*R/W***
0094	*2*	*Uint*	*IN/OUT level*	*RO*
0095	*4*	*long*	*Preset tare value*	*R/W***
0097	*4*	*Ulong*	***eNod4** 1ms counter**	*RO*

**for possible check of the performances*

*** Fields that are normally R/W but RO for implicit exchanges through read multiple registers function*

6. ETHERNET IP

When a configuration change occurs (change of Ethernet parameters, set default params via eNodView or eNodTouch, change of address « Name of product » after a reset with option « Use rotary switch in product name ») eNod4 Ethernet/IP absolutely must not be reset or power cycled within 10 seconds after send of the change or reset. This could permanently damage the eNod. MS LED blinks green cyclically when in this "damaged" state.

EtherNet / IP uses Ethernet layer network infrastructure. It is built on the TCP (Transmission Control Protocol) and IP (Internet Protocol) protocols, but the "IP" in the name stands for "**Industrial Protocol**" and not an abbreviation for "Internet Protocol".EtherNet / IP is supported by four independent networking organizations

- ControlNet International (CI),
- The Industrial Ethernet Organization (IEA),
- The Open DeviceNet Vendor Association (ODVA),
- The Industrial Automation Open Network Alliance (IAONA).

6.1. Physical interface

eNod4 is fitted with two EtherNet ports on RJ45 connectors that are galvanically isolated.

The Auto-Crossover function is supported. Due to this fact the signals RX and TX may be switched on ETH1 and ETH2 interfaces. Auto-negotiation of link parameters applies to 10/100Mbit and full/half duplex operation.

Because EtherNet / IP shares the same physical and data link layers of traditional IEEE 802.3 Ethernet, physical interface remains fully compatible with already installed Ethernet infrastructure (cables, connectors, network interface cards, hubs, and switches).

EtherNet / IP automatically benefits from all further technology enhancements such as Gigabit Ethernet and Wireless technologies.

Tree, line or star network topologies are allowed by ***eNod4***. Ring topology is also supported while Device Level Ring (DLR) protocol is implemented (as ***eNod4*** is not able to act as a ring supervisor, at least one active ring supervisor is required on the DLR network).

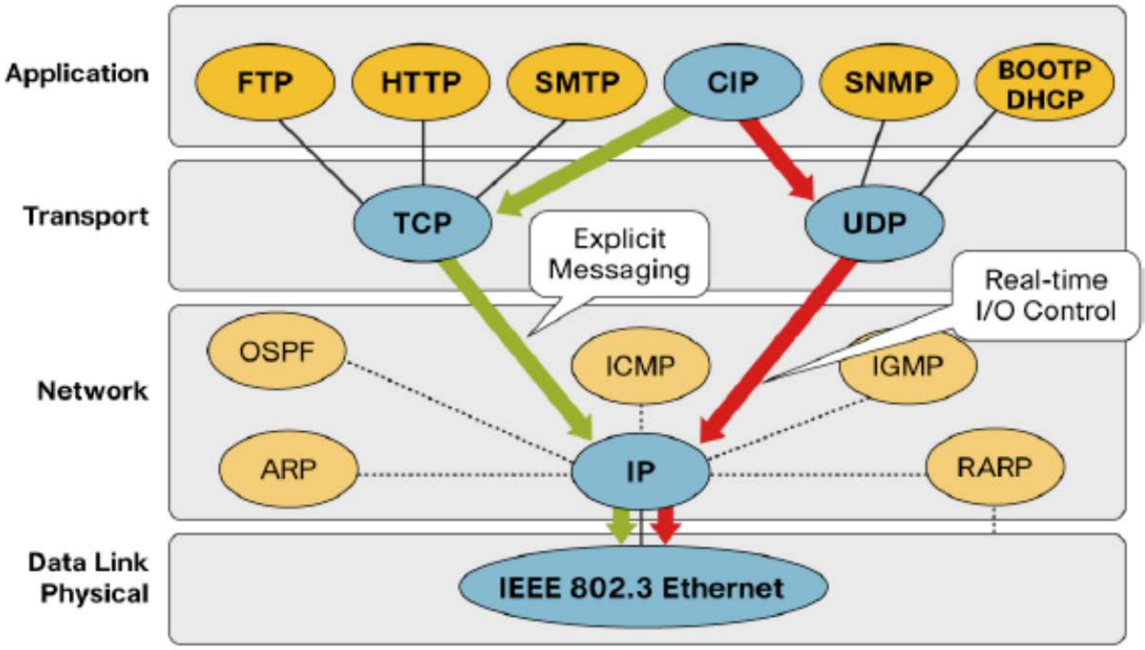
Every ***eNod4*** drives two Ethernet ports and has an internal switch and hub functions, respectively the different circuits which are related to the special features of some Real-Time-Ethernet systems to build up a line or ring structure.

6.2. General information

6.2.1. EtherNet / IP “Open standard” protocol

EtherNet / IP shares the same lower four layers of the OSI model common to all Ethernet devices. This makes it fully compatible with existing Ethernet hardware, such as cables, connectors, network interface cards, hubs, and switches. The application layer protocol is the Control and Information Protocol (CIP™).

eNod4 is fitted with an Ethernet communication interface that supports protocols **TCP** (Transmission Control Protocol), **UDP** (User Datagram Protocol) and **IP** (Internet Protocol). These protocols are used together and are the main transport protocol for the internet. When **CIP**^{TM} information is sent using these protocols, the data is encapsulated by TCP or UDP where additional information is attached and given to IP. IP then places the data in a packet (or datagram) and transmits it on **Ethernet** network.

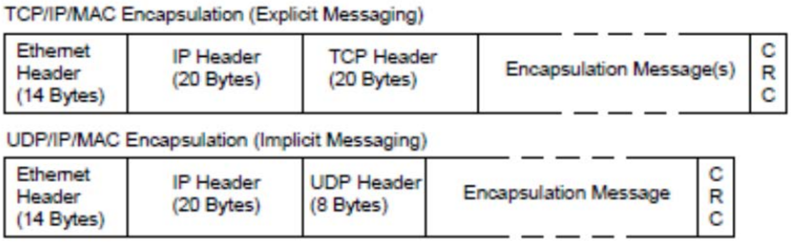


By using TCP/IP, EtherNet / IP is able to send explicit messages, which are used to perform **client-server** type transactions between nodes. Nodes must interpret each message, execute the requested task and generate responses. Uploading and downloading of configuration data like setpoints and applicative parameters uses **explicit (or Class 3) messaging**.

TCP is connection-oriented and use well known TCP port number 44818 (0xAF12) for EtherNet / IP.

For real-time messaging, EtherNet / IP also employs UDP over IP, which allows messages to be unicast (one to one) or multicast (one to a group of destination addresses) in a **producer-consumer** model. This is how CIP™ I/O data transfers called **implicit (or Class1) messaging** is sent on EtherNet / IP. With implicit messaging, the data field contains no protocol information, only real-time I/O data. Since the meaning of the data is pre-defined at the time the connection is established, processing time is minimized during runtime. UDP is connectionless and makes **no guarantee that data will get from one device to another**; however, UDP messages are smaller and can be processed more quickly than explicit messages. As a result, EtherNet / IP uses UDP/IP to transport I/O messages that typically contain time-critical control data. The CIP™ Connection mechanism provides timeout mechanisms that can detect data delivery problems, a capability that is essential for reliable control system performance.

UDP port used is port 2222 (0x08AE).



TCP/IP/MAC Encapsulation (Explicit Messaging)

Ethernet Header (14 Bytes)	IP Header (20 Bytes)	TCP Header (20 Bytes)	Encapsulation Message(s)	CRC

UDP/IP/MAC Encapsulation (Implicit Messaging)

Ethernet Header (14 Bytes)	IP Header (20 Bytes)	UDP Header (8 Bytes)	Encapsulation Message	CRC

The process of opening a connection is called Connection Origination, and the node that initiates the connection establishment request is called a Connection Originator, or just an **Originator** (so called Scanner). Conversely, the node that responds to the establishment request is called a Connection Target, or a **Target** (so called Adapter).

6.2.2. Common Industrial Protocol (CIP™)

Common Industrial Protocol (CIP™) has implementations based upon Ethernet with EtherNet / IP, but also through DeviceNet (CIP™ over CAN bus) and ControlNet (CIP™ over a dedicated network).

Most controllers (with appropriate network connections) can transfer data from one network type to the other, leveraging existing installations, yet taking advantage of Ethernet.

CIP™ is an object oriented protocol. Each CIP™ **object** has **attributes** (data), **services** (commands) and **behaviors** (reactions to events). Objects are also named **classes**. An object **instance** refers to one implementation of a class. Each instance of a class has the same attributes, but its own particular set of attribute values.

We use attributes to refer to the data of an object. You use methods to operate on the data. Every attribute of an object will have a corresponding method and you invoke a method by sending a service to it. Services are the communication mechanism between objects. CIP™ object models will use "**get**" and "**set**" messages as the methods to access their data.

The behavior of an object is what the object can do and this behavior is contained within its methods.

An integer ID value is assigned to each object **class**, each **instance** of the same class, each class **attribute** and each class **service.** There is only one assigned instance for ***eNod4*** application-specific classes.

CIP™ provides many standard services for control of network devices and access to their data via implicit and explicit messages. The key thing to remember about implicit messages is that there can be many consumers of a single network packet and this requires UDP, while TCP is instead reserved for point-to-point messages.

CIP™ also includes "device types" for which there are "device profiles". ***eNod4*** does not follow any device profile because functionality is specific. CIP™ already includes a large collection of commonly defined objects or object classes and only two objects referring to Ethernet, TCP/IP Interface Object & Ethernet Link Object.

Additional ***eNod4***-specific objects (EtherNet / IP-compliant) have been defined in order to support the functional requirements of particular applications.

eNod4 EtherNet/IP devices supports the following ODVA commonly defined objects:

- • An Identity Object (ID 0x01 class),
- • A Connection Manager Object (ID 0x06 class),
- • A TCP/IP Interface Object (ID 0xF5 class),
- • An Ethernet Link Object (ID 0xF6 class),
- • A DLR Object (ID 0x47 class),
- • A Quality of Service Object (ID 0x48 class).

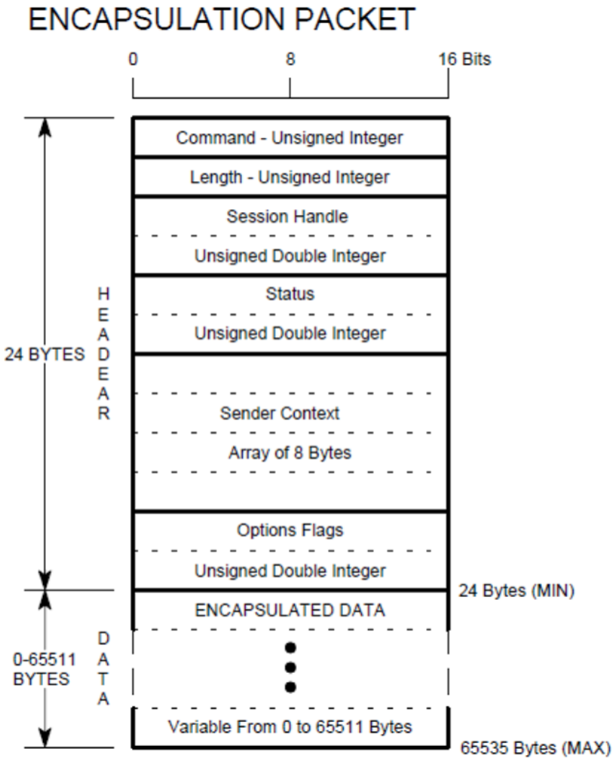
eNod4 application-specific objects are defined below:

- • A Metrology and Identification Object (ID 0x64 class),
- • A Calibration Object (ID 0x65 class),
- • A Filtering Object (ID 0x66 class),
- • A Logical Inputs/Outputs Object (ID 0x67 class),
- • A Command / Response Object (ID 0x68 class).

Corresponding Class Attributes and Services supported are described in **Appendix**.

6.2.3. CIP™ Encapsulation Format

The CIP™ Encapsulation Message (the data portion of the TCP or UDP frame) includes a 24 byte header followed by its own data (optional) and is limited to a total length of 65535 bytes. This packet takes the following format:



For any data to exchange, the encapsulated data format is **most significant bit (MSB) transmitted first.**

Access to the object model of a device is controlled by one of two objects: the Connection Manager, and the UnConnected Message Manager (**UCMM**). We have already stated that EtherNet / IP is a connection-based network and that most CIP™ messages are accomplished through connections. CIP™ also allows multiple connections to coexist in a device at any given time.

eNod4 allows up to 4 simultaneous EtherNet / IP connections (sum of explicit and implicit connections).

In addition, it is not possible on the same module to access to different device application-specific Class for multiple explicit connections. For implicit connection, ***eNod4*** accepts 1 exclusive owner and up to 2 listener only.

eNod4 supports only cyclic connection CIP™ trigger.

6.3. Network configuration

Every ***eNod4*** is identified on the network by an IP address, a subnet mask and a default gateway address. Network configuration can only be set using ***eNodView*** software at minimum version V.

IP address: the IP address is comprised of two parts: the network address or Net ID (first part), and the host address or Host ID (last part). This last part refers to a specific machine on the given sub-network identified by the first part. The numbers of bytes of the total four that belong to the network address depend on the Class definition (Class A, B, or C) and this refers to the size of the network.

Class C subnets share the first 3 octets of an IP address, giving 254 possible IP addresses for ***eNod4*** device. Recall that the first 00_H and last FF_H IP addresses are always used as a network number and broadcast address respectively.

eNod4 default local IP* address is **192.168.0.100**

**if IP static configuration set*

Subnet mask: a Subnet Mask is used to subdivide the host portion of the IP address into two or more subnets. The subnet mask will flag the bits of the IP address that belong to the network address, and the remaining bits correspond to the host portion of the address.

The unique subnet to which an ***eNod4*** IP address refers to is recovered by performing a bitwise AND operation between the IP address and the mask itself, with the result being the sub-network address.

eNod4 subnet mask default value is the default Class C subnet mask **255.255.255.0**

Gateway address: a gateway is being used to bridge Ethernet to other networks like a serial sub-network of Modbus RTU devices in order to provide communication compatibility.

The IP address of the default gateway has to be on the same subnet as the local IP address. The value 0.0.0.0 is forbidden. If no gateway is to be defined then this value is to be set to the local IP address of the ***eNod4*** device.

Default gateway address has been set to **192.168.0.254**

DHCP functionality (Dynamic Host Configuration Protocol):

It's a protocol that automates network-parameter assignment and allows an ***eNod4*** device to dynamically configure (without any particular action) an IP address and other information that is needed for network communication. ***eNod4*** device needs imperatively to be connected on the sub-network to a DHCP server that allocates IP address and also DHCP functionality has to be activated in ***eNod4*** device.

A label affixed on every ***eNod4*** contains 6 bytes of its MAC address (Media Access Control Address) which is a unique identifier assigned to network interfaces for communications on any physical network segment.

In DHCP when the Master of the sub-network attributes an IP address to a Slave(***eNod4*** device), it associates its unique MAC address to the IP address. So the MAC address is the only way for a Master to identify an ***eNod4*** device on the sub-network.

DHCP functionality is not activated by default (set to **static IP configuration**).

6.4. EtherNet / IP LED

State of the **NS (Network Status)** bicolor LED is described in the table below:

Color	*State*	*Meaning*
Green	*Blinking*	***NO CONNECTIONS****: device has no connections established, but has obtained an IP address*
	Always on	***CONNECTED*** *(at least one connection is established)*
Red	*Blinking*	***CONNECTION TIMEOUT****: one or more of the connections in which this device is a target has timed out.* *This shall be left only if all timed out connections are reestablished or if the device is reset.*
	Always on	***DUPLICATE IP****: the device has detected that its IP address is already in use*
Red / Green	*Blinking*	*Autotest at power on*
-	*Always off*	*Device not powered or defective*

State of the **MS (Module Status)** bicolor LED is described in the table below:

Color	State	Meaning
Green	*Blinking*	***STANDBY**: the device has not been configured*
	Always on	***DEVICE OPERATIONAL**: Device is operating correctly*
Red	*Blinking*	***MINOR FAULT**:* *the device a detected a recoverable minor fault*
	Always on	***MAJOR FAULT**:* *the device a detected a non-recoverable major fault*
Red / Green	*Blinking*	*Autotest at power on*
-	*Always off*	*Device not powered or defective*

State of the **ACT / LINK** ETH1 and ETH2 network RJ45 connector LED:

Color	State	Meaning
LINK *(Eth1 & Eth2)* *Green*	*Always on*	*A physical connection to the Ethernet exist*
	Always off	*Device not connected to the Ethernet*
ACT *(Eth1 & Eth2)* *Yellow*	*On*	*The device sends/receives Ethernet frames*
	Always off	*No traffic on the Ethernet*

6.5. I/O scanning / implicit messaging

eNod4 Target (Adapter) consumes necessarily one single register (2 bytes without header) of Output data (from the network's point of view) through Assembly Instance 0x64 (100) with a Cyclic transport trigger type and point to point connection type. Data exchanged is the command register which is the attribute 1 of device application-specific 0x68 class.

eNod4 produces Input data (from the network's point of view) through Assembly Instance 0x65 (101) with a Cyclic transport trigger type. Multicast or point to point connection type, connection rate, size and priority are defined when the connection is established by the Originator (Scanner) through the connection manager Object using the *Forward_open* Service (Connection is closed using the *Forward_close* Service).

Find in the table below the specified registers (28 bytes without header) that are produced through Assembly Instance 0x65 (101):

Register Modbus Address (Hex)	*Offset in bytes (without header)*	*Type*	*Name*
/	*0*	*long*	***eNod4** 1ms counter**
0094	*4*	*Uint*	*Input / Output levels*
007D	*6*	*Uint*	*Measurement status*
007E	*8*	*long*	*Gross measurement*
0080	*12*	*long*	*Tare value*
0082	*16*	*long*	*Net measurement*
0084	*20*	*long*	*Factory calibrated points*
0090	*24*	*Uint*	*Command register*
0091	*26*	*Uint*	*Response register*

**for possible check of the performances*

Find in the table below the specified register (2 bytes without header) that is consumed through Assembly Instance 0x64 (100):

Register Modbus Address (Hex)	*Offset in bytes (without header)*	*Type*	*Name*
0090	*0*	*Uint*	*Command register*

The register "Command register" uses the mechanism of ***eNod4*** functional commands defined in another chapter.

Note: "reset" and "Restore default settings" commands cannot be sent via cyclic and acyclic exchanges immediately after a restart of ***eNod4***. To be able to use these commands, it must first be processed another command ("cancel Tare" for example).

Note: The "Command register" data **must be** set to 0x0000 before each new command.

7. PROFINET IO

When a configuration change occurs (change of Ethernet parameters, set default params via eNodView or eNodTouch, change of address «Name of the station » after a reset with option « Use rotary switch in name of the station») eNod4 Profinet absolutely must not be reset or power cycled within 10 seconds after send of the change or reset. This could permanently damage the eNod. MS LED blinks green cyclically when in this "damaged" state.

PROFINET is the communication standard created by the PROFIBUS International organization. It allows use of an industrial Ethernet network for real time data exchange between automation components. Whereas PROFINET CBA variant allows splitting intelligence of the application over network components, the PROFINET IO variant allows the exchange of I/O data between an IO-controller (e.g. PLC (Programmable Logic Controller)) that contains the intelligence of the application and IO-devices. ***eNod4 ETH Profinet*** is an IO-device and can exchange data only with one IO-controller.

7.1. Physical interface

eNod4 is fitted with two Ethernet ports on RJ45 connectors that are galvanically isolated. They support the switch or hub functions, specific functions of real time Ethernet systems and facilitate the implementation of line or ring topology.

The function of automatic crossing of emission line and reception line (Auto-Crossover Rx/Tx) on ETH1 and ETH2 interfaces is supported. Auto-negotiation of Ethernet link layer settings applies to the choice of the 10/100Mbit speed as well as Full or Half-Duplex operations.

As PROFINET IO communicates on Ethernet II type frames, ***eNod4*** is compatible with most of the existing network infrastructures (cards, connectors, network, hub and switches).

Each ***eNod4*** has a hardware **MAC** address (Media Access Control address). A label affixed to each ***eNod4*** includes the 6-bytes MAC address. It is a unique identifier of any Ethernet network hardware.

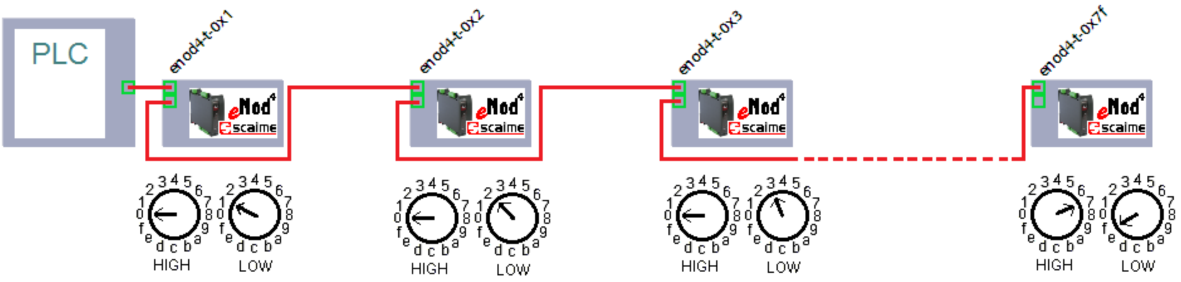
7.2. Network settings

All PROFINET IO network settings and options are configurable using the eNodView software to V version minimum.

IP settings: IP address, subnet mask and default gateway. Default values of these parameters are (192.168.0.100, 255.255.255.0, 192.168.0.254). Configuration of these settings via eNodView is of little interest. Usually it is the IO-Controller which assigns to each IO-Device its IP settings using the name of the station.

Name of the station: The name of the station is the primary key that allows the identification of the PROFINET IO node. So, it must be unique for each node on PROFINET IO subnet. It can only contain lowercase characters, figures, dashes and dots. The default value of this parameter is based upon (configurable option) the rotary switches located in front of ***eNod4***. It is set to:

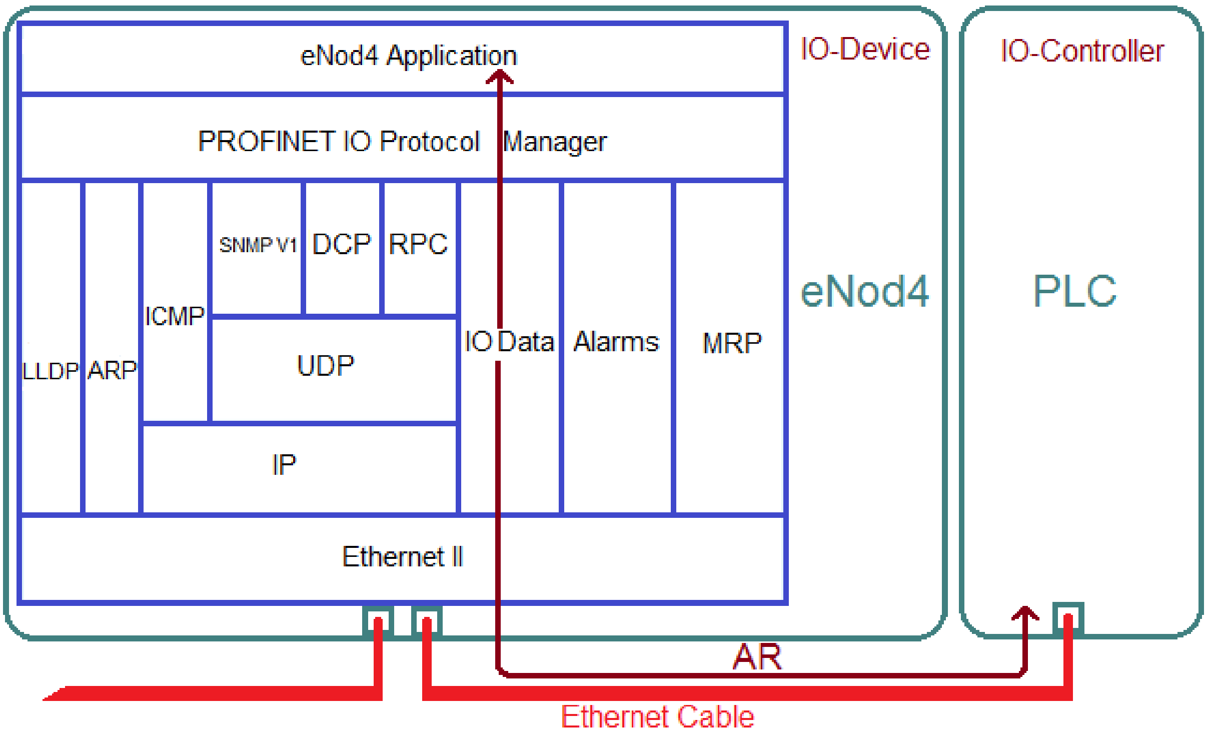
“enod4-t-0x'address_on_rotary_switches_in_lowercase_hexadecimal'” for ***eNod4***-T.



*PROFINET IO network and names of **eNod4-T** stations in factory configuration. Only rotary switches have been reconfigured.*

<u>Byte order:</u> The byte order defines the order in which the application data are emitted on the network. The two possibilities are "Big Endian" or "Little-Endian". With AA as least significant byte, data of 2 or 4 bytes length are coded for each possibility in this way: "Big Endian" 2 bytes: AA BB, 4 bytes: AA BB CC DD; "Little Endian" 2 bytes: BB AA, 4 bytes: DD CC BB AA. The default value of this parameter is "Little Endian".

7.3. Definition of protocols roles



*PROFINET IO protocols stack inside **eNod4**.*

Protocols involved in setting up an IO-Device (***eNod4***) and the establishment and maintenance of a cyclic data connection are described below:

- LLDP (Link Layer Discovery Protocol). The LLDP messages are sent regularly on the network and inform other nodes about the identity of ***eNod4***.
- IP (Internet Protocol) allows routing of packets on the sub network by using IP address.
- ARP (Address Resolution Protocol). This protocol allows the creation of a resolution table of MAC addresses from an IP address. This table will be used in each node when a layer protocol based on IP (which uses an IP address) may wish to send a packet to another node on the Ethernet (MAC address) network.
- ICMP (Internet Control Message Protocol). Allows the 'Ping' command on the ***eNod4***.

- UDP (User Datagram Protocol) allows specification of a port number for an IP packet. The port number is associated with a higher level protocol.

- SNMP V1 (Simple Network Management Protocol) allows the network administrator to manage and oversee the whole network, including ***eNod4***.

- DCP (Discovery and Configuration Protocol). Enables the discovery and configuration of PROFINET nodes. The main functionality is similar to the more commonly used protocol DHCP (unsupported). Main available services are:

    - Identify: Allows an application to identify all PROFINET nodes present on the network, including ***eNod4***.

    - Signal: Allows the user to flash an LED on a specified node to identify the corresponding hardware equipment.

    - Set IP (remanent or not). Allows the assignment of IP parameters (IP address, subnet mask, default gateway) for a node. Remanent means that parameters will keep their values after a power cycle, in non-remanent that they will be recovered to their previous values.

    - Set Name Of Station (remanent or not). Allows the allocation of the name of the station for a node. Used in remanent, this service disables the option “use rotary switches for name of the station”; to reactivate it you can use eNodView.

    - Set Reset Factory Settings: Allows the reset of all settings (application and networks) from ***eNod4*** to their default values. It places the IP settings to (0.0.0.0, 0.0.0.0, 0.0.0.0), turns the current name of the station into an empty field and disables the option to use rotary switches for name of the station.

- RPC (Remote Procedure Call): Allows the management of connections (called **AR (Application relation)** and CR (Communication relation)) for the exchange of cyclic data (IO Data) between the IO-Controller (PLC) and the Device-IO (***eNod4***). Allows also acyclic exchanges (called read/write Records).

- Profinet IO Data: Cyclic PROFINET IO data, these carrying data also contain status informations on the transported data. Compared with other communication standards based on Ethernet, useful cyclic data goes through fewer layers before reaching their destination. For example the IP network layer is not crossed by cyclic data (IO Data).

- Alarms: PROFINET IO alarms are sent by a node whenever a significant event occurs. ***eNod4*** sends an alarm on every appearance and disappearance of diagnostic that reports an application error. Error types corresponding to ***eNod4*** diagnostics are described in the appendix and in the GSDML file. This file can be imported into the engineering software used for the network monitoring.

- MRP (Media Redundancy Protocol): This Protocol allows ring topology. ***eNod4*** acts as a **MRP client** and is not able to act as manager. At least one manager (MRP Manager) is required on the network if the ring topology is desired.

7.4. Main scenario

The main scenario applies to PROFINET IO network; it can be used to diagnose possibly encountered problems on the network.

1. *PROFINET IO network is powered on.*
2. *IO-Devices emit LLDP frames to inform all nodes on the subnet of their presence and identity.*

3. *Network nodes resolve the IP addresses of the stations with which they wish to communicate in peer-to-peer using the ARP protocol.*
4. *With DCP services, IO-Controller identifies IO-Devices involved in its application. It configures their IP settings. ARP tables are updated consequently.*
5. *Using RPC, the IO-Controller opens and configures cyclic connections (AR) for data exchange with IO-Devices and if needed reads and writes application parameters.*
6. *Cyclic data exchanges begin between IO-Devices and the IO-Controller in both directions.*
7. *The application of IO-Controller operates with the data provided by IO-Devices and supplies data to IO-Devices to advance the process of the application.*

7.5. Alternative scenario: control, maintenance, supervision

On point 4 of the main scenario:

4 A. If the network manager wants to control, maintain or supervise the network

*4. A.1. The network manager Ping the **eNod4.***

*4. A.2. The network manager consults the network information base of the **eNod4** with SNMP V1.*

7.6. Alternative scenario: eNod4 error application detected

On point 7 of the main scenario:

*7 A. **eNod4** detects an application error*

*7. A.1. **eNod4** sends an alarm of appearance of diagnostic to the IO-Controller which opened and configured a data exchange connection with it.*

7. A.2. The network manager consults diagnostics, determines the cause of the problem and fixes it.

*7. A.3. **eNod4** sends an alarm of disappearance of diagnostic to the IO-Controller which opened and configured a data exchange connection with it.*

7.7. PROFINET IO LEDs

State of the **BF (Bus Fault) labeled NS** (Network Status) bicolor LED is described in the table below:

Color	*State*	*Meaning*
Green	*Blinking*	*A data connection is established and the DCP Signal service was initiated via the bus.*
Red	*Blinking*	*No exchange of data.*
	Always on	*Ethernet physical connection low speed detected or no physical connection detected.*
Red/Green	*Blinking*	*Self-test on power up*
-	*Always off*	*No error*

State of the **SF (System Fault) labeled MS** (Module Status) bicolor LED is described in the table below:

Color	*State*	*Meaning*
Green	*Blinking*	***STANDBY****: the device has not been configured*
	Always on	***DEVICE OPERATIONAL****: Device is operating correctly*
Red	*Blinking*	***MINOR FAULT****:* *the device detected a recoverable minor fault*
	Always on	***MAJOR FAULT****:* *the device detected a non-recoverable major fault*
Red/Green	*Blinking*	*Self-test on power up*
-	*Always off*	*Device not powered or defective*

State of the **ACT / LINK** ETH1 and ETH2 network RJ45 connector LED:

Color	*State*	*Meaning*
LINK ***(Eth1 & Eth2)*** ***Green***	*Always on*	*A physical connection to the Ethernet exist*
	Always off	*Device not connected to the Ethernet*
ACT ***(Eth1 & Eth2)*** ***Yellow***	*On*	*The device sends/receives Ethernet frames*
	Always off	*No traffic on the Ethernet*

7.8. Data arrangement

The provision model of data is very similar to the one used in PROFIBUS DP, this will allow users of ***eNod4*** **Profibus** an easy recycling of their application.

7.8.1. Cyclic data (IO Data)

Cyclic exchanged data are either provided by the IO-Device and consumed by the IO-Controller or provided by the IO-Controller and consumed by the IO-Device.

Data are contained in input or input/output modules (from the point of view of the IO-Controller). These modules are defined in the GSDML file and are presented in a separate chapter.

The designer can select modules that he needs and place them in communication slots. Thus, the slots contain modules. Slots are numbered. Slot 0 is not usable for data exchange, it contains DAP (Device Access Point) informations which defines, among other, which data module can be contained in which slots.

7.8.2. Acyclic data (Records)

Acyclic data are available in read-only or read/write access. They are accessed by using a slot, a sub slot and an **index**. ***eNod4*** acyclic data are accessible with any slot and sub slot. Indexes for the ***eNod4*** specific application data are presented in appendix.

7.9. PROFINET IO exchange of cyclic data

Acyclic data modules are described in GSDML file. This file can be imported into the engineering tool used for application design. Data modules can be freely plugged into any slot from 1 to 8. This will define the organization of cyclic data in the AR (Application Relation). Unnecessary modules for the application may not be plugged. Inserting data provided by ***eNod4*** automatically implies the insertion of data consumed by ***eNod4*** if the concerned module contains consumed data.

Presentation of provided data in modules:

Module name	*Provided size in bytes*	*Provided Data*
Status+Gross Measurement	*6*	*Measurement status (2 bytes)*
		Gross measurement (4 bytes)
Net Measurement	*4*	*Net measurement*
Factory calibrated Meas.	*4*	*Factory calibrated points*
Logical I/O level	*2*	*Logical I/O level*
Measurement Status	*2*	*Measurement status*
Command/Response Reg	*2*	*Response register*
R/W request Reg.	*6*	*Transaction status (2 bytes)*
		Data read/written (4 bytes)
1 ms counter	*4*	***eNod4** 1ms counter **

**for possible check of performances*

Presentation of consumed data in input/output modules:

Module name	*Consumed size in bytes*	*Consumed Data*
Command/Response Reg	*2*	*Command register*
R/W request Reg.	*6*	*Transaction request (2 bytes)*
		Data to be written (4 bytes)

The module "Command/Response Reg" uses the mechanism of ***eNod4*** functional commands defined in another chapter. The only difference is for "reset" and "Restore default settings" commands which cannot be sent via cyclic exchanges immediately after a restart of ***eNod4***. To be able to use these commands, it must first be processed another command ("cancel Tare" for example).

Note: The "Command register" data **must be** set to 0x0000 before each new command.

The module "R/W request Reg." allows requesting read/write of Record (acyclic data). So this **substitute** read/write of Record via the RPC protocol. The protocol described below (which is the same than the one used on ***eNod4*** Profibus product) allows performing read/write operations:

IN	*OUT*
Transaction status (2 bytes)	*Transaction request (2 bytes)*
Data read/written (4 bytes)	*Data to be written (4 bytes)*

An IO-Controller can transmit a read or write request to ***eNod4*** by writing a specific code (see the codes listed in the appendix) into the transaction request register.

⇨ For a write request, the 4 following OUT bytes can be used so as to enter the new value.
⇨ ***eNod4*** IN are then updated :

- Transaction status is set to 0xFFFF in case of an error otherwise it takes the same value as the one entered in the transaction request word.
- For a read transaction, the value of the requested setting is set into the four IN following bytes.

- For a write transaction the value of the data to be written is copied into the four IN following bytes.

Note: For 2-bytes size data, the data is read/written through the 2 least significant bytes. Ignore the 2 most significant bytes.

Note: The "Transaction request" register **must be** set to 0x0000 before every new transaction.

8. MEASUREMENT AND STATUS

Name	*Modbus address*	*Ethernet/IP Class/Attribute (hex/dec)*	*Profinet Record Index*	*Profinet cyclic Req Code*	*Type*	*Access*
Measurement status	*0x007D*	*/*	*/*	*/*	*Uint*	*RO*
Gross measurement	*0x007E*	*/*	*/*	*/*	*Long*	*RO*
Tare value	*0x0080*	*/*	*0x0060*	*R:0x0470* *W: /*	*Long*	*RO*
Net measurement	*0x0082*	*/*	*/*	*/*	*Long*	*RO*
Factory calibrated points	*0x0084*	*/*	*/*	*/*	*Long*	*RO*
Preset Tare	*0x0095*	*0x65/16*	*0x0061*	*R:0x0496* *W:0x0497*	*Ulong*	*RW*

Note: eNod4 Ethernet, see Ethernet I/O scanning chapter.

8.1. Measurement transmission

The ***eNod4*** transmits measurement after signal and data processing through different protocols available. The accessible variables are:

8.1.1. Measurement status

The measurement status contains information on eNod4 measurement parameters.

8.1.2. Gross measurement

The *'gross measurement'* stands for the digital value after measurement scaling. It is affected by all the *'zero'* functions (power-up zero, zero tracking and zero requests).

8.1.3. Net measurement

The *'net measurement'* stands for the digital value after measurement scaling and tare subtraction.

8.1.4. Tare value

The *'tare value'* stores the calibrated value that is subtracted from the *'gross measurement'* so as to give the *'net measurement'*.

8.1.5. Factory calibrated points

The *'factory calibrated points"* contains the measurement value without the user calibration layer. It is directly linked to the analog input voltage.

8.1.6. Logical IN/OUT level

The *'logical IN/OUT level'* allows reading any time ***eNod4*** logical inputs and outputs level.

8.1.7. Preset Tare value

A previous calculated tare can be restored using this variable

8.2. Measurement status

The *'measurement status'* bytes contain information about every measurement processed by ***eNod4***. See the flags meaning in the table below:

Bits	*Meaning*	*Note*
$b_1 b_0$		
00	*gross measurement*	*only in SCMBus/fast communication protocols* *not significant otherwise (00)*
01	*net measurement*	
10	*factory calibrated measurement*	
11	*tare value*	
$b_3 b_2$		
00	*measurement OK*	*causes an output assigned to the 'defective measurement' function to be set active*
10	*gross meas.< (- max capacity)*	
10	*gross meas. > (max capacity*	
11	*analog signal out of the A/D converter input range*	
b_4		
0	*motion*	*causes an output assigned to the 'motion' function to be set active*
1	*no motion*	
b_5		
0	*measurement out of the ¼ of division*	
1	*zero in the ¼ of division*	
b_6		
0	*EEPROM OK*	*See Note 1*
1	*EEPROM failure*	
b_7		
0	*reserved*	*1 in SCMBus and fast SCMBus, 0 otherwise*
1		
b_8		
0	*IN1 logical level*	
1		
b_9		
0	*IN2 logical level*	
1		

Bits	*Meaning*	*Note*
b_{10}		
0	*OUT1 logical level*	
1		
b_{11}		
0	*OUT2 logical level*	
1		
b_{12}		
0	*OUT3 logical level*	
1		
b_{13}		
0	*OUT4 logical level*	
1		
b_{14}		
0	*no tare*	
1	*at least a tare has been processed*	
b_{15}		
0	*reserved*	*1 in SCMBus and fast SCMBus, 0 otherwise*
1		

Note 1: Functioning and calibration parameters are stored in EEPROM. After every reset the entireness of parameters stored in EEPROM is checked. If a default appears, measurements are set to 0xFFFF and default is pointed out in measurement status.

9. PROCESSING FUNCTIONAL COMMANDS

Name	*Modbus address*	*Ethernet/IP Class/ Attribute (hex/dec)*	*Profinet Record Index*	*Profinet cyclic Req Code*	*Type*	*Access*
Command register	*0x0090*	*0x68/1*	*/*	*/*	*Uint*	*RW*
Response register	*0x0091*	*0x68/2*	*/*	*/*	*Uint*	*RO*

9.1. Principles

The ***eNod4*** is able to handle several functional commands thanks to a couple of registers (except in SCMBus protocols):

***the command register** : dedicated to accept the functional commands*

***the response register :** gives the state of the command currently being processed by **eNod4** (no command, in progress, finished, failed)*

- **00_H** ⇒ free to accept a new command
- **01_H** ⇒ command execution in progress
- **02_H** ⇒ command execution complete
- **03_H** ⇒ error during command execution

Note 1: **IMPORTANT** except in SCMBus/fast SCMBus protocols, to accept a new command, the command register **must be set to 00_H** first. This causes the response register to be set back to **00_H.**

9.2. Functional commands list

Functional command	Command code	Note
reset*	D0$_H$	
EEPROM Back up	D1$_H$	
restore default settings	D2$_H$	
zero*	D3$_H$	
tare*	D4$_H$	
cancel tare*	D5$_H$	
cancel last command	D6$_H$	
theoretical scaling	D7$_H$	
zero adjustment	D8$_H$	
start physical calibration	D9$_H$	physical calibration procedure
calibration zero acquisition	DA$_H$	
segment 1 acquisition	DB$_H$	
segment 2 acquisition	DC$_H$	
segment 3 acquisition	DD$_H$	
Back up calibration	DE$_H$	End of calibration (physical/theoretical) procedure
OUT1 activation/deactivation*	E6$_H$	only possible if the outputs are assigned to the associated function
OUT2 activation/deactivation*	E7$_H$	
OUT3 activation/deactivation*	E8$_H$	
OUT4 activation/deactivation*	E9$_H$	
zero offset	F0$_H$	
Preset tare*	F2$_H$	

Note: Only the commands with a * can be handled by ***eNod4*** in SCMBus and fast SCMBus protocols.

9.3. Functional commands description

9.3.1. Reset

The *'reset'* functional command execution is similar to the device power-up. This reboot phase is necessary if the address or/and the baud rate are modified and some settings changes are only taken into account after an EEPROM storage (see § .3.2) followed by a reset.

9.3.2. EEPROM storage

eNod4 configuration and calibration are stored in a non-volatile memory (EEPROM). If changes are made in the device configuration, sending to ***eNod4*** the *'EEPROM storage'* functional command will allow ***eNod4*** to keep these modifications after a power shutdown or the reception a *'reset'* functional command.

Moreover the settings listed below need to be stored and will only be taken into account at the next device reboot:

- span adjusting coefficient
- calibration place **g** value
- place of use **g** value
- stability criterion
- legal for trade activation switch
- power-up zero
- A/D conversion rate
- communication protocol

9.3.3. Restore default settings

The *'restore default settings'* command causes ***eNod4*** to be set back to its default configuration. The default configuration corresponds to the one on delivery that means with factory settings. Be careful when using this command, all the default settings are recovered including the stored calibration and the legal for trade indicators.

Note: this functional command is not available in CANopen® communication protocol.

9.3.4. Zero

When receiving a *'zero'* functional command***,*** ***eNod4*** acquires a volatile zero (gross measurement is set to 0) value if the following conditions are respected:

- measurement is stable
- Current gross measurement is within a ±10% (±2% if the legal for trade option is enabled) range of the *'maximum capacity'*.

Otherwise, after five seconds the command is cancelled and an execution error is reported.

9.3.5. Tare

When receiving a *'tare'* functional command***,*** ***eNod4*** acquires a volatile tare (net measurement is set to 0) value if the measurement is stable otherwise, after five seconds the command is cancelled and an execution error is reported. If the tare acquisition is successful b_{14} bit of the *'measurement status'* (see §6) is set to 1.

9.3.6. Cancel tare

This command erases the current tare value if at least one tare has been previously processed. It also causes b_{14} bit of the *'measurement status'* to be set back to 0.

9.3.7. Cancel last command

This command sets the response register to $\mathbf{00}_H$ and allows ***eNod4*** to ignore the functional command previously received (for example to exit a sequential procedure like a physical calibration).

9.3.8. Theoretical scaling

The *'theoretical scaling'* functional command involves the *'maximum capacity'* and *the 'sensor sensitivity'* settings. When used, this command realizes an automatic scaling to migrate from the factory calibration to the user calibration (see §8). This calibration must then be saved by sending to ***eNod4*** the *'store calibration'* functional command. Using the *'zero adjustment'* functional command is also recommended so as to completely adapt ***eNod4*** to the application.

9.3.9. Zero adjustment

The *'zero adjustment'* functional command allows the user to set his calibration zero value by asking ***eNod4*** to acquire the current factory calibrated measurement. This acquisition duration depends on the measurement stability; if stability is not reach after 5 seconds, '*zero adjustment*' command is cancelled and an execution error is reported. If it is correctly achieved, this calibration zero modification must then be saved by sending to ***eNod4*** the '*store calibration'* functional command. This functional command can be used any time and has no effect on the user-span that can have been previously configured through a physical or a theoretical calibration procedure.

9.3.10. Start physical calibration

In order to handle a physical calibration with 1 up to 3 know references, ***eNod4*** first must be told to enter the physical calibration mode. It is the first step of a sequential procedure.

9.3.11. Calibration zero acquisition

The *'calibration zero acquisition'* is the second step of the physical calibration procedure. It can only be used if the *'start physical calibration'* functional command has been previously received. This acquisition duration depends on the measurement stability; if stability is not reach after 5 seconds, '*calibration zero acquisition*' command is cancelled and an execution error is reported.

Note: In specific cases (silo for example), this step is not mandatory because it is possible to command a "zero adjustment" when the silo is empty.

9.3.12. Segment 1/2/3 acquisition

Next step consists in applying a known reference on the sensor then sending the *'segment X acquisition'* functional command where X depends on the value stored in the *'number of calibration segments'* register (see § calibration). This acquisition duration depends on the measurement stability; if stability is not reach after 5 seconds, ' *actual segment acquisition*' command is cancelled and an execution error is reported.

9.3.13. Back up calibration

Only if the *'segment 1/2/3 acquisition'* is successful, next step consists in validating the new calibration by storing the zero and the span that have been determined in EEPROM.

Note: This functional command has to be transmitted at the end of a physical calibration, after a *'zero adjustment'*, a *'theoretical scaling'* or a *'zero offset'*.

9.3.14. Logical outputs 1-4 activation/deactivation

If the corresponding logical outputs are assigned to the *'level on request'* function, they can be enabled/disabled by transmitting one of these functional commands. Upon first reception, the corresponding output is enabled and on next reception it will be disabled. If the requesting logical output is assigned to the wrong function, ***eNod4*** reports an error.

9.3.15. Zero offset

It is also possible to adjust the calibration zero value without acquiring a new one. By entering a positive or negative value into the *'delta zero'* register, the user can quantify the offset (in factory calibrated points) that has to be added or subtracted from the actual calibration zero. This calibration zero modification must then be saved by sending to ***eNod4*** the *'store calibration'* functional command.

9.3.16. Preset tare:

With this command it is possible to retrieve a tare value defined previously.

Important: Preset tare value must be stored in corresponding parameter before to send this command.

10. CALIBRATION SETTINGS AND PROCEDURES

Name	Modbus address	Ethernet/IP Class/ Attribute (hex/dec)	Profinet Record Index	Profinet cyclic Req Code	Type	Access
Maximum capacity	*0x000C*	*0x65/1*	*0x0020*	*R:0x0420* *W:0x0421*	*Ulong*	*RW*
Number of calibration segments	*0x000E*	*0x65/2*	*0x0021*	*R:0x0222* *W:0x0223*	*Uint*	*RW*
Calibration load 1	*0x000F*	*0x65/3*	*0x0022*	*R:0x0424* *W:0x0425*	*Ulong*	*RW*
Calibration load 2	*0x0011*	*0x65/4*	*0x0023*	*R:0x0426* *W:0x0427*	*Ulong*	*RW*
Calibration load 3	*0x0013*	*0x65/5*	*0x0024*	*R:0x0428* *W:0x0429*	*Ulong*	*RW*
Sensor sensitivity	*0x0015*	*0x65/6*	*0x0025*	*R:0x042A* *W:0x042B*	*Ulong*	*RW*
Scale interval	*0x0017*	*0x65/7*	*0x0026*	*R:0x022C* *W:0x022D*	*Uint*	*RW*
Zero calibration	*0x0018*	*0x65/8*	*0x0027*	*R:0x0434* *W:0x0435*	*Long*	*RW*
Span coefficient 1	*0x001A*	*0x65/9*	*0x002B*	*R:0x0436* *W:0x0437*	*Float*	*RW*
Span coefficient 2	*0x001C*	*0x65/10*	*0x002C*	*R:0x0438* *W:0x0439*	*Float*	*RW*
Span coefficient 3	*0x001E*	*0x65/11*	*0x002D*	*R:0x043A* *W:0x043B*	*Float*	*RW*
Span adjusting coefficient	*0x0020*	*0x65/12*	*0x0028*	*R:0x042E* *W:0x042F*	*Ulong*	*RW*
Calibration place g value	*0x0022*	*0x65/13*	*0x0029*	*R:0x0430* *W:0x0431*	*Ulong*	*RW*
Place of use g value	*0x0024*	*0x65/14*	*0x002A*	*R:0x0432* *W:0x0433*	*Ulong*	*RW*
Zero offset	*0x0092*	*0x65/15*	*0x002E*	*R:0x0472* *W:0x0473*	*Long*	*RW*
Preset tare value	*0x0095*	*0x65/16*	*0x0061*	*R:0x0496* *W:0x0497*	*Ulong*	*RW*

10.1. Principles

Both ***eNod4*** analog channels are configured to deliver points depending on the analog signal range:

- **500 000 pts for 2 mV/V** on the Wheatstone bridge input
- **100 000 pts for 10 V$_{DC}$** on the 0-10V analog channel

The measurement scaling in ***eNod4*** can be adapted to his application by the user. Some settings and the 2 calibration methods allow the user to define his specific span according to his sensors characteristics.

10.2. Calibration methods

Measurement scaling can be defined using one of the two following methods:

- **Theoretical calibration** involving the sensitivity of the sensor and a user-defined corresponding capacity

- **Physical calibration** involving 1, 2 or 3 know loads (for a load cell) or 1,2 or 3 measured voltages (for the 0-10 V analog channel)

Both can be achieved thanks to the functional commands.

10.3. Settings description

10.3.1. Maximum capacity

The *'maximum capacity'* stands for the maximum sensor/load cell signal range. When the absolute value of the gross measurement exceeds its value plus 9 divisions, the b_3 bit (positive overloading) or the b_2 bit (negative overloading) of the measurement status is set to 1 (it can activate a logical output if it is assigned to the *'defective measurement'* function).

The zero acquisition (on request or at power-up) is done only if the gross measurement value is contained between a ±10% range of the *'maximum capacity'* (±2% if the *legal for trade* option is active).

The *'maximum capacity'* setting also allows calibrating ***eNod4*** in case of a theoretical calibration in association with the sensor sensitivity. Measurement scaling will be automatically adapted so as to deliver a gross measurement value equivalent to the *'maximum capacity'* for an analog signal corresponding to the sensor sensitivity.

After a theoretical calibration, the maximum capacity can be changed to fit to the application.

Admitted values : from 1 up to 10000000.

10.3.2. Number of calibration segments

The *'number of calibration segments'* defines how many calibration segments are used during the physical calibration procedure. Linear installations only need one segment.

Admitted values : from 1 up to 3.

10.3.3. Calibration loads 1/2/3

Before starting a physical calibration procedure, each calibration segment must be given a corresponding user value (for example, 1000 points for a 1 kg load).

Admitted values : from 1 up to 1000000.

10.3.4. Sensor sensitivity

The *'sensor sensitivity'* setting is used to achieve a theoretical calibration. The stored value for this parameter can be:

- the **load cell sensitivity in mV/V** for the low-level analog channel
- an **input signal voltage in V** for the analog 0-10V analog channel

The user can adapt the value delivered by ***eNod4*** for the associated signal using the *'maximum capacity'* and the *'sensor sensitivity'*.

This setting is expressed with a 10^{-5} factor (197500 is equivalent to a 1.975 mV/V load cell sensitivity or a 1.975 V input voltage).

Admitted values : from 1 up to 1000000.

10.3.5. Scale interval

The *'scale interval'* is the minimal difference between two consecutive indicated values (either gross or net).

Admitted values : 1/2/5/10/20/50/100

10.3.6. Zero calibration

Zero calibration value corresponds to the A/D converter points measured during the '*zero acquisition*' step of a physical calibration.

For a theoretical calibration this value must be set. It can be set automatically with the '*zero adjustment*' command.

Note: To be applied, any modification of this setting must be followed by an EEPROM back up and device reboots (hardware or software).

Admitted values : from 0 up to +-1000000

10.3.7. Span coefficients 1/2/3

These coefficients are computed and written during calibration process. Writing these coefficients could be done if you want to restore a previous calibration.

Note: To be applied, any modification of this setting must be followed by an EEPROM back up and device reboots (hardware or software).

Admitted values : different from 0.

10.3.8. Span adjusting coefficient

The *'span adjusting coefficient'* allows adjusting initial calibration. Adjustment applies linearly on the whole calibration curve. This coefficient has a 10^{-6} factor (1000000 is equivalent to a span adjusting coefficient that is equal to 1).

Note: To be applied, any modification of this setting must be followed by an EEPROM back up and device reboots (hardware or software).

Admitted values : from 900000 up to 1100000.

10.3.9. Calibration place g value / place of use g value

When the calibration place and the place of use of a measuring chain are different, a deviation can appear due to the difference of g (gravity) between the 2 places.

The eNod4 calculates a ratio applied to the measure which compensates the difference of gravity between the 2 places.

The g value are expressed in 10^{-6} m.s^{-2} (9805470 is equivalent to g = 9.805470 m.s^{-2}).

The ***eNodView*** software can help to determine the g value of a place.

Note: To be applied, any modification of this setting must be followed by an EEPROM back up and device reboots (hardware or software).

Admitted values : different from 0.

10.3.10. Zero offset

The *'Zero offset'* value contains the offset in factory calibrated points that can be added/subtracted (if its value is positive or negative) to the zero calibration value when using the *'zero offset'* functional command. Once the command has been successfully achieved, this register is set to 0.

Note: The 'Zero offset' value is not stored into EEPROM memory and is always equal to 0 after a device power-up or a software reset

Admitted values : different from 0.

11. FILTERS

Name	Modbus address	Ethernet/IP Class/ Attribute (hex/dec)	Profinet Record Index	Profinet cyclic Req Code	Type	Access
A/D conversion rate	*0x0036*	*0x66/1*	*0x0030*	*R:0x0240* *W:0x0241*	*Uint*	*RW*
filters activation	*0x0037 LSB*	*0x66/2 LSB*	*0x0031 LSB*	*R:0x0242 LSB* *W:0x0243 LSB*	*Byte*	*RW*
Low-pass order	*0x0037 MSB*	*0x66/2 MSB*	*0x0031MSB*	*R:0x0242 MSB* *W:0x0243 MSB*	*Byte*	*RW*
Low-pass cut-off frequency	*0x0038*	*0x66/3*	*0x0032*	*R:0x0244* *W:0x0245*	*Uint*	*RW*
Band-stop high cut-off frequency	*0x0039*	*0x66/4*	*0x0033*	*R:0x0246* *W:0x0247*	*Uint*	*RW*
Band-stop low cut-off frequency	*0x003A*	*0x66/5*	*0x0034*	*R:0x0248* *W:0x0249*	*Uint*	*RW*

11.1. Principles

eNod4 contains 4 filtering layers that are user-configurable :

- filtering related to the A/D conversion rate (with rejection of the mains frequency)
- a low-pass Bessel-type filter
- a band-stop filter
- a self-adaptive filter

Except for the A/D conversion rate that is always enabled, none of these filters is mandatory. However, to perform accurate measurements we recommend setting a combination of filters. ***eNodView*** software may be helpful in designing the best filter configuration for the application.

11.2. Settings list

Here is the list of the settings that have an impact on the filters configuration:

11.3. Settings description

11.3.1. A/D conversion rate

It contains a code which represents the A/D conversion rate and the rejection. See table below:

b_4	Rejection
0	*60 Hz*
1	*50 Hz*

$b_3\ b_2\ b_1\ b_0$	A/D conversion rate (measures/s)	
	50-Hz rejection	60-Hz rejection
0000	*100*	*120*
0001	*50*	*60*
0010	*25*	*30*
0011	*12.5*	*15*
0100	*6.25*	*7.5*
1001	*1600*	*1920*
1010	*800*	*960*
1011	*400*	*480*
1100	*200*	*240*

Note: To be applied, any modification of this setting must be followed by an EEPROM back up and device reboots (hardware or software).

11.3.2. Filters activation & order

This setting allows to define what filters are enabled in ***eNod4*** signal processing chain.

Note : the filters activation & order setting can be accessed through a 16-bits register except in CANopen® communication protocol where this word is divided into 2 8-bits registers :

b_0	*Meaning*
0	*band-stop filter disabled*
1	*band-stop filter enabled*
b_1	
0	*self-adaptive filter disabled*
1	*self-adaptive filter enabled*
$b_{10}\ b_9\ b_8$	
000	*low-pass filter disabled*
010	*2nd order low-pass filter*
011	*3rd order low-pass filter*
100	*4th order low-pass filter*

Note: In CANopen® communication protocol (according to version), this word is divided into 2 bytes of 8-bits registers. Bits b8 to b15 are therefore equivalent to bits b0 to b7 of the corresponding address (see CANopen® Register table).

11.3.3. Low-pass filter cut-off frequency

This register contains the low-pass filter cut-off frequency expressed in Hz and multiplied by 100. That means that 690 is equivalent to 6.90 Hz. The value must be compliant with the table shown in §11.4.

Admitted values : from 10 up to 20000.

11.3.4. Band-stop filter high cut-off frequency

This register contains the band-stop filter high cut-off frequency expressed in Hz and multiplied by 100. That means that 690 is equivalent to 6.90 Hz. **The value must be higher than the band-stop filter low cut-off frequency.**

Admitted values : from 10 up to 20000.

11.3.5. Band-stop filter low cut-off frequency

This register contains the band-stop filter low cut-off frequency expressed in Hz and multiplied by 100. That means that 690 is equivalent to 6.90 Hz. **The value must be lower than the band-stop filter high cut-off frequency.**

Admitted values : from 10 up to 20000.

11.4. Limitations

Recursive filters like ***eNod4*** low-pass filters are computed according to the filter order, the desired cut-off frequency and the sampling rate. There are some limitations to respect in order to ensure a safe functioning of the signal processing. They are listed in the table below :

A/D conversion rate (meas/s)		min low-pass cut-off frequency (Hz)		A/D conversion rate (meas/s)		min low-pass cut-off frequency (Hz)	
	50 Hz rejection				60 Hz rejection		
	2nd order	3rd order	4th order		2nd order	3rd order	4th order
6.25	0.10	0.10	0.10	**7.5**	0.10	0.10	0.15
12.5	0.10	0.10	0.15	**15**	0.10	0.15	0.20
25	0.10	0.15	0.25	**30**	0.15	0.20	0.30
50	0.15	0.25	0.50	**60**	0.20	0.30	0.60
100	0.25	0.50	1.00	**120**	0.30	0.60	1.20
200	0.50	1.00	2.00	**240**	0.60	1.20	2.40
400	1.00	2.00	4.00	**480**	1.20	2.40	4.80
800	2.00	4.00	8.00	**960**	2.40	4.80	9.60
1600	4.00	8.00	16.00	**1920**	4.80	9.60	19.20

12. CONFIGURATION OF LOGICAL INPUT/OUTPUT

Name	*Modbus address*	*Ethernet/IP Class/ Attribute (hex/dec)*	*Profinet Record Index*	*Profinet cyclic Req Code*	*Type*	*Access*
Logical input 1 functioning	*0x0042 LSB*	*0x67/1 LSB*	*0x0040 LSB*	*R:0x0250 LSB* *W:0x0251 LSB*	*Byte*	*RW*
Logical input 2 functioning	*0x0042 MSB*	*0x67/1 MSB*	*0x0040 MSB*	*R:0x0250 MSB* *W:0x0251 MSB*	*Byte*	*RW*
holding time	*0x0043*	*0x67/2*	*0x0041*	*R:0x0252* *W:0x0253*	*Uint*	*RW*
Output 1 functioning	*0x0044 LSB*	*0x67/3 LSB*	*0x0050 LSB*	*R:0x0254 LSB* *W:0x0255 LSB*	*Byte*	*RW*
Output 2 functioning	*0x0044 MSB*	*0x67/3 MSB*	*0x0050 MSB*	*R:0x0254 MSB* *W:0x0255 MSB*	*Byte*	*RW*
Output 3 functioning	*0x0045 LSB*	*0x67/4 LSB*	*0x0051 LSB*	*R:0x0256 LSB* *W:0x0257 LSB*	*Byte*	*RW*
Output 4 functioning	*0x0045 MSB*	*0x67/4 MSB*	*0x0051 MSB*	*R:0x0256 MSB* *W:0x0257 MSB*	*Byte*	*RW*
Set point 1 high value	*0x0046*	*0x67/5*	*0x0052*	*R:0x045A* *W:0x045B*	*Long*	*RW*
Set point 1 low value	*0x0048*	*0x67/6*	*0x0053*	*R:0x045C* *W:0x045D*	*Long*	*RW*
Set point 2 high value	*0x004A*	*0x67/7*	*0x0054*	*R:0x045E* *W:0x045F*	*Long*	*RW*
Set point 2 low value	*0x004C*	*0x67/8*	*0x0055*	*R:0x0460* *W:0x0461*	*Long*	*RW*
Set point 3 high value	*0x004E*	*0x67/9*	*0x0056*	*R:0x0462* *W:0x0463*	*Long*	*RW*
Set point 3 low value	*0x0050*	*0x67/10*	*0x0057*	*R:0x0464* *W:0x0465*	*Long*	*RW*
Set point 4 high value	*0x0052*	*0x67/11*	*0x0058*	*R:0x0466* *W:0x0467*	*Long*	*RW*
Set point 4 low value	*0x0054*	*0x67/12*	*0x0059*	*R:0x0468* *W:0x0469*	*Long*	*RW*
1&2 Set points functioning	*0x0056 LSB*	*0x67/13 LSB*	*0x005A LSB*	*R:0x0258 LSB* *W:0x0259 LSB*	*Byte*	*RW*
3&4 Set points functioning	*0x0056 MSB*	*0x67/13 MSB*	*0x005A MSB*	*R:0x0258 MSB* *W:0x0259 MSB*	*Byte*	*RW*
Input level	*0x0094 LSB*	*/*	*/*	*/*	*Byte*	*RO*
Output level	*0x0094 MSB*	*/*	*/*	*/*	*Byte*	*RO*

12.1. Principles

eNod4 is equipped with 2 logical inputs and 4 logical outputs that are fully configurable.

12.1.1. Logical inputs

Each input can work individually in either positive or negative logic. A holding time attached to both inputs can be configured. The available functions are:

- **None:** the input has no function

- **Tare:** a rising (positive logic) or a falling edge (negative logic) causes a tare function to be triggered.
- **Zero:** a rising (positive logic) or a falling edge (negative logic) causes a zero function to be triggered.
- **Cancel tare:** a rising (positive logic) or a falling edge (negative logic) causes the current stored tare to be erased.
- **Transmit measurement:** (depend on communication protocol). A rising (positive logic) or a falling edge (negative logic) triggers a measurement transmission.
- **Measurement window:** only available in SCMBus/fast SCMBus protocols. Measurements are continuously transmitted at a rate defined by the *'sampling period'* while the input is maintained at the defined level.

12.1.2. Logical outputs

The available functions are:

- **None:** the output has no function assigned.
- **Motion:** the output is assigned to copying the stability flag level.
- **Defective measurement:** the output level is set by the logical OR operation between the various defects concerning measurement range.
- **Set point :** each output can be assigned to a configurable set point (set point 1 corresponds to output 1, set point 2 to output 2, set point 3 to output 3 and set point 4 to output 4) .
- **Input image:** the output is at the same level as the logical input level (outputs 1 and 3 correspond to input 1, outputs 2 and 4 correspond to input 2).
- **Level on request:** the input level is set on master requests.

12.2. Settings description

12.2.1. Logical inputs assignment

The following tables describe the possible assignments.

Bits	Meaning	Note
$b_3 b_2 b_1 b_0$	*input 1 assignment*	
0000	*none*	*the input has no function*
0001	*tare*	
0010	*zero*	
0011	*cancel tare*	
0100	*transmit measurement* *(note 1)*	*data is transmitted on the bus at every rising or falling edge (depending on the chosen logical)*
0101	*measurement window* *(note 1)*	*data is transmitted on the bus while the input is maintained at the right level (depending on the chosen logical). Transmission rate is fixed by the 'sampling rate' setting*
b_4	*input 1 logical*	
0	*negative logic*	*defines the edge (or level) that triggers input 1 function*
1	*positive logic*	
$b_6 b_5$	*measurement to be transmitted (note 1)*	
00	*gross*	
01	*net*	
10	*factory calibrated points*	

Bits	Meaning	Note
b_{11} b_{10} b_9 b_8 (note 2)	***input 2 assignment***	
0000	*none*	*the input has no function*
0001	*tare*	
0010	*zero*	
0011	*cancel tare*	
0100	*transmit measurement* *(note 1)*	*data is transmitted on the bus at every rising or falling edge (depending on the chosen logical)*
0101	*measurement window* *(note 1)*	*data is transmitted on the bus while the input is maintained at the right level (depending on the chosen logical). Transmission rate is fixed by the 'sampling rate' setting*
b_{12} (note 2)	***input 2 logical***	
0	*negative logic*	*defines the edge (or level) that triggers input 1 function*
1	*positive logic*	
b_{14} b_{13} (note 2)	***measurement to be transmitted (note 1)***	
00	*gross*	
01	*net*	
10	*factory calibrated points*	

Note 1: Only for SCMBus and fast SCMBus protocols communication, no effect otherwise.

Note 2: In CANopen® communication protocol (according to version), this word is divided into 2 bytes of 8-bits registers. Bits b8 to b15 are therefore equivalent to bits b0 to b7 of the corresponding address (see CANopen® Register table).

12.2.2. Holding time

The holding time corresponds to the minimum required stabilization time of the logical inputs before their activation. If the input level varies within this interval, the assigned command is ignored.

12.2.3. Logical outputs 1&2 assignment

The following table describes the possible assignments.

Bits	Meaning	Note
b3 b2 b1 b0	**output 1 assignment**	
0000	none	the output level does not vary
0001	set point 1	functioning described by the 'set point functioning' setting and by the 'set point 1 high and low values'
0010	motion	copies the motion flag of the status bytes
0011	defective measurement	error flag representing the OR logical operation between the error bits of the status bytes
0100	input 1 image	copies input 1 level
0101	level on request	output 1 level is driven by the 'OUT1 activation/deactivation' functional command
b4	**output 1 logical**	
0	negative logic	defines the output level when enabled
1	positive logic	
b11 b10 b9 b8(note 1)	**output 2 assignment**	
0000	none	the output level does not vary
0001	set point 2	functioning described by the 'set point functioning' setting and by the 'set point 2 high and low values'
0010	motion	copies the motion flag of the status bytes
0011	defective measurement	error flag representing the OR logical operation between the error bits of the status bytes
0100	input 2 image	copies input 2 level
0101	level on request	output 2 level is driven by the 'OUT2 activation/deactivation' functional command
b12 (note 1)	**output 2 logical**	
0	negative logic	defines the output level when enabled
1	positive logic	defines the output level when enabled

Note 1: In CANopen® communication protocol (according to version), this word is divided into 2 bytes of 8-bits registers. Bits b8 to b15 are therefore equivalent to bits b0 to b7 of the corresponding address (see CANopen® Register table).

12.2.4. Logical outputs 3&4 assignment

The assignment is similar to the outputs 1&2 configuration parameter, see previous paragraph (replacing all references to output 1 with output 3 and all references to output 2 with output 4).

12.2.5. Set points functioning

The following table describes the possible assignments.

Bits	Meaning	Note
b0	**set point 1 commutation mode**	
0	*window*	*only if output 1 assigned to the 'set point' function*
1	*hysteresis*	
b2 b1	**set point 1 comparison measurement**	
00	*gross*	
01	*net*	
b3	**reserved (0)**	
b4	**set point 2 commutation mode**	
0	*window*	*only if output 2 assigned to the 'set point' function*
1	*hysteresis*	
b6 b5	**set point 2 comparison measurement**	
00	gross	
01	net	
b7	**reserved (0)**	
b8 (note 1)	**set point 3 commutation mode**	
0	*window*	*only if output 3 assigned to the 'set point' function*
1	*hysteresis*	
b10 b9 (note 1)	**set point 3 comparison measurement**	
00	*gross*	
01	*net*	
b11 (note 1)	**reserved (0)**	
b12 (note 1)	**set point 4 commutation mode**	
0	*window*	*only if output 4 assigned to the 'set point' function*
1	*hysteresis*	
b14 b13 (note 1)	**set point 4 comparison measurement**	
00	*gross*	
01	*net*	
b15 (note 1)	**reserved (0)**	

Note 1: In CANopen® communication protocol (according to version), this word is divided into 2 bytes of 8-bits registers. Bits b8 to b15 are therefore equivalent to bits b0 to b7 of the corresponding address (see CANopen® Register table).

12.2.6. Set points high and low values

Each set point can be configured by its commutation mode (hysteresis/window) and by a couple of values that are continuously compared to the gross or net measurement (depending on the configuration of the set point) in order to define the corresponding output logical level.

For more details about the set points functioning, see the *user manual, characteristics and functioning*.

Admitted values : from -1000000 to 1000000.

12.3. Input/output level

The level of the eNod4 Input/output can be read according to the following table:

Bits	Meaning	Note
b0		
0	low	IIN1 level
1	high	
b1		
0	low	IN2 level
1	high	
b7 b2		
0	reserved (0)	
b8 (note 1)		
0	low	OUT1 level
1	high	
b9 (note 1)		
0	low	OUT2 level
1	high	
b10 (note 1)		
0	low	OUT3 level
1	high	
b11 (note 1)		
0	low	OUT4 level
1	high	
b15 ... b12 (note 1)		
0	reserved (0)	

Note 1: In CANopen® communication protocol (according to version), this word is divided into 2 bytes of 8-bits registers. Bits b8 to b15 are therefore equivalent to bits b0 to b7 of the corresponding address (see CANopen® Register table).

13. LEGAL FOR TRADE OPTIONS

Name	Modbus address	Ethernet/IP Class/ Attribute (hex/dec)	Profinet Record Index	Profinet cyclic Req Code	Type	Access
Legal for trade version	*0x0004 LSB*	*0x64/3*	*0x0010 LSB*	*R: 0x0210 LSB* *W: /*	*Byte*	*RO*
Legal for trade switch	*0x0004 MSB*	*0x64/4*	*0x0010 MSB*	*R: 0x0210 MSB* *W: 0x0211MSB*	*Byte*	*RW*
Legal for trade counter	*0x0005*	*0x64/5*	*0x0011*	*R: 0x0212* *W: /*	*Byte*	*RO*
Legal for trade checksum	*0x0006*	*0x64/6*	*0x0012*	*R: 0x0214* *W: /*	*Uint*	*RO*
Zero functions	*0x0007*	*0x64/7*	*0x0013*	*R:0x0216* *W:0x0217*	*Uint*	*RW*
Stability criterion	*0x0008 LSB*	*0x64/8 LSB*	*0x0014 LSB*	*R:0x0218 LSB* *W:0x0219 LSB*	*Byte*	*RW*
decimal point position	*0x0008 MSB*	*0x64/8 MSB*	*0x0014 MSB*	*R:0x0218 MSB* *W:0x0219 MSB*	*Byte*	*RW*
Unit	*0x0009*	*0x64/9*	*0x0015*	*R:0x041A* *W:0x041B*	*String*	*RW*

13.1. Principles

The legal for trade options are a set of functions and indicators that are generally used in weighing applications. They have an impact on the device behavior regarding the metrological requirements and track every configuration change that may affect the measurement determination.

13.2. Settings description

12.1.1 Legal for trade switch

This setting activates (b_0 bit set to 1) or deactivates (b_0 bit set to 0) criteria and parameters related to the use of ***eNod4*** in OIML compliance.

The *'legal for trade'* option activation leads to the following changes:

- the *'legal for trade counter'* is incremented every time a storage into EEPROM is requested if one or several metrological settings have been modified.
- a new *'legal for trade checksum'* value is calculated every time a storage into EEPROM is requested if one or several metrological settings have been modified (cf. §11.3.3).
- taring is now impossible if gross measurement is negative.
- the measurement value variations cannot be read during the 15 seconds that follow the device reset (error frame in Modbus RTU, value set to -1 in CANopen® and in Profibus DP) and during zero and tare acquisitions

12.1.2 Legal for trade software version

This RO value identifies the version of the part of the software that is dedicated to the metrology and the measurement exploitation.

12.1.3 Legal for trade counter

If the *'legal for trade'* option is enabled, the legal for trade counter is incremented every time a backup into EEPROM is requested if at least one (or several) of these settings has been modified:

- legal for trade switch
- stability criterion
- decimal point position
- maximum capacity
- number of calibration segments
- calibration loads 1/2/3
- scale interval
- span adjusting coefficient
- calibration place/place of use g values
- sensitivity
- A/D conversion rate
- filtering configuration (activation option, order and cut-off frequencies)
- unit
- zero functions

12.1.4 Legal for trade checksum

If the 'legal for trade' option is enabled, a new legal for trade checksum is calculated every time a backup into EEPROM is requested if at least one (or several) of the settings listed above has been modified.

12.1.5 Zero functions

The zero tracking and the initial zero setting can be respectively enabled by setting b_0 bit or b_1 bit to 1. When activated, both options are effective on a ±10% range of the *'maximum capacity'* (±2% if the *'legal for trade'* option is enabled).

When the initial zero is used, you must use a stability criterion other than 0 to be not affected by transient effects at power-up.

12.1.6 Stability criterion

The stability criterion defines the interval on which measurements are considered as stable. Motion is indicated by b_4 bit of the measurement status register. A measurement is stable if X consecutive measurements following the reference measurement are included in the stability interval (see following table) else the current measurement becomes the new reference measurement. X depends on the A/D conversion rate.

Bits b2 b1 b0	*Stability criterion*	*Note*
000	*no motion detection (always stable)*	
001	*0,25d*	*1d = 1 scale interval*
010	*0,5d*	
011	*1d*	
100	*2d*	

A/D conversion rate (meas/s)		*X*
50-Hz rejection	*60-Hz rejection*	
6,25	*7,5*	*1*
12,5	*15*	*2*
25	*30*	*3*
50	*60*	*5*
100	*120*	*9*
200	*240*	*17*
400	*480*	*33*
800	*960*	*65*
1600	*1920*	*129*

12.1.7 Decimal point position

Although ***eNod4*** measurements are integer values it is however possible to store a '*decimal point position'* so as to design a display related to the application. Its value represents the number of decimal digits. If the variable is set to Zero, it means that decimal point is not used.

Note: the decimal point is directly integrated to SCMBus protocol frames (see § SCMBus).

Admitted values : from 0 up to 7.

12.1.8 Unit

It is possible to store the display unit into the ***eNod4***.

Note: the unit is directly integrated to SCMBus protocol frames (see § SCMBus).

14. PROFINET IO

Chapter	Name	Access	Size in bytes	Standard for Read Write Parameter via Profinet RPC Record Index	Substitute for Read Parameters via Profinet Cyclic Transaction Req	Substitute for Write Parameters via Profinet Cyclic Transaction Req
Legal for trade	Legal for trade switch and version	RW	2	0x0010	0x0210	0x0211
Legal for trade	Legal for trade counter	RO	2	0x0011	0x0212	/
Legal for trade	Legal for trade checksum	RO	2	0x0012	0x0214	/
Legal for trade	Zero functions	RW	2	0x0013	0x0216	0x0217
Legal for trade	Stability criterion / decimal point position	RW	2	0x0014	0x0218	0x0219
Legal for trade	Unit	RW	4	0x0015	0x041A	0x041B
Calibration	Maximum capacity	RW	4	0x0020	0x0420	0x0421
Calibration	Number of calibration segments	RW	2	0x0021	0x0222	0x0223
Calibration	Calibration load 1	RW	4	0x0022	0x0424	0x0425
Calibration	Calibration load 2	RW	4	0x0023	0x0426	0x0427
Calibration	Calibration load 3	RW	4	0x0024	0x0428	0x0429
Calibration	Sensor sensitivity	RW	4	0x0025	0x042A	0x042B
Calibration	Scale interval	RW	2	0x0026	0x022C	0x022D
Calibration	Zero calibration	RW	4	0x0027	0x0434	0x0435
Calibration	Span adjusting coefficient	RW	4	0x0028	0x042E	0x042F
Calibration	Calibration place g value	RW	4	0x0029	0x0430	0x0431
Calibration	Place of use g value	RW	4	0x002A	0x0432	0x0433
Calibration	Span coefficient 1	RW	4	0x002B	0x0436	0x0437
Calibration	Span coefficient 2	RW	4	0x002C	0x0438	0x0439
Calibration	Span coefficient 3	RW	4	0x002D	0x043A	0x043B
Calibration	Zero offset	RW	4	0x002E	0x0472	0x0473
Filter	A/D conversion rate	RW	2	0x0030	0x0240	0x0241
Filter	Low-pass order / filters activation	RW	2	0x0031	0x0242	0x0243
Filter	Low-pass cut-off frequency	RW	2	0x0032	0x0244	0x0245
Filter	Band-stop high cut-off frequency	RW	2	0x0033	0x0246	0x0247
Filter	Band-stop low cut-off frequency	RW	2	0x0034	0x0248	0x0249
I/O	Logical inputs functioning	RW	2	0x0040	0x0250	0x0251

I/O	holding time	RW	2	0x0041	0x0252	0x0253
I/O	Outputs 1 & 2 functioning	RW	2	0x0050	0x0254	0x0255
I/O	Outputs 3 & 4 functioning	RW	2	0x0051	0x0256	0x0257
I/O	Set point 1 high value	RW	4	0x0052	0x045A	0x045B
I/O	Set point 1 low value	RW	4	0x0053	0x045C	0x045D
I/O	Set point 2 high value	RW	4	0x0054	0x045E	0x045F
I/O	Set point 2 low value	RW	4	0x0055	0x0460	0x0461
I/O	Set point 3 high value	RW	4	0x0056	0x0462	0x0463
I/O	Set point 3 low value	RW	4	0x0057	0x0464	0x0465
I/O	Set point 4 high value	RW	4	0x0058	0x0466	0x0467
I/O	Set point 4 low value	RW	4	0x0059	0x0468	0x0469
I/O	Set points functioning	RW	2	0x005A	0x0258	0x0259
Measurement	Tare value	RO	4	0x0060	0x0470	/
Measurement	Preset tare value	RW	4	0x0061	0x0496	0x0497

Error Type	*Diagnostic Name*	*Diagnostic Help*
4197	*Input analog signal out of the A/D conversion range (negative quadrant)*	*Possible Cause: Short circuit on sensor connection.*
4198	*Input analog signal out of the A/D conversion range (positive quadrant)*	*Possible Cause: Short circuit on sensor connection.*
4199	*Gross meas. < (- max capacity)*	*Cause: The value of the gross measurement exceeds the opposed maximum capacity minus 9 divisions.*
4200	*Gross meas. > (max capacity)*	*Cause: The value of the gross measurement exceeds the maximum capacity plus 9 divisions.*
4201	*Default EEPROM*	*Cause: Error of checksum while reading EEPROM after reset.*

15. ETHERNET / IP REGISTER MAP

Chapter	Name	Ethernet/IP Class	Ethernet/IP Attribute (dec)	Type	Service
	Class 0x64 (100d) / Instance 1				***Get Attribute All***
Modbus	Firmware revision	0x64	1	Uint	Get Attribute Single
Modbus	Node number / baud rate	0x64	2	Uint	Get Attribute Single
Legal for trade	Legal for trade version	0x64	3	Byte	Get Attribute Single
Legal for trade	Legal for trade switch	0x64	4	Byte	Get Attribute Single / Set Attribute Single
Legal for trade	Legal for trade counter	0x64	5	Byte	Get Attribute Single
Legal for trade	Legal for trade checksum	0x64	6	Uint	Get Attribute Single
Legal for trade	Zero functions	0x64	7	Uint	Get Attribute Single / Set Attribute Single
Legal for trade	Stability criterion	0x64	8 LSB	Byte	Get Attribute Single / Set Attribute Single
Legal for trade	decimal point position	0x64	8 MSB	Byte	Get Attribute Single / Set Attribute Single
Legal for trade	Unit	0x64	9	String	Get Attribute Single / Set Attribute Single
	0x65 (101d) / Instance 1				***Get Attribute All***
Calibration	Maximum capacity	0x65	1	Ulong	Get Attribute Single / Set Attribute Single
Calibration	Number of calibration segments	0x65	2	Uint	Get Attribute Single / Set Attribute Single
Calibration	Calibration load 1	0x65	3	Ulong	Get Attribute Single / Set Attribute Single
Calibration	Calibration load 2	0x65	4	Ulong	Get Attribute Single / Set Attribute Single
Calibration	Calibration load 3	0x65	5	Ulong	Get Attribute Single / Set Attribute Single
Calibration	Sensor sensitivity	0x65	6	Ulong	Get Attribute Single / Set Attribute Single
Calibration	Scale interval	0x65	7	Uint	Get Attribute Single / Set Attribute Single
Calibration	Zero calibration	0x65	8	Long	Get Attribute Single
Calibration	Span coefficient 1	0x65	9	Float	Get Attribute Single
Calibration	Span coefficient 2	0x65	10	Float	Get Attribute Single
Calibration	Span coefficient 3	0x65	11	Float	Get Attribute Single
Calibration	Span adjusting coefficient	0x65	12	Ulong	Get Attribute Single / Set Attribute Single
Calibration	Calibration place g value	0x65	13	Ulong	Get Attribute Single / Set Attribute Single
Calibration	Place of use g value	0x65	14	Ulong	Get Attribute Single / Set Attribute Single
Calibration	Zero offset	0x65	15	Long	Get Attribute Single / Set Attribute Single
State	Preset tare value	0x65	16	Ulong	Get Attribute Single / Set Attribute Single

Chapter	Name	Ethernet/IP Class	Ethernet/IP Attribute (dec)	Type	Service
Register					
	0x66 (102d) / Instance 1				**Get Attribute All / Set Attribute All**
Filter	A/D conversion rate	0x66	1	Uint	Get Attribute Single / Set Attribute Single
Filter	filters activation	0x66	2 LSB	Byte	Get Attribute Single / Set Attribute Single
Filter	Low-pass order	0x66	2 MSB	Byte	Get Attribute Single / Set Attribute Single
Filter	Low-pass cut-off frequency	0x66	3	Uint	Get Attribute Single / Set Attribute Single
Filter	Band-stop high cut-off frequency	0x66	4	Uint	Get Attribute Single / Set Attribute Single
Filter	Band-stop low cut-off frequency	0x66	5	Uint	Get Attribute Single / Set Attribute Single
	Class 0x67 (103d) / Instance 1				**Get Attribute All / Set Attribute All**
I/O	Logical input 1 functioning	0x67	1 LSB	Byte	Get Attribute Single / Set Attribute Single
I/O	Logical input 2 functioning	0x67	1 MSB	Byte	Get Attribute Single / Set Attribute Single
I/O	holding time	0x67	2	Uint	Get Attribute Single / Set Attribute Single
I/O	Output 1 functioning	0x67	3 LSB	Byte	Get Attribute Single / Set Attribute Single
I/O	Output 2 functioning	0x67	3 MSB	Byte	Get Attribute Single / Set Attribute Single
I/O	Output 3 functioning	0x67	4 LSB	Byte	Get Attribute Single / Set Attribute Single
I/O	Output 4 functioning	0x67	4 MSB	Byte	Get Attribute Single / Set Attribute Single
I/O	Set point 1 high value	0x67	5	Long	Get Attribute Single / Set Attribute Single
I/O	Set point 1 low value	0x67	6	Long	Get Attribute Single / Set Attribute Single
I/O	Set point 2 high value	0x67	7	Long	Get Attribute Single / Set Attribute Single
I/O	Set point 2 low value	0x67	8	Long	Get Attribute Single / Set Attribute Single
I/O	Set point 3 high value	0x67	9	Long	Get Attribute Single / Set Attribute Single
I/O	Set point 3 low value	0x67	10	Long	Get Attribute Single / Set Attribute Single
I/O	Set point 4 high value	0x67	11	Long	Get Attribute Single / Set Attribute Single
I/O	Set point 4 low value	0x67	12	Long	Get Attribute Single / Set Attribute Single
I/O	1&2 Set points functioning	0x67	13 LSB	Byte	Get Attribute Single / Set Attribute Single
I/O	3&4 Set points functioning	0x67	13 MSB	Byte	Get Attribute Single / Set Attribute Single
I/O	Input level	/	/	Byte	
I/O	Output level	/	/	Byte	
	0x68 (104d) / Instance 1				**Get Attribute All**
Command	command register	0x68	1	Uint	Get Attribute Single / Set Attribute Single
Command	response register	0x68	2	Uint	Get Attribute Single

The register "Command register" uses the mechanism of ***eNod4*** functional commands defined in another chapter.

Note: “reset” and “Restore default settings” commands cannot be sent via cyclic and acyclic exchanges immediately after a restart of ***eNod4***. To be able to use these commands, it must first be processed another command (“cancel Tare” for example).

Note: The “Command register” data **must be** set to 0x0000 before each new command.

16. ETHERNET / IP ODVA COMMONLY DEFINED REGISTER MAP

Name	*Ethernet/IP Class*	*Ethernet/IP Attribute*	*Type*	*Service*
Identity Object Class 0x01 (01_d) / Instance 0				***Get Attribute All***
Class Revision	0x01	1	Uint	Get Attribute Single
Max. Class Instance	0x01	2	Uint	Get Attribute Single
Class Max. Attributes	0x01	6	Uint	Get Attribute Single
Class Max. Instance Attributes	0x01	7	Uint	Get Attribute Single
Identity Object Class 0x01 (01_d) / Instance 1				***Get Attribute All***
Vendor ID	0x01	1	Uint	Get Attribute Single / Reset
Device type	0x01	2	Uint	Get Attribute Single / Reset
Product Code	0x01	3	Uint	Get Attribute Single / Reset
Major Revision / Minor Revision	0x01	4	Uint	Get Attribute Single / Reset
status	0x01	5	Uint	Get Attribute Single / Reset
Serial Number	0x01	6	Ulong	Get Attribute Single / Reset
Length (bytes) / Product Name	0x01	7	string (14 bytes)	Get Attribute Single / Reset
State	0x01	8	byte	Get Attribute Single / Reset
Conf. Consist. Value	0x01	9	Uint	Get Attribute Single / Reset
Heart Interval	0x01	10	Uint	/
Assembly Object Class 0x04 (04_d) / Instance 0				
Class Revision	0x04	1	Uint	Get Attribute Single
Max. Class Instance	0x04	2	Uint	Get Attribute Single
Connection Manager Object Class 0x06 (06_d) / Instance 0				
Class Revision	0x06	1	Uint	Get Attribute Single
Max. Class Instance	0x06	2	Uint	Get Attribute Single
Connection Manager Object Class 0x06 (06_d) / Instance 1				***Forward Close / Forward Open***
DLR (Device Level Ring) 0x47 (71_d) / Instance 0				
Class Revision	0x47	1	Uint	Get Attribute Single
DLR (Device Level Ring) Object Class 0x47 (71_d) / Instance 1				***Get Attribute All***
Network Topology	0x47	1	Byte	Get Attribute Single
Network Status	0x47	2	Byte	Get Attribute Single
Active Supervisor Address	0x47	10	Array of 10 bytes	Get Attribute Single
Capability Flags	0x47	12	Ulong	/
QoS (Quality of Service) Object Class 0x48 (72_d) / Instance 0				
Class Revision	0x48	1	Uint	Get Attribute Single

Name	Ethernet/IP Class	Ethernet/IP Attribute	Type	Service
Max. Class Instance	0x48	2	Uint	Get Attribute Single
QoS (Quality of Service) Object Class 0x48 (72d) / Instance 0				
Class Revision	0x48	1	Uint	Get Attribute Single
Max. Class Instance	0x48	2	Uint	Get Attribute Single
QoS (Quality of Service Object Class 0x48 (72_d) / Instance 1				
802.1Q Tag Enable	0x48	1	Byte	Get Attribute Single
DSCP Urgent	0x48	4	Byte	Get Attribute Single
DSCP Scheduled	0x48	5	Byte	Get Attribute Single
DSCP High	0x48	6	Byte	Get Attribute Single
DSCP Low	0x48	7	Byte	Get Attribute Single
DSCP Explicit	0x48	8	Byte	Get Attribute Single
TCP/IP Interface Object Class 0xF5 (245_d) / Instance 0				
Class Revision	0xF5	1	Uint	Get Attribute Single
Max. Class Instance	0xF5	2	Uint	Get Attribute Single
TCP/IP Interface Object Class 0xF5 (245_d) / Instance 1				***Get Attribute All***
Status	0xF5	1	Ulong	Get Attribute Single
Configuration Capability	0xF5	2	Ulong	Get Attribute Single
Configuration Control	0xF5	3	Ulong	Get Attribute Single
Physical Link Object: Struct Path size Uint Path Padded Epath	0xF5	4	Array of n bytes	Get Attribute Single
Interface Configuration: Struct IP address Uint Network mask Uint Gateway address Uint Name server Uint Name server Ulong Domain name String	0xF5	5	Array of n bytes	Get Attribute Single
Host Name	0xF5	6	Array of n bytes	Get Attribute Single
Safety Network Number	0xF5	7	Array of n bytes	/
Time To Live value	0xF5	8	Array of n bytes	/
Multicast configuration	0xF5	9	Array of n bytes	/
Select ACD	0xF5	10	Array of n bytes	Get Attribute Single (01_H)
Last Conflict Detected	0xF5	11	Array of n bytes	Get Attribute Single (01_H)
Ethernet Link Object Class 0xF6 (246_d) / Instance 0				
Class Revision	0xF6	1	Uint	Get Attribute Single
Max. Class Instance	0xF6	2	Uint	Get Attribute Single
Ethernet Link Object Class 0xF6 (246_d) / Instance 1				
Interface Speed	0xF6	1	Ulong	Get Attribute Single

Name	*Ethernet/IP Class*	*Ethernet/IP Attribute*	*Type*	*Service*
Interface Flags	0xF6	2	Ulong	Get Attribute Single
Physical Address	0xF6	3	Array of 6 bytes	Get Attribute Single
Interface Control	0xF6	6	Ulong	Get Attribute Single
Length (byte) / Interface Label	0xF6	10	string	Get Attribute Single

Note:

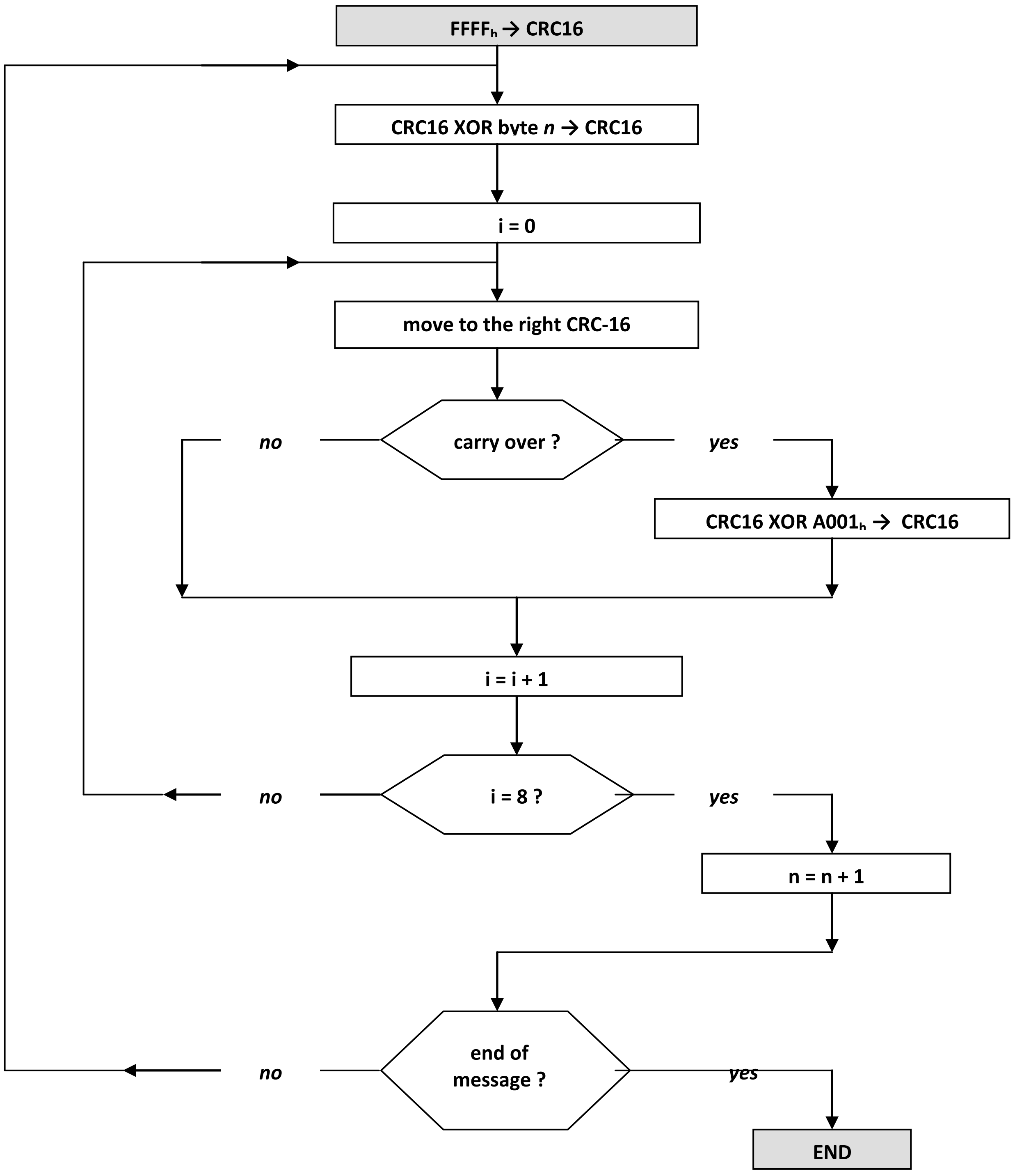
- Get attribute All**: 0x01**
- Get attribute Single**: 0x0E**
- Reset**: 0x05**
- Forward close**: 0x4E**
- Forward open**: 0x54**

17. MODBUS RTU AND MODBUS TCP REGISTER TABLE

Chapter	Name	Modbus Address	Type	Access
Modbus	Firmware revision	0x0000	Uint	RO
Modbus	Node number / baud rate	0x0001	Uint	RO
Legal for trade	Legal for trade version	0x0004 LSB	Byte	RO
Legal for trade	Legal for trade switch	0x0004 MSB	Byte	RW
Legal for trade	Legal for trade counter	0x0005	Byte	RO
Legal for trade	Legal for trade checksum	0x0006	Uint	RO
Legal for trade	Zero functions	0x0007	Uint	RW
Legal for trade	Stability criterion	0x0008 LSB	Byte	RW
Legal for trade	decimal point position	0x0008 MSB	Byte	RW
Legal for trade	Unit	0x0009	String	RW
Calibration	Maximum capacity	0x000C	Ulong	RW
Calibration	Number of calibration segments	0x000E	Uint	RW
Calibration	Calibration load 1	0x000F	Ulong	RW
Calibration	Calibration load 2	0x0011	Ulong	RW
Calibration	Calibration load 3	0x0013	Ulong	RW
Calibration	Sensor sensitivity	0x0015	Ulong	RW
Calibration	Scale interval	0x0017	Uint	RW
Calibration	Zero calibration	0x0018	Long	RO
Calibration	Span coefficient 1	0x001A	Float	RO
Calibration	Span coefficient 2	0x001C	Float	RO
Calibration	Span coefficient 3	0x001E	Float	RO
Calibration	Span adjusting coefficient	0x0020	Ulong	RW
Calibration	Calibration place g value	0x0022	Ulong	RW
Calibration	Place of use g value	0x0024	Ulong	RW
Filter	A/D conversion rate	0x0036	Uint	RW
Filter	filters activation	0x0037 LSB	Byte	RW
Filter	Low-pass order	0x0037 MSB	Byte	RW
Filter	Low-pass cut-off frequency	0x0038	Uint	RW
Filter	Band-stop high cut-off frequency	0x0039	Uint	RW
Filter	Band-stop low cut-off frequency	0x003A	Uint	RW
Protocol	Functioning mode / Serial protocol	0x003E	Uint	RW
SCMBus	SCMBus transmission period	0x003F	Uint	RW
I/O	Logical input 1 functioning	0x0042 LSB	Byte	RW

Chapter	Name	Modbus Address	Type	Access
I/O	Logical input 2 functioning	0x0042 MSB	Byte	RW
I/O	holding time	0x0043	Uint	RW
I/O	Output 1 functioning	0x0044 LSB	Byte	RW
I/O	Output 2 functioning	0x0044 MSB	Byte	RW
I/O	Output 3 functioning	0x0045 LSB	Byte	RW
I/O	Output 4 functioning	0x0045 MSB	Byte	RW
I/O	Set point 1 high value	0x0046	Long	RW
I/O	Set point 1 low value	0x0048	Long	RW
I/O	Set point 2 high value	0x004A	Long	RW
I/O	Set point 2 low value	0x004C	Long	RW
I/O	Set point 3 high value	0x004E	Long	RW
I/O	Set point 3 low value	0x0050	Long	RW
I/O	Set point 4 high value	0x0052	Long	RW
I/O	Set point 4 low value	0x0054	Long	RW
I/O	1&2 Set points functioning	0x0056 LSB	Byte	RW
I/O	3&4 Set points functioning	0x0056 MSB	Byte	RW
State Register	Measurement status	0x007D	Uint	RO
State Register	Gross measurement	0x007E	Long	RO
State Register	Tare value	0x0080	Long	RO
State Register	Net measurement	0x0082	Long	RO
State Register	Factory calibrated points	0x0084	Long	RO
Command	command register	0x0090	Uint	RW
Command	response register	0x0091	Uint	RO
Calibration	Zero offset	0x0092	Long	RW
State Register	Input levels	0x0094 LSB	Byte	RO
I/O	Input levels	0x0094 LSB	Byte	RO
State Register	output levels	0x0094 MSB	Byte	RO
I/O	output levels	0x0094 MSB	Byte	RO
State Register	Preset tare value	0x0095	Ulong	RW

18. CRC-16 CALCULATION ALGORITHM


FFFFh → CRC16
CRC16 XOR byte n → CRC16
i = 0
move to the right CRC-16
carry over ?
no
yes
CRC16 XOR A001h → CRC16
i = i + 1
i = 8 ?
no
yes
n = n + 1
end of message ?
no
yes
END