

ifm electronic



Device manual
Encoder
with DeviceNet interface

UK

efector 400[®]

**RM7
RN7**

08/2014

706373/00

Content

1	Preliminary note	4
1.1	Symbols used	4
1.2	Warning signs used	4
2	Safety instructions	4
3	Functions and features	5
3.1	Encoders	5
3.2	Operating modes	5
4	Installation	6
4.1	Electrical connection	6
4.2	Settings in the terminal cap	7
4.3	Cable	7
4.4	Connector	7
5	Common Industrial Protocol (CIP)	8
5.1	Object model	8
6	Data transmission	9
6.1	Object directory	9
6.2	CAN ID	10
7	Configuration	11
7.1	Operating parameters	11
7.2	Resolution per revolution	11
7.3	Total resolution	12
7.4	Preset value	13
7.5	MAC ID	13
7.6	Baud rate	13
8	Operating modes	14
8.1	Polled Mode	14
8.1.1	Allocate Master / Slave Connection Set	14
8.1.2	Release Master / Slave Connection Set	15
8.2	Change of State Mode	15
8.2.1	Allocate Master / Slave Connection Set	15
8.2.2	Expected_packet_rate of the Explicit Message	16
8.2.3	Expected_packet_rate of the Change of State	16
8.2.4	Release Master / Slave Connection Set	16
8.3	Transfer of stored data	17
9	Transfer of the actual process value	17
10	Set-up	18
10.1	Operating parameters	18
10.2	Resolution per revolution	18
10.3	Total resolution	19
10.4	Preset value	20

10.5 Baud rate	21
10.6 MAC ID	21
10.7 Transfer of stored data	22
11 Configuration in RsNetwork	23
11.1 Install EDS file	23
11.2 Configure the drivers	26
11.3 Put the network online	27
11.4 Configure the encoder	28
12 Terms and abbreviations.	30

Licences and trademarks

Microsoft®, Windows®, Windows XP® and Windows Vista® are registered trademarks of Microsoft Corporation.

All trademarks and company names are subject to the copyright of the respective companies.

Open source software



This unit contains (maybe modified) open source software which is subject to special licensing terms.

For copyright information and licensing terms please refer to: www.ifm.com/int/GNU

For software subject to the GNU General Public License or the GNU Lesser General Public License the source code can be requested against payment of the copying and shipping costs.

1 Preliminary note

1.1 Symbols used

- ▶ Instructions
- > Reaction, result
- [...] Designation of keys, buttons or indications
- Cross-reference
-  Important note
Non-compliance may result in malfunction or interference
-  Information
Supplementary note

1.2 Warning signs used

NOTE

Warning of damage to property

2 Safety instructions

This manual is part of the device. It contains texts and figures concerning the correct handling of the device and must be read before installation or use.

Observe the operating instructions.

Non-observance of the instructions, operation which is not in accordance with use as prescribed below, wrong installation or incorrect handling can affect the safety of operators and machinery.

The installation and connection must comply with the applicable national and international standards. Responsibility lies with the person installing the device.

Only the signals indicated in the technical data or on the device label may be supplied to the connections or wires.

3 Functions and features

3.1 Encoders

The encoders supply one increment for each angular position. The values are shown on one or several code disks as code pattern. The code disks are illuminated by an infrared LED and the resulting bit pattern is detected by an opto array. The resulting signals are electronically amplified and passed on to the interface for processing.

The encoder has a maximum basic resolution of 65536 steps a revolution (16 bits). The multiturn version has up to 16384 revolutions (14 bits). This results in a total resolution of max. 30 bits = 1,073,741,824 increments. The standard singleturn version has 12 bits, the standard multiturn version 24 bits.

UK

3.2 Operating modes

The encoder has the following operating modes:

- Polled Mode
- Change of State Mode

Via the integrated CAN bus the following functions can be programmed:

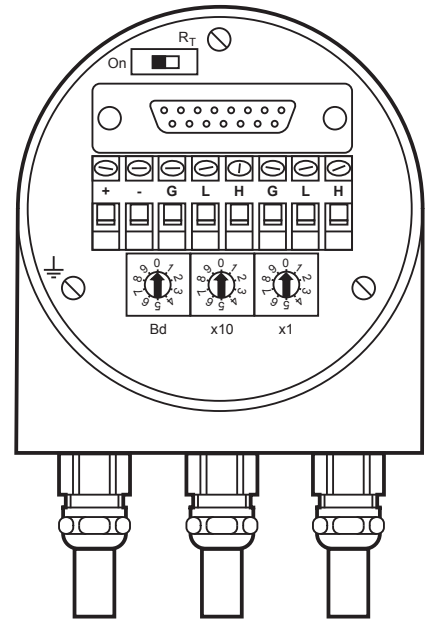
- Direction of rotation (complement)
- Resolution per revolution
- Total resolution
- Preset value
- Baud rate
- MAC ID

4 Installation

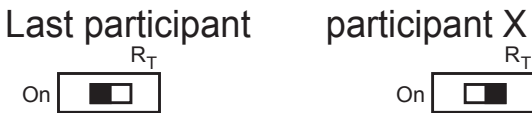
4.1 Electrical connection

The encoder is connected via three cables. The voltage is supplied via a two-wire connection cable and a PG 9 cable gland. The two-wire screened bus cable goes into or out of the encoder via a PG9 cable gland.

Terminal	Description
⊥	Ground
+	24 V voltage supply
-	0 V voltage supply
G	CAN Ground
L	CAN Low
H	CAN High
G	CAN Ground
L	CAN Low
H	CAN High



The terminal cap contains a resistor which can be activated as terminating resistor if necessary. Terminating resistor:



The node number is selected via 2 rotary switches in the terminal cap. The addresses can be between 0 and 63, each address may occur only once. For installation, the user can easily take off the terminal cap by loosening two screws on the encoder. Two diagnostic LEDs on the back of the terminal cap show the operating status of the encoder.

DeviceNet units BCD rotary switch	
x1	Device address 0..63
x10	CAN node number
xBd	Baud rate

4.2 Settings in the terminal cap

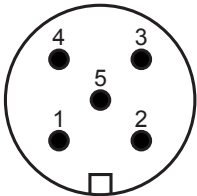
Baud rate in kbit/s	BCD rotary switch
125	0
250	1
500	2
125	3
reserved	4...9

UK

4.3 Cable

Pin	Signal	Description	Colours
1	V-	GND	Black
2	CAN_L	CAN bus signal (dominant low)	Blue
3	CAN_H	CAN bus signal (dominant high)	White
4	V+	External voltage supply Vcc	Red

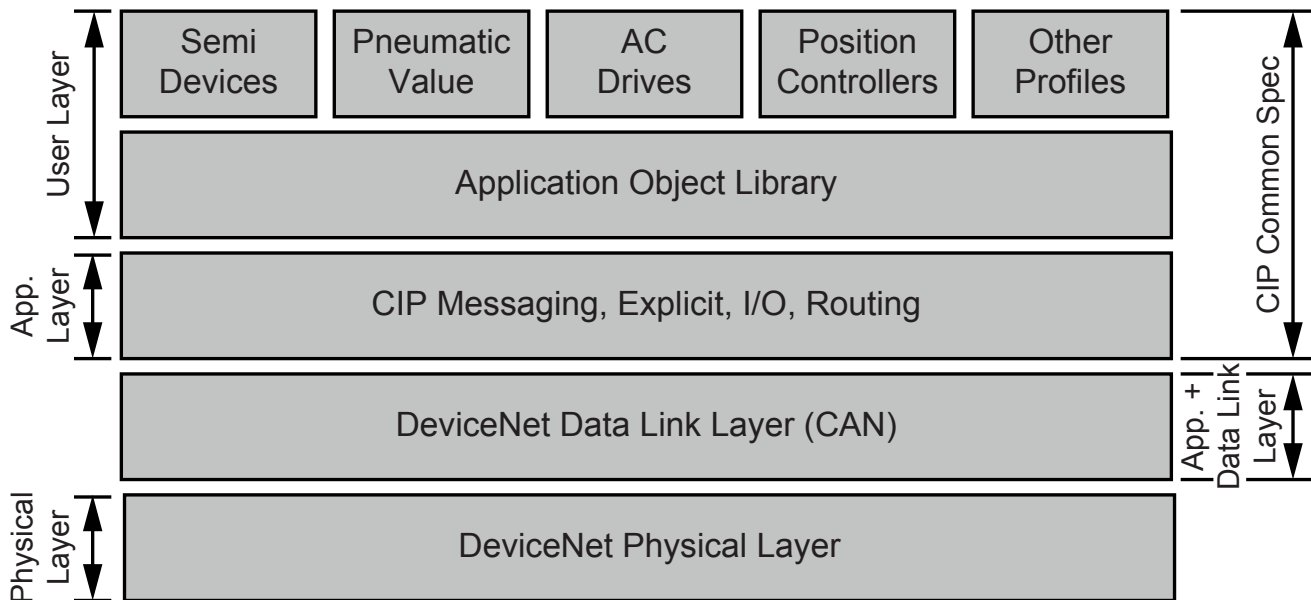
4.4 Connector



5-pole connector

Pin	Signal	Description	Colours
2	V+	External voltage supply Vcc	Red
3	V-	GND	Black
4	CAN_H	CAN bus signal (dominant high)	White
5	CAN_L	CAN bus signal (dominant low)	Blue

5 Common Industrial Protocol (CIP)



The DeviceNet specification defines the application layer and the physical layer. The Data Link layer is based on the CAN specification. For optimum industrial control two different types of messages are available. I/O messages (implicit messaging) and explicit messages (explicit messaging). With implicit messaging I/O data is exchanged in real time and with explicit messaging data is exchanged for the configuration of a device. CIP (Common Industrial Protocol) provides four functions for the user:

- Uniform control services
- Uniform communication services
- Uniform distribution of messages
- Common knowledge base

5.1 Object model

DeviceNet describes all data and functions of a device based on an object model. With this object-oriented description a device can be completely defined using individual objects.

Definition of an object:

- summary of the relevant attributes (e.g. process data),
- functions provided externally (read or write access to an individual attribute),
- defined behaviour.

DeviceNet distinguishes between three types of object:

- **Communication objects** define the messages exchanged via DeviceNet and are called connection objects (DeviceNet object, message router object, connection object, acknowledge handler object).
- **System objects** define general DeviceNet-specific data and functions (identity object, parameter object).
- **Application-specific objects** define device-specific data and functions (application object, assembly object).

UK

6 Data transmission

Data in DeviceNet is transferred via telegrams. In general, the telegrams can be schematically divided into CAN-ID and 8 following bytes:

COB ID	Message Header	Message Body
11 bits	1 byte	7 bytes

6.1 Object directory

The object directory describes instance attributes of the position sensor object.

Class code: 23 hex

Attribute ID	Access	Name	Data length	Description
1 hex	read	Number of attributes	USINT	Number of supported attributes
2 hex	read	Attributes	Array of USINT	List of supported attributes
3 hex	read	Position value	DINT	Output of the current position
70 hex	read / write	Rotational direction control	Boolean	Controls the rising/falling code sequence
71 hex	read / write	Resolution per revolution	INT	Resolution for one revolution
72 hex	read / write	Total resolution	DINT	Total resolution which can be represented unambiguously
73 hex	read / write	Preset value	DINT	Allocation position value
6E hex	read / write	Baud rate		Baud rate setting
6F hex	read / write	MAC ID		Setting of the MAC ID

6.2 CAN ID

DeviceNet is based on the standard CAN protocol and uses an 11-bit message identifier (max. 2048 messages). 6 bits are sufficient to identify a device or a node in a DeviceNet network since one network is limited to 64 participants. This identification is called MAC ID (device or node address). The CAN identifier consists of the identification of the message group, the message ID within this group and the MAC ID of the device.

The absolute encoder is a group 2 server. The following table lists the most important CAN IDs for certain types of communication.

10	9	8	7	6	5	4	3	2	1	0	Identity Usage	Hex Range	
0	Group 1 Message ID				Source MAC-ID						Group 1 Message	000-3 and following	
0	1	1	0	1	Source MAC-ID						Slave I/O Change of State or Cyclic Message		
0	1	1	1	1	Source MAC-ID						Slave I/O Poll Response or Change of State/Cyclic Acknowledge Message		
1	0	MAC ID					Group 2 Message ID				Group 2 Messages	400-5 and following	
1	0	Destination MAC-ID					0	1	0	Master Change of State or Cyclic Acknowledge Message			
1	0	Source MAC-ID					0	1	1	Slave Explicit/Unconnected Response Messages			
1	0	Destination MAC-ID					1	0	0	Master Explicit Request Message			
1	0	Destination MAC-ID					1	0	1	Master I/O Poll Command/Change of State/Cyclic Message			
1	0	Destination MAC-ID					1	1	0	Group 2 Only Unconnected Explicit Request Message (reserved)			
1	0	Destination MAC-ID					1	1	1	Duplicate MAC-ID Check Messages			

7 Configuration

The text below describes how an encoder is configured.

7.1 Operating parameters

The rotational direction can be selected as operating parameter.

Attribute ID	Default value	Value range	Data length
0b hex	1 hex	0hex - 1hex	Boolean

The parameter rotational direction (complement) defines the code sequence of the actual process value if the shaft is rotated clockwise (CW) or counter-clockwise (CCW) when looking at the shaft. The code sequence is defined in the attribute 0bhex:

Bit 0	Rotational direction	Output code
1	CW	Rising
0	CCW	Falling

7.2 Resolution per revolution

The parameter resolution per revolution is used to program the encoder in a way that the selected number of steps referred to one revolution can be achieved.

Attribute ID	Default value	Value range	Data length
2C hex	see type label	0hex - 2000hex	Unsigned Integer16

Maximum resolution for the

24 bit version: 1,000 hex

25 bit version: 2,000 hex

If a value greater than the basic resolution of the encoder is selected for the resolution per revolution, the output value no longer proceeds by one step. It should thus be guaranteed that the selected resolution does not exceed the encoder's resolution.

7.3 Total resolution

The total resolution indicates the selected number of measuring units over the complete travel length. The value must not exceed the total resolution of the encoder. The total resolution of the encoder is indicated on the type label.

Attribute ID	Default value	Value range	Data length
72 hex	see type label	0h - 2,000,000h	Unsigned Integer 32

Maximum resolution for the

24 bit version: 1,000,000 hex

25 bit version: 2,000,000 hex

The following abbreviations are used below:

- PGA Physical total resolution of the encoder (see type label)
- PAU Physical resolution per revolution (see type label)
- GA Total resolution (user input)
- AU Resolution per revolution (user input)

If the selected resolution per revolution is smaller than the real physical resolution of the encoder per revolution, the total resolution has to be entered as follows:

Total resolution

$GA = PGA * AU / PAU$, if $AU < PAU$

Example: User settings: $AU = 2048$,

Encoder values $PGA = 24$ bits, $PAU = 12$ bits

$$GA = 16777216 * 2048 / 4096$$

$$GA = 8388608$$

If the total resolution of the encoder is smaller than the total physical resolution, the parameter total resolution must be an integer multiple of the total physical resolution.

$$k = PGA / GA$$

$k = \text{integer}$

7.4 Preset value

The preset value is the selected position value to be reached in case of a certain physical position of the axis. The parameter Preset value allows to set the actual position value to the selected actual process value. The preset value must not exceed the parameter Total resolution.

Attribute ID	Default value	Value range	Data length
73 hex	0 hex	0hex - total resolution	Unsigned Integer 32

UK

7.5 MAC ID

For absolute encoders without terminal cap, the MAC ID is configured via Explicit Messaging. The DeviceNet encoders address 64 different nodes.

Attribute ID	Default value	Value range	Data length
6F hex	(*)	0hex – 3Fhex	Unsigned Integer8

7.6 Baud rate

For absolute encoders without terminal cap, the baud rate is configured via Explicit Messaging. The DeviceNet encoders support all DeviceNet baud rates of the following table.

Attribute ID	Default value	Value range	Data length
6E hex	(*)	0hex – 2hex	Unsigned Integer8

Bytes	Baud rate
0	125 kbaud
1	250 kbaud
2	500 kbaud

8 Operating modes

8.1 Polled Mode

The Polled Mode is a master/slave communication. The master can query the current actual position value of the encoder via the Poll Command Message. The encoder then transmits the actual process value to the master via a Poll Response Message. In the example a master MAC ID of 0A hex and a slave MAC ID of 03 hex are taken as a basis.

8.1.1 Allocate Master / Slave Connection Set

Allocate Polling

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Frag [0]	XID	MAC ID					
1	R/R [0]	Service [4B]						
	Class ID [03]							
	Instance ID [01]							
	Allocation Choice [03]							
	0	0	Allocator MAC ID					

Definition CAN ID

10	9	8	7	6	5	4	3	2	1	0	Identity Usage	Hex Range
1	0	Destination MAC ID						1	1	0	Group 2 only unconnected explicit request message (reserved)	

Example

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
41E	0A	4B	03	01	03	0A

Set Expected_packet_rate of the Explicit Message Connection to 0

Definition CAN ID

10	9	8	7	6	5	4	3	2	1	0	Identity Usage	Hex Range
1	0	Destination MAC ID						1	0	0	Master explicit request message	

Example

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
41C	0A	10	05	01	09	00	00

Set Expected_packet_rate of the Polling Connection to 0

Example

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
41C	0A	10	05	02	09	00	00

8.1.2 Release Master / Slave Connection Set

Release Polling

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Frag [0]	XID	MAC ID					
1	R/R [0]	Service [4C]						
	Class ID [03]							
	Instance ID [01]							
	Release Choice [03]							

UK

Example

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
41E	0A	4C	03	01	03

8.2 Change of State Mode

The absolute encoder transmits data without query by the host as soon as the current actual process value changes. As long as the process value remains unchanged, there is no transmission. This reduces the bus load.

8.2.1 Allocate Master / Slave Connection Set

Allocate COS

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Frag [0]	XID	MAC ID					
1	R/R [0]	Service [4B]						
	Class ID [03]							
	Instance ID [01]							
	Allocation Choice [51]							
	0	0	Allocator MAC ID					

Example

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
41E	0A	4B	03	01	51	0A

8.2.2 Expected_packet_rate of the Explicit Message

Set the example for the connection to 0

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
41C	0A	10	05	01	09	00	00

8.2.3 Expected_packet_rate of the Change of State

Set the example for the connection to 0

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
41C	0A	10	05	04	09	00	00

8.2.4 Release Master / Slave Connection Set

Release COS

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Frag [0]	XID	MAC ID					
1	R/R [0]	Service [4C]						
	Class ID [03]							
	Instance ID [01]							
	Release Choice [51]							

Example

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
41E	0A	4C	03	01	51

8.3 Transfer of stored data

The encoder is equipped with a flash EPROM which stores the contained data non-volatily. Since a flash EPROM loses its memory capacity after approx. 1000 writing cycles, the modified data is first stored in the working memory. After verification the data is then copied to the flash EPROM.

When the data has been copied successfully to the flash EPROM, the encoder carries out a MAC ID check on the bus. To query the process value, the slave has to be allocated once again.

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Frag [0]	XID	MAC ID					
1	R/R [0]	Service [32]						
	Class ID [23]							
	Instance ID [01]							

UK

Example (MAC ID master: 0A hex, MAC ID slave 03 hex)

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3
41C	0A	32	23	01

9 Transfer of the actual process value

The following telegram describes how the actual process value is transferred.

CAN ID	Actual process value			
11 bits	Byte 0	Byte 1	Byte 2	Byte 3
	2^7 to 2^0	2^{15} to 2^8	2^{23} to 2^{16}	2^{31} to 2^{24}

10 Set-up

After power on, the absolute encoder carries out a MAC ID check on the bus.



For the CAN ID and the MAC ID, 0A (hex) is used for the master and 03 (hex) for the slave as an example. Parameters to remain unchanged can be skipped. The numerical values are indicated as a hexadecimal number.

10.1 Operating parameters

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Service Code	Class ID	Instance ID	Attribute ID	Data		
	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
41C	0A	10	23	01	0B	X	-	-

X: 0 (hex) for CW (standard)

1 (hex) for CCW

Absolute encoder to master: Confirmation

CAN ID	MAC ID	Service Code
	Byte 0	Byte 1
41B	0A	90

10.2 Resolution per revolution

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Service Code	Class ID	Instance ID	Attribute ID	Data		
	Byte 0	Byte 1	Byte 2	byte 3	Byte 4	Byte 5	Byte 6	byte 7
41C	0A	10	23	01	2C	X	X	-

X: Resolution per revolution

Absolute encoder to master: Confirmation

CAN ID	MAC ID	Service Code
	Byte 0	Byte 1
41B	0A	90

10.3 Total resolution

To transfer the total resolution, a fragmented transmission has to be carried out. The following telegrams show how the total resolution is transmitted.

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Fragment	Service Code	Class ID	Instance ID	Attribute ID		
						Byte 5	Byte 6	Byte 7
41C	8A	00	10	23	01	2D	X	X

X: Total resolution

UK

Absolute encoder to master: Confirmation

CAN ID	MAC ID		
	Byte 0	Byte 1	Byte 2
41B	8A	C0	00

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Fragment						
			Byte 2	Byte 3				
41C	8A	81	X	X				

X: Total resolution

Absolute encoder to master: Confirmation

CAN ID	MAC ID		
	Byte 0	Byte 1	Byte 2
41B	8A	C1	00

Absolute encoder to master: Confirmation

CAN ID	MAC ID	Service Code
	Byte 0	Byte 1
41B	0A	90

10.4 Preset value

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Fragment	Service Code	Class ID	Instance ID	Attribute ID		
	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
41C	8A	00	10	23	01	2E	X	X

X: Preset value

Absolute encoder to master: Confirmation

CAN ID	MAC ID		
	Byte 0	Byte 1	Byte 2
41B	8A	C0	00

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Fragment						
	Byte 0	Byte 1	Byte 6	Byte 7				
41C	8A	81	X	X				

X: Total resolution

Absolute encoder to master: Confirmation

CAN ID	MAC ID		
	Byte 0	Byte 1	Byte 2
41B	8A	C1	00

Absolute encoder to master: Confirmation

CAN ID	MAC ID	Service Code
	Byte 0	Byte 1
		90

10.5 Baud rate

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Service Code	Class ID	Instance ID	Attribute ID	Attribute ID		
	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
41C	0A	10	23	01	6E	X	-	-

X: Baud rate

X	Baud rate
0	125 kbaud
1	250 kbaud
2	500 kbaud

UK

Absolute encoder to master: Confirmation

CAN ID	MAC ID	Service Code
	Byte 0	Byte 1
41B	0A	90

10.6 MAC ID

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Service Code	Class ID	Instance ID	Attribute ID	Attribute ID		
	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
41C	0A	10	23	01	6F	X	-	-

X: MAC ID

Absolute encoder to master: Confirmation

CAN ID	MAC ID	Service Code
	Byte 0	Byte 1
41B	0A	90

10.7 Transfer of stored data

Master to absolute encoder: Set parameters

CAN ID	MAC ID	Service Code	Class ID	Instance ID
	Byte 0	Byte 1	Byte 2	Byte 3
41C	0A	32	23	01

If the transmission is successful, the absolute encoder returns a duplicated MAC ID after 2 s. Then the master has to be allocated to the slave again.

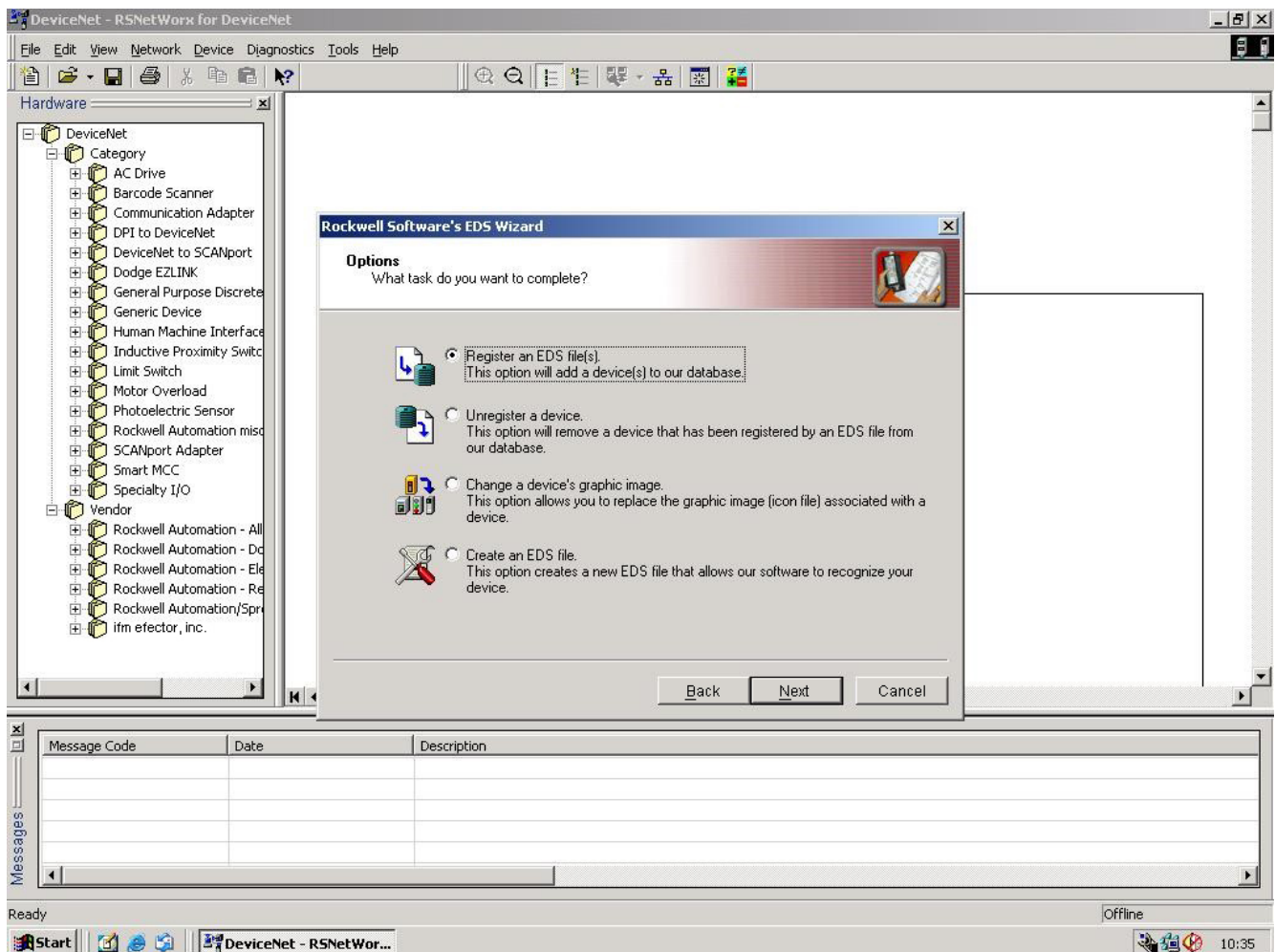
If the transmission is not successful, an error message is displayed.

The service code used to save the values is manufacturer-specific.

11 Configuration in RsNetworx

11.1 Install EDS file

The EDS file contains information on device-specific parameters as well as possible operating modes of the encoder. An electronic data sheet is thus available which can be used for configuration in RsNetworx for example.



UK

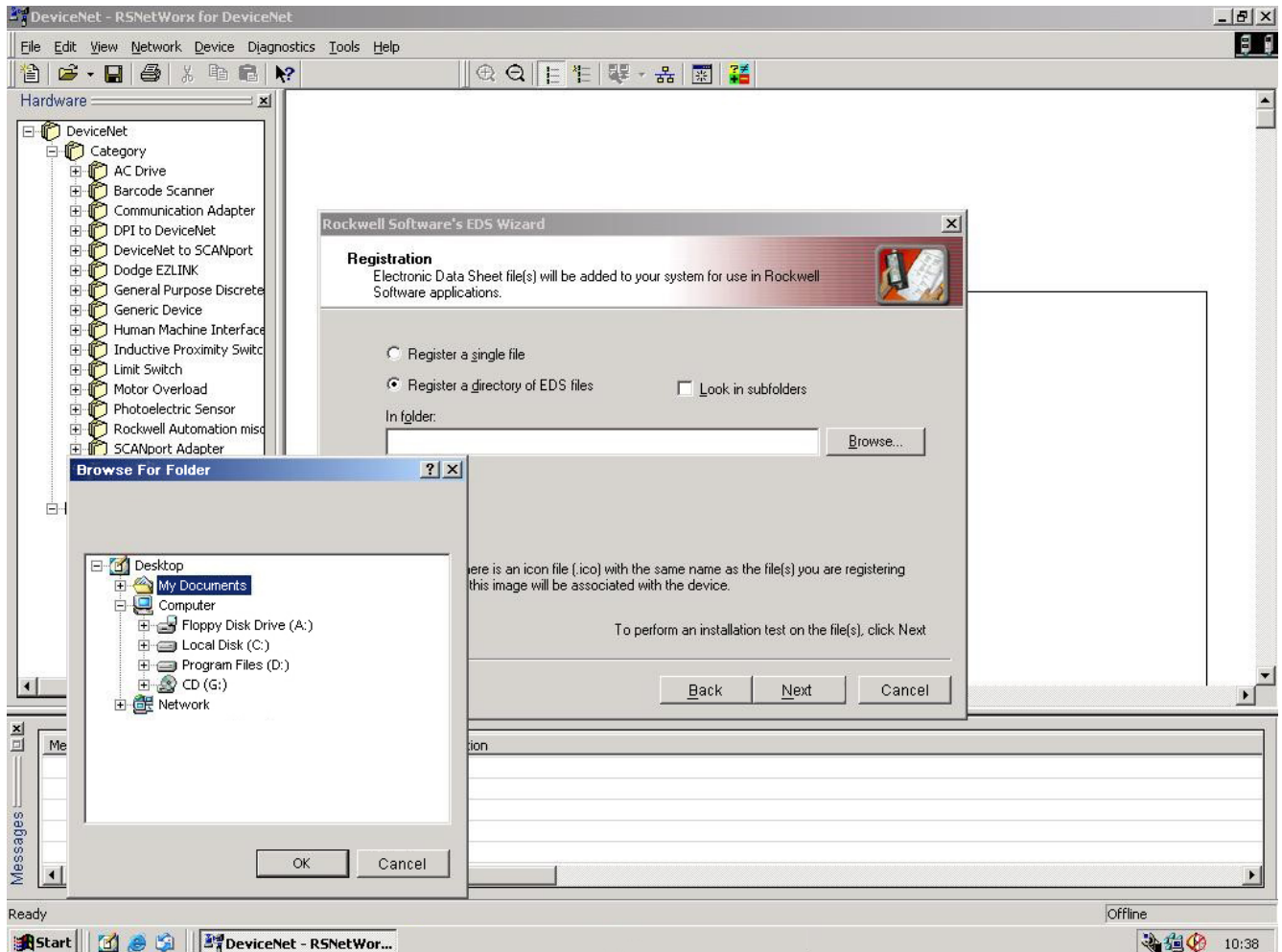
Before an encoder can be connected to the bus, the EDS file has to be installed. The EDS file is installed using the EDS wizard.

To start the EDS wizard, click on "Tools / EDS Wizard" in the menu bar. After successful start, the EDS wizard appears (see screenshot above).

Then select the item "Register an EDS file(s)" and click on "Next".

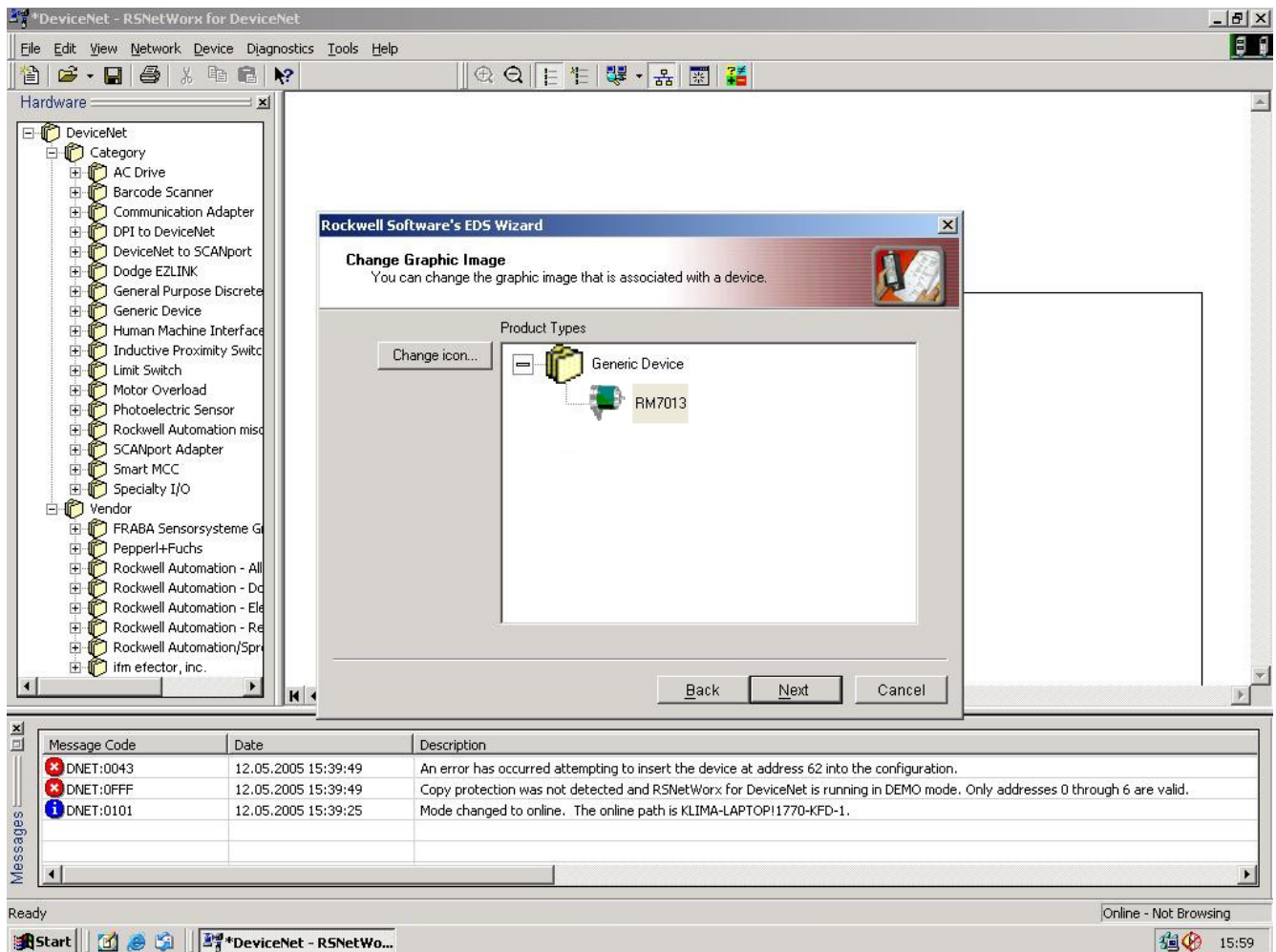
In the next step, select "Register a directory of EDS files" and click on "Browse" to indicate the path of the EDS file (see screenshot below).

The EDS wizard finds all EDS files which are in this path and verifies if any errors occur in the EDS files.



Click on the button "Next" to go to the Change Graphic Image window (see next screenshot).

In the Change Graphic Image window, graphic images are assigned to the nodes used.



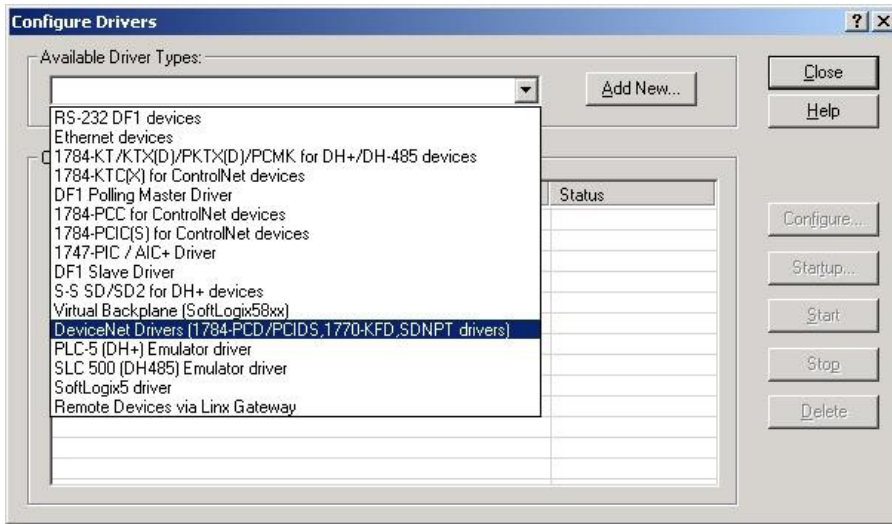
UK

Click on the button "Next" to terminate the installation of the EDS file.

11.2 Configure the drivers

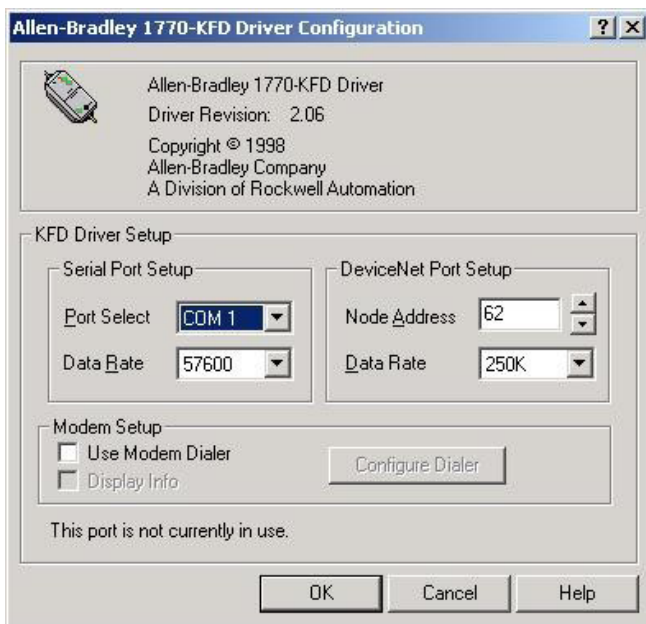
When the EDS file has been installed, the suitable driver is configured. In the following the driver 1770-KFD is configured.

With RSLinx the suitable driver is selected. Start the program RSLinx via "Start / Programs / Software / Rockwell Software / RSLinx". In RSLinx click on the menu bar "Communications / Configure Drivers". The window "Configure Drivers" opens (see screenshot below).



Click on the dropdown menu "Available Driver Types" and select the driver 1770-KFD. Click on the button "Add New" to add the driver.

The added driver can now be configured. Select the driver and click on the button "Configure". The window "Driver Configuration" opens (see screenshot below).



Set the baud rate of the DeviceNet network in the menu "Data Rate" and click on the "OK" button. Close the window "Configure Drivers" by clicking on the "Close" button. Then the configured driver can be used.



For a successful communication of the driver with the DeviceNet network, the driver and the network must use the same baud rate. Use the baud rate of the DeviceNet network for the driver.

11.3 Put the network online

Click on "Network / Online" in the menu bar to put networks online in RsNetworkx. The window "Browse for network" opens and connected networks are browsed. Then select the configured driver (e.g. 1770-KFD) to put the network online.

When the network has been put online, RsNetworkx browses available nodes in the network (see screenshot below).

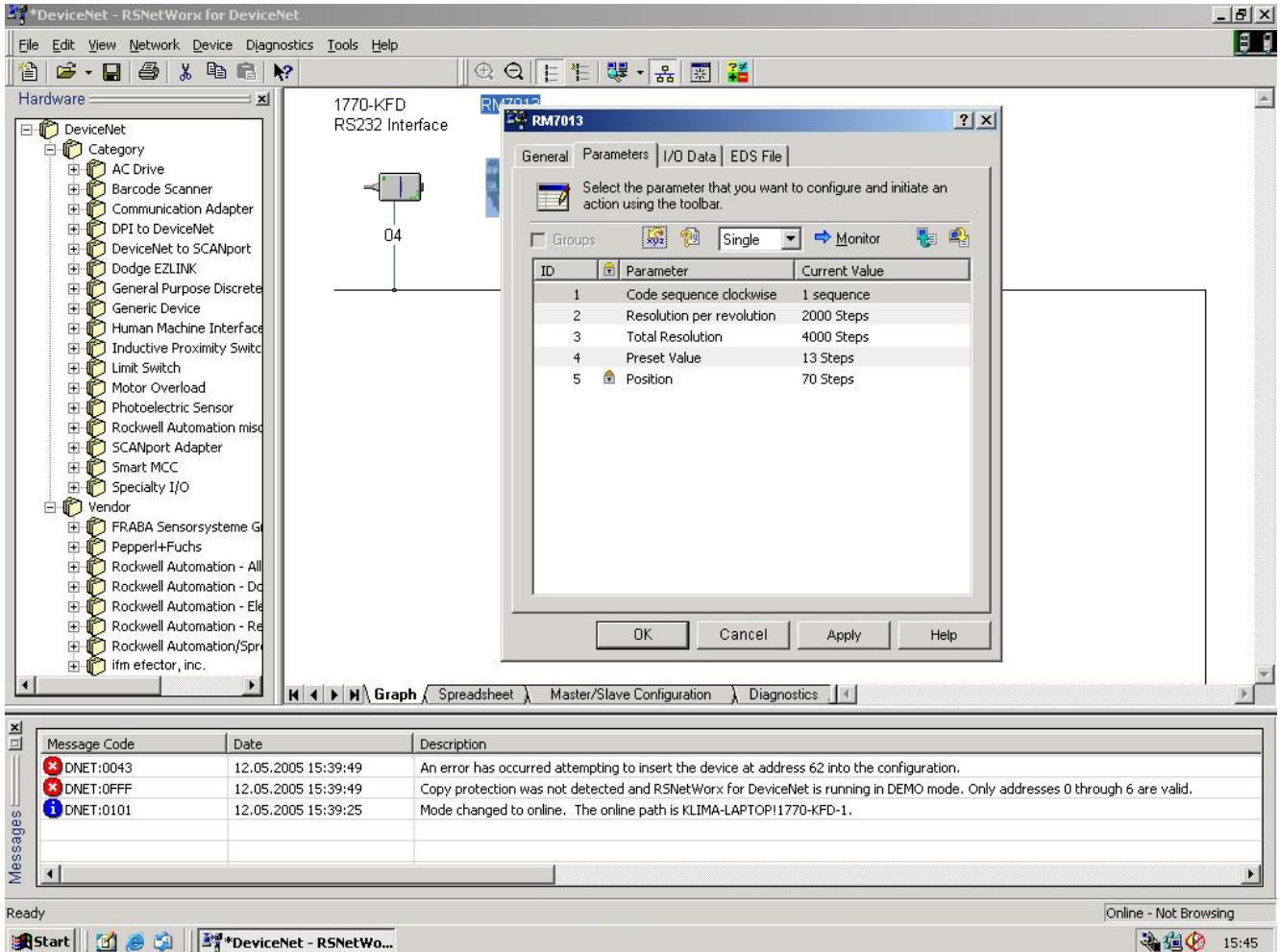
The screenshot displays the 'DeviceNet - RSNetWo...' software interface. The main window shows a network diagram with two nodes: '1770-KFD RS232 Interface' (ID 14) and 'RM7013' (ID 03). A 'Browsing network...' dialog box is open in the center, displaying 'Not found: Device at address 19' and a progress bar. The left sidebar shows a tree view of hardware categories and vendors. The bottom status bar shows 'Ready' and 'Browsing - 19'. A message log at the bottom shows a message: 'DNET:0101 12.05.2005 15:39:25 Mode changed to online. The online path is KLIMA-LAPTOP!1770-KFD-1.'

Message Code	Date	Description
DNET:0101	12.05.2005 15:39:25	Mode changed to online. The online path is KLIMA-LAPTOP!1770-KFD-1.

UK


11.4 Configure the encoder

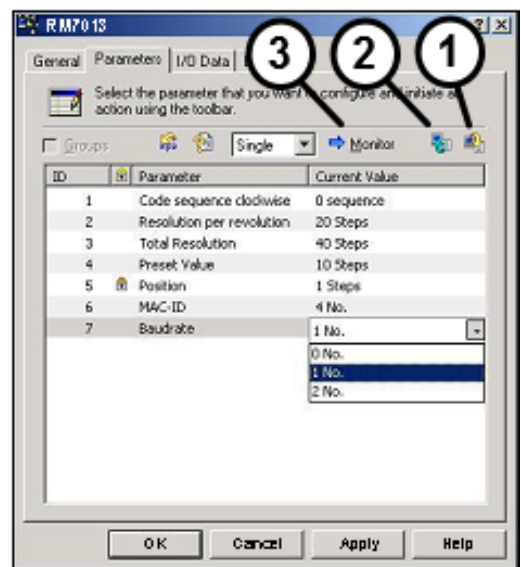
To configure encoders in RsNetworx click on "Device / Properties" in the menu bar. The configuration window of the encoder opens (see screenshot below). The parameters of the encoder can be configured via the "Parameters" tab.



After configuration, the parameters can be uploaded (see screenshot to the right):

- ① Download parameters
- ② Upload parameters
- ③ Show current position value

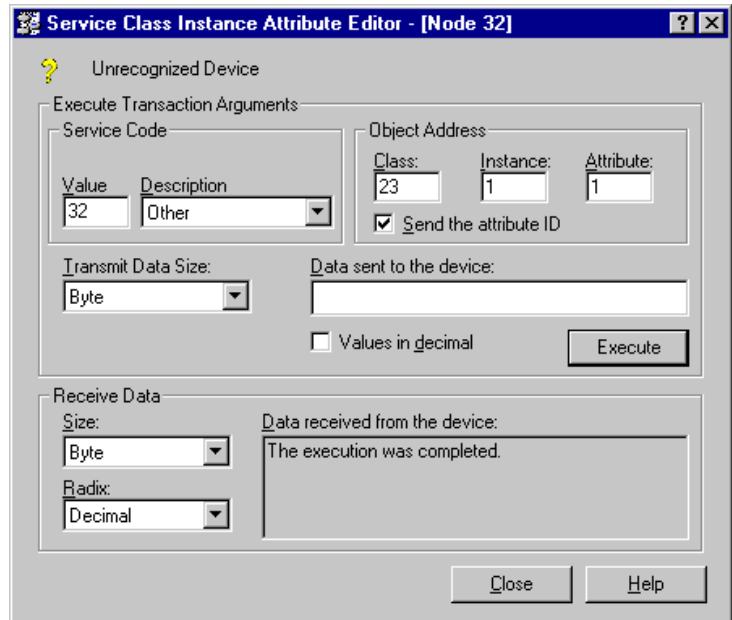
 The MAC ID and the baud rate can only be changed for encoders without terminal cap.



After uploading, the parameters have to be stored in the EEPROM in order to be used. To store the parameters in RsNetwork click on "Device / Class Instance Editor" in the menu bar. The window "Service Class Instance Attribute Editor" opens (see screenshot below).

Adopt the values indicated in the screenshot. Then click on the "Execute" button.

The values are now stored in the EEPROM and can be used.



UK

12 Terms and abbreviations

0b ...	binary value (for bit coding), e.g. 0b0001 0000
0d ...	decimal value, e.g. 0d100
0x ...	hexadecimal value, e.g. 0x64 (= 100 decimal)
Baud rate	transmission speed (1 baud = 1 bit/s)
CAL	CAN Application Layer CAN-based network protocol on application level
CAN	Controller Area Network (bus system for the use in mobile vehicles)
CAN_H	CAN-High; CAN connection/cable with high voltage level
CAN_L	CAN-Low; CAN connection/cable with low voltage level
CANopen	CAN-based network protocol on application level with an open configuration interface (object directory).
CiA	"CAN in Automation e.V." (user and manufacturer organisation in Germany/Erlangen) definition and control body for CAN and CAN-based network protocols
CiA DS	draft standard (published CiA specification which is usually not modified or supplemented for one year)
CiA DSP	draft standard proposal (published CiA specification draft)
CiA WD	work draft (work draft accepted for discussion within CiA)
CiA DS 301	specification concerning the CANopen communication profile; describes the basic communication between network participants such as the transfer of process data in real time, the exchange of data between units or the configuration stage; depending on the application this is completed by the following CiA specifications:
CiA DS 401	device profile for digital and analogue I/O modules
CiA DS 402	device profile for drives
CiA DS 403	device profile for HMI
CiA DS 404	device profile for measurement and control technology
CiA DS 405	specification for interfaces to programmable systems (IEC 61131-3)
CiA DS 406	device profile for encoders
CiA DS 407	application profile for local public transport
COB	CANopen communication object (PDO, SDO, EMCY, ...)
COB ID	CANopen identifier of a communication object
Communication cycle	the synchronisation time to be monitored; max. time between 2 synch objects
EMCY object	emergency object (alarm message; device indicates an error)
Error reg	error register (entry with an error code)
Guarding error	node or network participant could or can no longer be found Guard MASTER: one or several SLAVES no longer reply Guard SLAVE: no polling of the device (SLAVE)

Guard time	within this time the network participant expects a "node guarding" of the network master
Heartbeat	configurable cyclic monitoring among network participants. In contrast to "node guarding" no superior NMT master is required.
ID (also identifier)	identifier; identifies a CAN message. The numerical value of the ID also contains a priority for the access to the bus system.
	ID 0 = top priority.
Idx	index; together with the S index it forms the address of an entry in the object directory
Life time factor	number of attempts in case of a missing guarding reply
Monitoring	is used to describe the error class (guarding monitoring, synch etc.)
NMT	network management
NMT master/ slaves	the NMT master controls the operating states of the NMT slaves
Node Guarding	adjustable cyclic monitoring of slave network participants by a higher master node as well as the monitoring of this polling process by the slave participants
Node ID	node identifier (identification of a participant in the CANopen network)
Object (also OBJ)	term for data/messages which can be exchanged in the CANopen network
Object directory	contains all CANopen communication parameters of a device as well as device-specific parameters and data. Access to the individual entries is possible via the index and S-index.
Operational	operating status of a CANopen participant. In this mode SDOs, NMT commands and PDOs can be transferred.
PDO	Process Data Object; in the CANopen network for transfer of process data in real time such as the speed of a motor. PDOs have a higher priority than SDOs; in contrast to the SDOs they are transferred without confirmation. PDOs consist of a CAN message with identifier and up to 8 bytes of user data.
PDO mapping	describes the application data transferred with a PDO
Pre-Op	pre-operational; operating status of a CANopen participant. After application of the supply voltage each participant automatically goes into this state. In the CANopen network only SDOs and NMT commands can be transferred in this mode but no process data
Prepared	(also stopped) operating state of a CANopen participant In this mode only NMT commands are transferred.
Rec PDO (also Rx PDO)	receive process data object
ro	read only (unidirectional; reading only)
rw	read-write (bidirectional)
Rx queue	input buffer
s16	data type signed 16 bits (incl. sign, 16-bit format)
SDO	Service Data Object. With this object direct access to the object directory of a network participant is possible (read/write). An SDO can consist of several CAN messages. The transfer of the individual messages is confirmed by the addressed participant. With the SDOs, devices can be configured and parameters can be set.

Server SDO	process and parameter set to make the object directory of a network participant available to other participants (clients)
S-Idx (also SIdx)	sub index within the object directory of a CANopen-capable device
Start Guarding	start node monitoring
str	data type string (variable for strings such as text "load")
Sync error	missing sync object OBJ in the adjustable synchronisation time
Sync OBJ	synchronisation object for simultaneous update in the complete network or for accepting process data of the respective parameterised PDOs
Sync windows	time during which the synchronous PDOs have to be transferred.
Time stamp	time stamp to align existing clocks in network participants
Trans Type	type of process data transmission; synchronous/asynchronous
Trans PDO (also Tx PDO)	transmit process data object
Trans SDO (also Tx SDO)	transmit service data object
Tx queue (Transmit)	transmission buffer
u8 (16, 32)	data type unsigned 8 (16, 32) bits (without sign, 8 (16, 32) bit format)
wo	write only