User Manual UMAX140X00/UMAX14032X
Firmware: AX140[1/2/4]00, AX140[1/2/4]00-X00: V2.0.0 and above, AX14032[0/1/2] V1.0 all versions
Document Revision: R
Date: January 23, 2024

USER MANUAL

Protocol Converter

with J1939, CANopen® & Modbus RTU / J1587

P/Ns: AX140100, AX140100-100, AX140200, AX140400, AX140320, AX140321, AX140322

Axiomatic Technologies Oy
Höytämöntie 6
33880 LEMPÄÄLÄ, Finland
Tel. +358 103 375 750
salesfinland@axiomatic.com
www.axiomatic.fi

Axiomatic Technologies Corporation
1445 Courtneypark Dr. E.
Mississauga, ON Canada L5T 2E3
Tel. 1 905 602 9270
sales@axiomatic.com
www.axiomatic.com

ACRONYMS

CAN	Controller Area Network
CANopen®	CANopen® is a registered community trademark of CAN in Automation e.V.
EA	Axiomatic Electronic Assistant
COB	Communication Object
EDS	Electronic Data Sheet
EMCY	Emergency
LSB	Least Significant Byte (or Bit)
LSS	Layer Settling Service
MB	Modbus
MID	Message Identification
MSB	Most Significant Byte (or Bit)
NMT	Network Management
PID	Parameter Identification Character
RO	Read Only Object
RPDO	Received Process Data Object
RW	Read/Write Object
SDO	Service Data Object
SID	Sub-system Identification Number
TPDO	Transmitted Process Data Object
WO	Write Only Object

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

This user manual outlines the general configuration and operation of AX140100, AX140100-100, AX140200, AX140400, AX140320, AX140321 and AX140322 Protocol Converter devices. The Protocol converter is capable or routing messages between all of its three communication interfaces found at the main connector. These interfaces include two CAN ports (both SAE J1939 and CiA CANopen® are supported) and one RS485 port (Modbus RTU or SAE J1587, depending on the software loaded). The AX140320, AX140321 and AX140322 have an RS422 port which can be used as RS-485 with some wiring changes.

PROTOCOL CONVERTER USER INTERFACE

The protocol converter can be configured using the Axiomatic Electronic Assistant (EA) tool from Axiomatic Technologies. In general, the configuration should be done using the most recent version of Axiomatic Electronic Assistant (can be downloaded from www.axiomatic.com). The CANopen® interface supports CANopen® configuration tools generally available on the market.

The main network configuration options for different interfaces include setting CAN and RS485 port communication parameters (baud rate, etc.). From communication protocol point of view, J1939/CANopen® receive and transmit configurations, Modbus slave parameters and a number of Modbus client messages can be defined.

The message routing can be configured rather freely. Data routing between all three interfaces is supported. Protocol Converter can be also configured to actively request data from remote Modbus RTU/J1587 node (acts as a master, not just a slave device).

Further, the AX140100, AX140400, AX140320 and AX140322 support also direct, low level routing of J1939 messages between the two CAN channels. This provides a fast filtering/gateway option for easily connecting two J1939 buses, even having different baud rates.

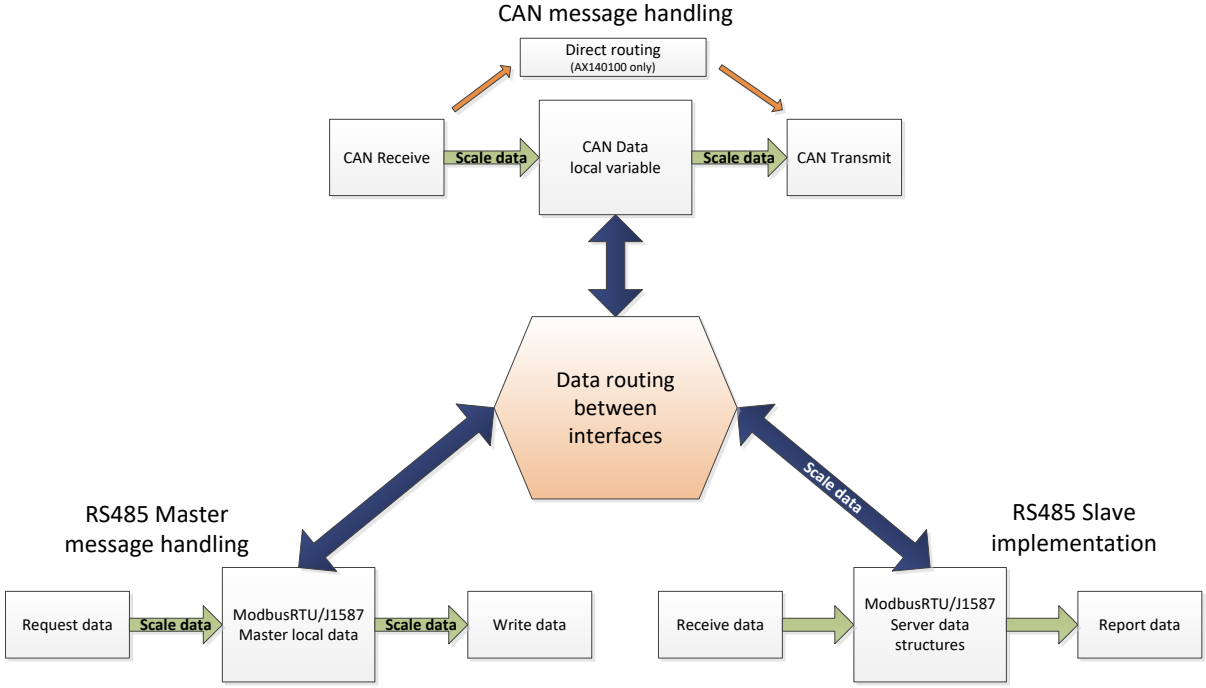


Figure 1: Protocol Converter data routing and scaling

CONFIGURATION USING AXIOMATIC ELECTRONIC ASSISTANT

The Protocol Converter devices support EA configuration via CAN #1 J1939 interface. Further, the bootloader supplied with the device supports application updating via CAN #1 interface.

It should be noted that EA configuration is targeted mostly for configuring the J1939 and RS485 related parameters of the Protocol Converter. EA configuration on AX140200 & AX140321 support the configuration of the default CANopen parameters, such as the PDO parameters (see 1.18 & 1.19) and mappings (see 1.20 & 1.21), but any other, more advanced CANopen configuration needs to be using CANopen tools.

In general, the most recent version of Axiomatic Electronic Assistant should be used when configuring the Protocol Converter (can be downloaded from www.axiomatic.com).

The following sections describe the different configuration options in more detail.

1.1. *J1939 Network Parameters*

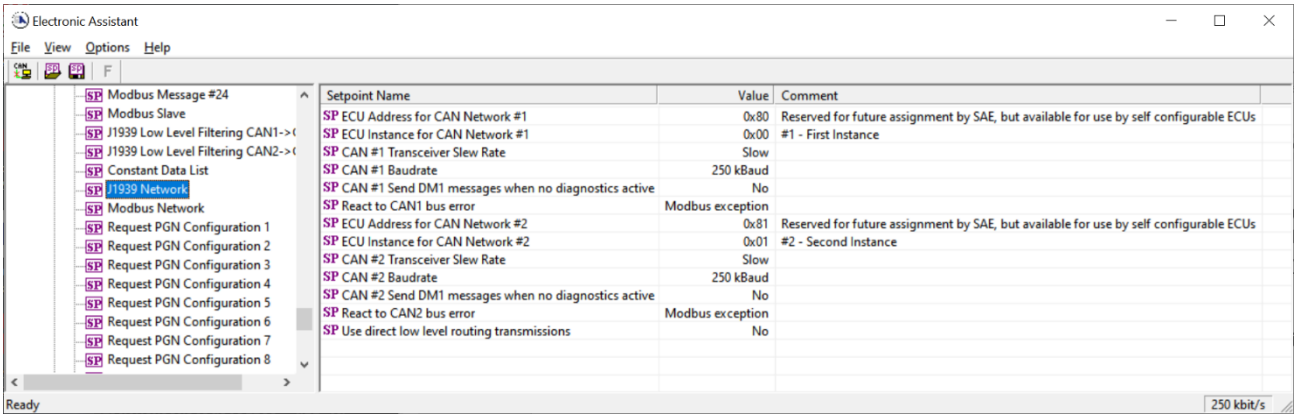


Figure 2: J1939 network parameters

The J1939 Network Parameters consist of *ECU Instance*, *ECU Address* and *CAN Transceiver Slew Rate* settings for both interfaces. Also, a custom baud rate for J1939 interface can be set (available selections include 250k, 500k and 1000k). There is also an option for disabling the transmission of empty diagnostics messages (DM1) in case the diagnostics messages with no errors should not be transmitted to the J1939 bus.

In addition to a fixed baud rate setting, the AX140320 and AX140322 support auto baud rate feature.

When the *React to CANx bus error* is set to 'Modbus exception', the Modbus (slave) interface responds with exception when data is read from it by other Modbus nodes. The other options are 'J1939 DM1' and 'Both Modbus exception and J1939 DM1'.

The *Use direct low level routing transmissions* defines the algorithm to use when handling the CAN frames on the lowest level of direct CAN to CAN message routing. If the *Use direct low level routing transmissions* is set to 'Yes', the received frames are not buffered, but instead are sent out directly from the other CAN interface.

Note, if ECU Instance/Address parameters are changed, the Protocol Converter will restart its communication functionality. New baud rate will be taken into use at next boot up.

1.1.1. Note on J1939 network baud rate setting on AX140100, AX140200 and AX140400

The Protocol Converter device supports configurable baud rate on its J1939 network(s). The new baud rate will be put into use on the next boot-up. When changing the settings, please keep in mind the following:

J1939 baud rate can be changed as many times as needed, but only if the bootloader is not started. Setting the "*Force Bootloader To Load on Reset*" –flag programs the current interface baud rates into the bootloader data block. **This will prevent changing the J1939 baud rate further, and the configuration can be only reset by reprogramming the Protocol Converter application using EA and CAN-USB adaptor.**

Hint: All J1939 interfaces of the protocol converter support EA configuration tool. In case the baud rate is programmed by mistake to a wrong value, the other J1939 interface (or CANopen interface in case of CANopen capable devices, object 5600h) can be used to reprogram the J1939 baud rate.

1.2. *CANopen Network Parameters (AX140200/AX140321 only)*

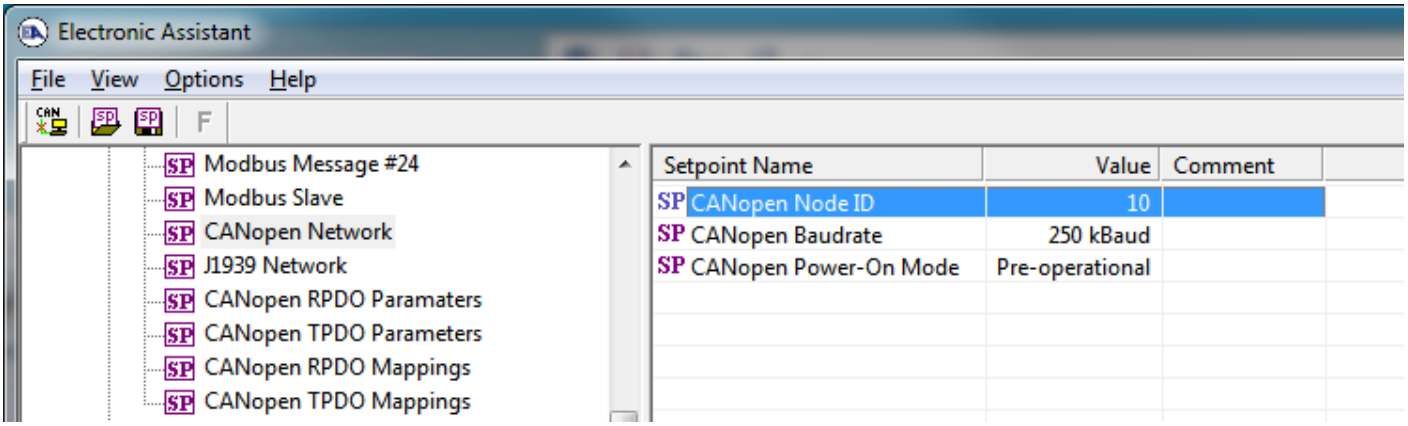


Figure 3: CANopen® network parameters

The Node ID, Baud rate and the startup mode of the CANopen® interface can be configured using the EA configuration interface.

Note, that the device needs to be restarted (power cycled) for taking the new CANopen parameters into use.

1.3. *Modbus RTU Network Parameters (AX140100, AX140200 and AX140320, AX140321 only)*

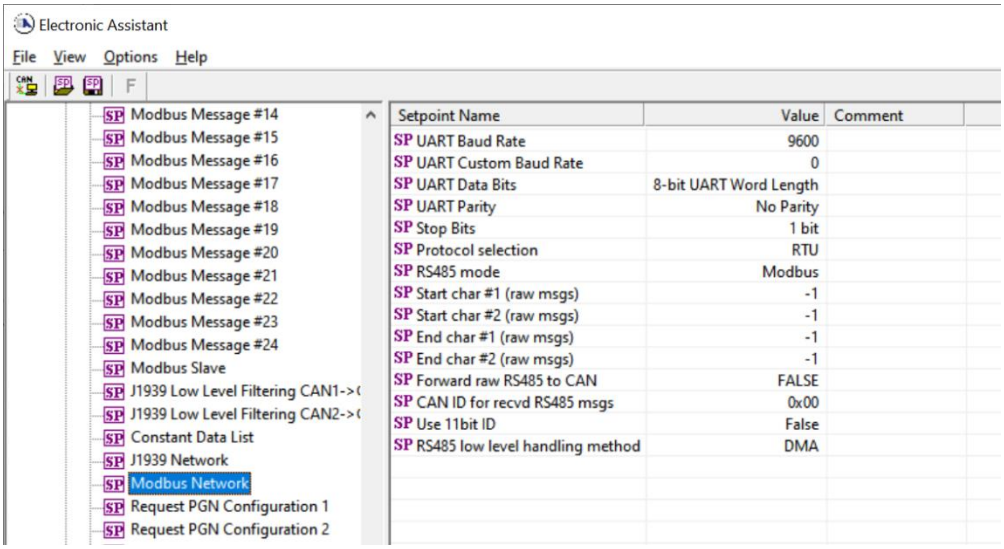


Figure 4: Modbus RTU network parameters

Modbus RTU parameters include settings for the RS485 port, such as baud rate, data bits, parity and stop bits. The list of available baud rates contains predefined values starting from 1200 bps up to 115200 bps and also an entry for defining a custom baud rate.

Note, that the device needs to be restarted (power cycled) for taking the new Modbus RTU parameters into use.

In case Odd or Even parity is selected to be used with 8-bit UART data length, the *UART Data Bits* setpoint needs to be set to 9 bits to accommodate the 8 data bits and the parity bit.

1.4. *J1587 Network Parameters (AX140400/AX140322 only)*

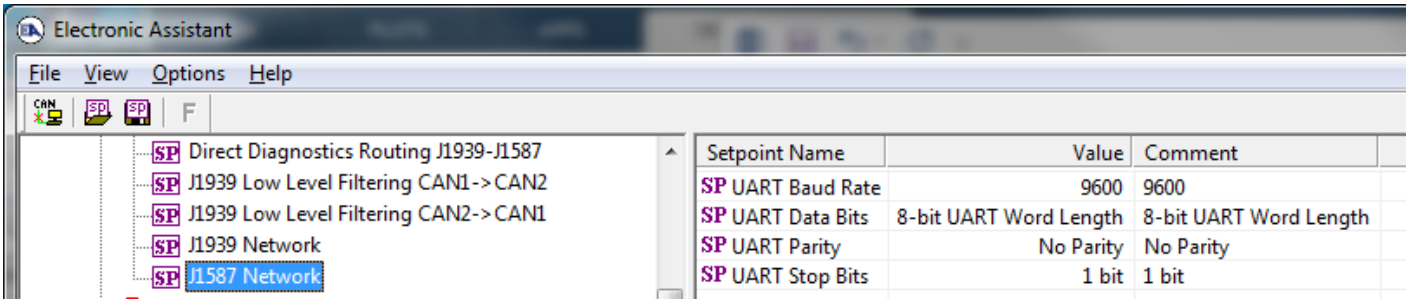


Figure 5: J1587 network parameters

J1587 network parameters include settings for the RS485 port, such as baud rate, data bits, parity and stop bits. The list of available baud rates contains values starting from 1200 bps up to 115200 bps.

Note, that the device needs to be restarted (power cycled) for taking the new J1587 parameters into use.

1.5. *CAN Output Message Specification*

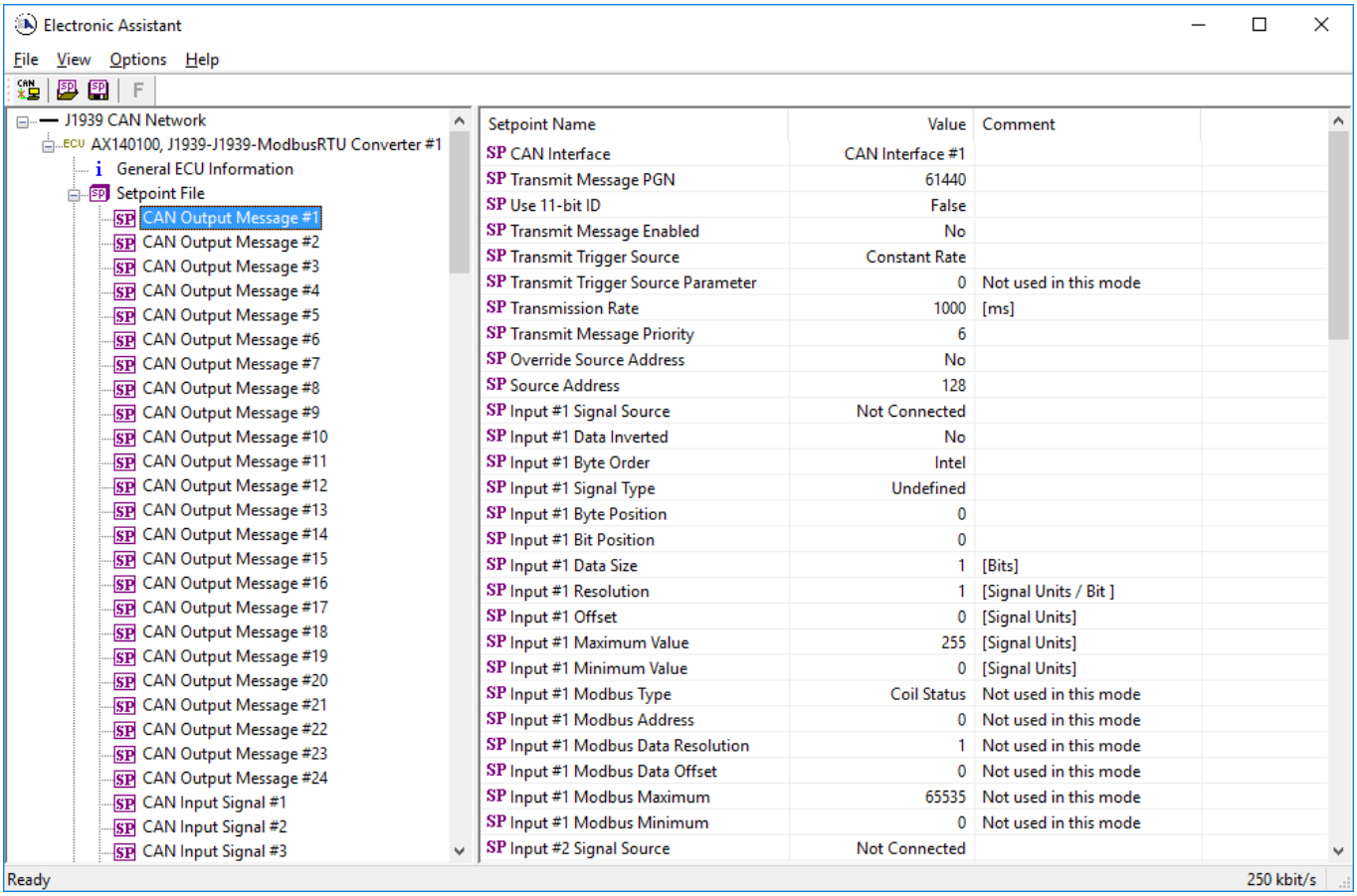


Figure 6: CAN output message definition

AX140100&AX140320 support up to 24 CAN output messages for J1939 interfaces in total. AX140200&AX140321 support 12 CAN output messages and AX140400&AX140322 support 16 CAN output messages for J1939 interfaces in total. Each message can hold up to 5 signals.

The special CAN to Modbus slave version, AX140100-100 supports only 4 output messages. This is because the main purpose of that Protocol Converter version is to listen to CAN messages and make the received data available in Modbus.

1.5.1. Output message specific settings

Parameter name	***Value range***	***Comment***
CAN Interface	1, 2	The CAN interface to use for transmitting this message
Transmit Message PGN	0x0000 – 0x3FFFF	PGN for the output message
Use 11-bit ID	No, Yes	Use 11-bit ID instead of the standard 29 bit ID (only for custom messaging!)
Transmit Message Enabled	No, Yes	Disable / enable the particular message
Transmit Trigger Source	Constant rate, CAN Rx Msg reception, Modbus message reception	Transmit triggering method to use. Constant rate is defined by the “Transmission Rate” parameter.

Transmit Trigger Source Parameter	<uint8>	In the case of CAN Rx Msg triggering, this field specifies the CAN Rx Msg definition number (1-56) for triggering the transmission. In case of transmit triggering is done on a Modbus message reception event, this field is used for specifying the received Modbus function code for triggering the transmission.
Transmission Rate	0-60000	Periodic transmission rate in milliseconds
Transmit Message Priority	0-7	Priority bits for the J1939 message
Override Source Address	No, Yes	Should the source address be set to a specific value (by default the protocol converter's claimed address shall be used).
Source Address	0x00 – 0xFF	The new source address to use, if above is set to '1'.

Note, if transmission rate is set to 0ms, the message is sent only on request or on a configured trigger event.

1.5.2. Output signal specific settings

Parameter name	***Value range***	***Comment***
Signal Source	Control not used … Modbus data … CAN Input data … CANopen Input data** … Constant data	The data source for this particular signal. Modbus data specifies that the data is routed from the Modbus interface, CAN data specifies the CAN input message data to use.
Data Inverted	No, Yes	Whether to invert data.
Byte Order	Intel, Motorola	Intel – LSB first/little endian data, Motorola – MSB first/big endian data
CAN Signal Type	0 – 2	CAN data type: 0 – not used, 1 – discrete data, 2 – continuous data
Byte Position	0 – 7	The byte position in the message data field to use for signal data.
Bit Position	0 – 7	The bit position within the above byte position to use for signal data.
Data Size	1 – 32	Width of data in bits.
Resolution*	<float>	Data resolution to use.
Offset*	<float>	Data offset to use.
Minimum Value*	<float>	Minimum value for data.
Maximum Value*	<float>	Maximum value for data.
Modbus Type	Coil status, Input status, Holding reg., Input reg.	If data source is "Modbus Local Data", this specifies the data source.
Modbus Address	<uint16>	If data source is "Modbus Local Data", this specifies the Modbus address (local structures).

		If data source is “Modbus Remote Data”, this specifies the Modbus Master #” message number (1-24) to use for requesting data
Modbus Data Resolution	<float>	Data resolution for Modbus Data, used when scaling Modbus data -> CAN data.
Modbus Data Offset	<float>	Data offset for Modbus Data, used when scaling Modbus data -> CAN data.
Modbus Minimum	<float>	Minimum value for Modbus Data, used when scaling Modbus data -> CAN data.
Modbus Maximum	<float>	Maximum value for Modbus Data, used when scaling Modbus data -> CAN data.
CANopen Message Number**	1-16	CANopen message (RPDO) number for data
CANopen Message Subindex**	1-4	CANopen message (RPDO) subindex number for data

*For more detailed explanation for different data scaling variables, please see section 1.3.
**Only available on devices having CANopen® interfaces, such as AX140200&AX140321.

1.6. *CAN Input Message Specification*

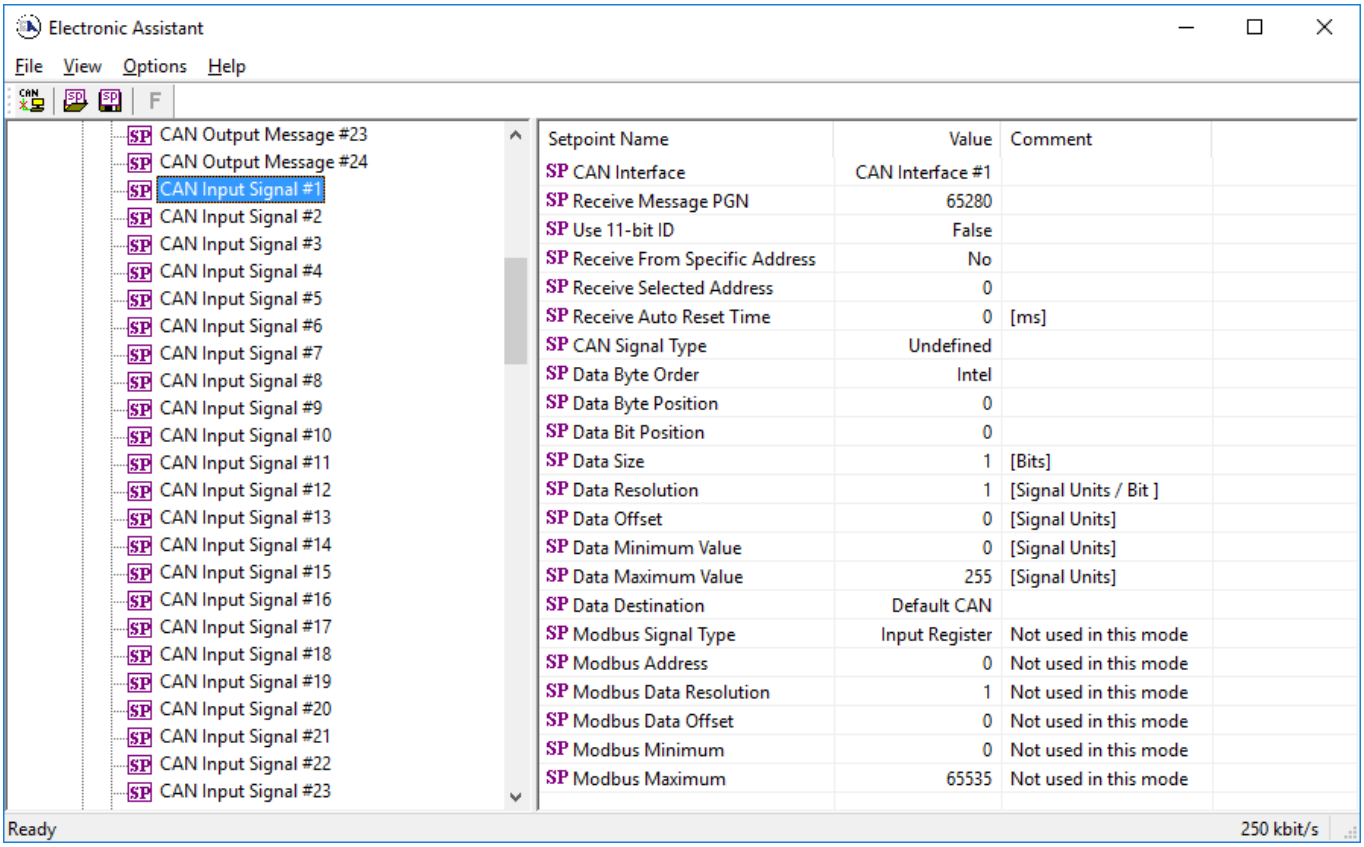


Figure 7: CAN input message definition

AX140100&AX140320 supports up to 56 CAN input messages for J1939 interfaces in total. AX140200&AX140321 and AX140400&AX140322 support 32 CAN input messages in total. Each message can hold one input signal.

The special CAN to Modbus version AX140100-100 supports 120 CAN input messages.

Parameter name	***Value range***	***Comment***
CAN Interface	1, 2	CAN Interface to listen for this message.
Receive Message PGN	0x0000 – 0x3FFFF	The PGN to listen to.
Use 11-bit ID	No, Yes	Use 11 bit ID instead of the standard 29 bit ID (only for custom messaging!)
Receive From Specific Address	No, Yes	Whether to match the message source address with a specific address value.
Receive Selected Address	0x00 – 0xFF	The source address to match for this PGN, valid only if “Receive From Specific Address” is set to ‘1’.
Receive Auto Reset Time	0 – 60000	Received data reset time in milliseconds. If set to a non-zero value, the received data is zeroed after the specified time has elapsed.
CAN Signal Type	0 – 2	CAN data type: 0 – not used, 1 – discrete data, 2 – continuous data
Data Byte Order	Intel, Motorola	Intel – LSB first/little endian data, Motorola – MSB first/big endian data

Data Byte Position	0 – 7	The byte position in the message data field to use for signal data.
Data Bit Position	0 – 7	The bit position within the above byte position to use for signal data.
Data Size	1 – 32	Width of data in bits.
Data Resolution*	<float>	Data resolution to use.
Data Offset*	<float>	Data offset to use.
Data Minimum Value*	<float>	Minimum value for data.
Data Maximum Value*	<float>	Maximum value for data.
Data Destination	Default CAN, Modbus Local Destination, CANopen Master Data**, DIRECT RS485 FORWARDING	Selects whether to store the data into the built-in CAN Rx data storage or into the local Modbus slave database. The AX140200 and AX140200 also support direct CAN to RS485 data forwarding.
Modbus Signal Type	Coil status, Input status, Holding reg., Input reg.	If data source is “Modbus Local Data”, this specifies the data source.
Modbus Address	<uint16>	If data destination is “Modbus Local Data”, this specifies the Modbus address (local structures). If data destination is “Modbus Remote Data”, this specifies the Modbus Master #” message number (1-24) to use for writing the received data.
Modbus Minimum	<float>	Minimum value for Modbus Data, used when scaling Modbus data -> CAN data
Modbus Maximum	<float>	Maximum value for Modbus Data, used when scaling Modbus data -> CAN data
Modbus Data Resolution	<float>	Data resolution for Modbus Data, used when scaling Modbus data -> CAN data.
Modbus Data Offset	<float>	Data offset for Modbus Data, used when scaling Modbus data -> CAN data.
CANopen Master Message Number**	<uint8>	CANopen Master Message TX data object subindex, used when Data Detination is CANopen Master Data.

* For more detailed explanation for different data scaling variables, please see section 1.3.
** Only available on devices having CANopen® interfaces, such as AX140200&AX140321.

It must be noted that in case multiple CAN Input Signals are listening to the same PGN, only the last one with the same PGN will send a trigger signal to the other Protocol Converter function blocks. For example, if CAN Input Signals #1, #2 and #4 are configured to receive the same PGN, only CAN Input Signal #4 can be used as a trigger source in other Protocol Converter functions.

1.7. *Modbus Slave Definition*

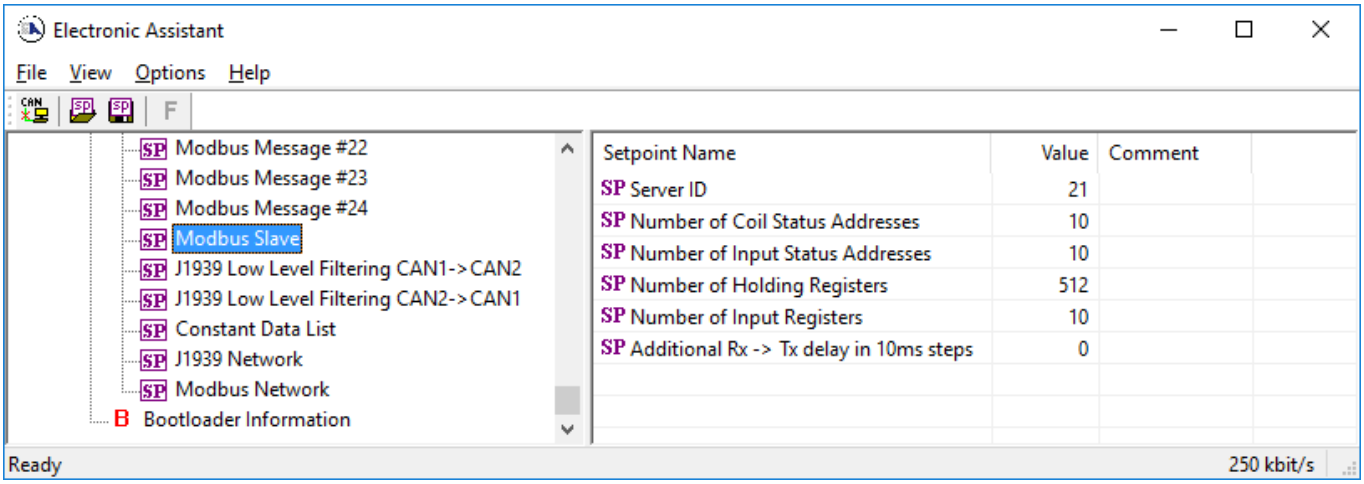


Figure 8: Modbus slave node definition

These parameters define the number of Modbus slave registers to support. The maximum number of each type is 1024. The starting address is 1 for each type. This makes the maximum address range for Holding registers (40000-41024) and Input registers (30000-31024). Note, the implementation takes automatically into account the type when determining the actual addresses.

Parameter name	***Value range***	***Comment***
Server Id	<uint8>	Modbus Server Id
Number of Coil Status Addresses	<uint16>	Number of coil status bit addresses to implement
Number of Input Status Addresses	<uint16>	Number of input status bit addresses to implement
Number of Holding Registers	<uint16>	Number of holding register addresses to implement
Number of Input Registers	<uint16>	Number of input register addresses to implement
Additional Rx->Tx delay in 10ms steps	0 – 20	The delay between incoming read / write command and the response message sent by the Protocol Converter.

Note, that the device needs to be restarted (power cycled) for taking the new Modbus RTU parameters into use.

1.7.1. Special Modbus Holding registers in AX140100 & AX140320

Starting from firmware version 3.14 (AX140100, Nov 2016), the Modbus slave interface has built in custom holding registers at selected addresses. These custom registers can be used for starting the bootloader (with or without default settings) and for carrying out a SW reset.

The custom registers are described in the Table 1 below, together with example Modbus messages that are needed for triggering the functionality. Please note, that the example messages assume that the Protocol Converter uses Modbus address of 21 (0x15).

Table 1: Special Modbus holding registers and data

Command	**Holding register**	**Data to write**
SW Reset	65535 (0xFFFF)	29299 (0x7273)
Start Bootloader	65534 (0xFFFE)	25196 (0x626C)
Start Bootloader and erase Bootloader data	65533 (0xFFFD)	25954 (0x6562)

Example Modbus commands for triggering the special functionality, all numbers are in hexadecimal format.

SW reset

```
15 06 FF FF 72 73 EF BF
```

Start Bootloader

```
15 06 FF FE 62 6C F2 77
```

Start Bootloader and erase Bootloader data

```
15 06 FF FD 65 62 81 83
```

1.8. *Modbus Master (Client Message) Definition*

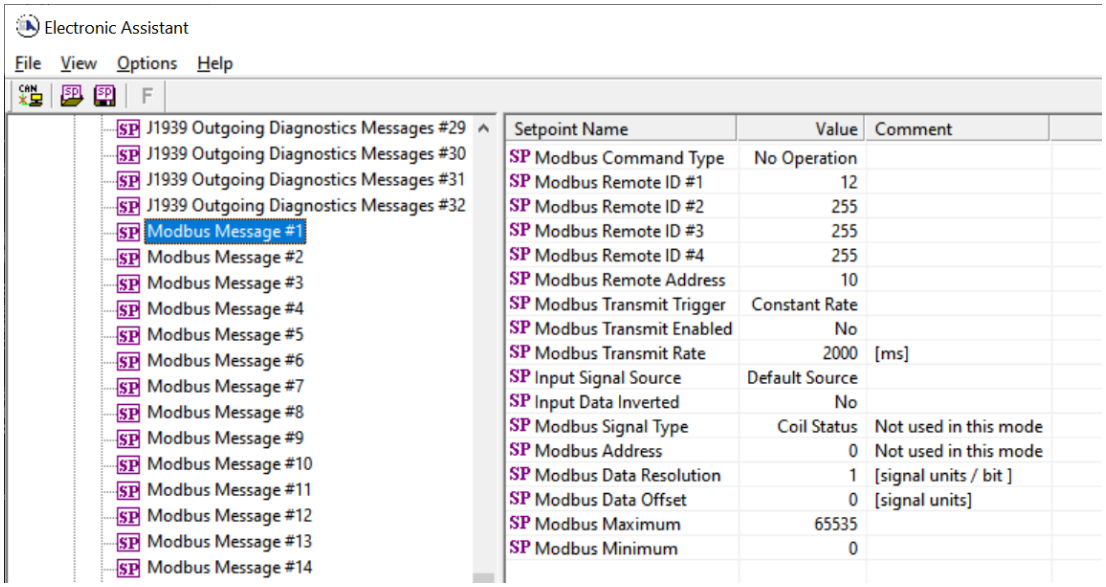


Figure 9: Modbus master (client) message definition

The Modbus versions of the Protocol Converter (AX140100, AX140200 and AX140320, AX140321) support 24 Modbus (master/client) messages for actively reading/writing data into other Modbus nodes.

The special CAN to Modbus slave version, AX140100-100 supports only 4 Modbus messages. This is because the main purpose of that Protocol Converter version is to listen to CAN messages and make the received data available in Modbus slave interface.

Parameter name	***Value range***	***Comment***
Modbus Command Type	No operation … Write remote holding register	Selects Modbus client command to execute.
Modbus Remote ID #1	<uint8>	The remote Modbus Device ID to access
Modbus Remote ID #2…#4	<uint8>	Additional remote Modbus Device IDs for sending multiple write commands using a single Message definition.
Modbus Remote Address	<uint16>	The remote Modbus device address to request/write data.
Modbus Transmit Trigger	No configuration, Transmit at constant rate, on reception of a specific CAN message	Specifies in what circumstances this particular message is sent. Note, that “No configuration” still accepts separate transmit triggering by selecting “Remote Modbus” as CAN Input messages’ Output Destination.
Modbus Transmit Enabled	No, Yes	Disable / enable this particular message.
Modbus Transmit Rate	0 – 60000	The transmit interval in milliseconds for this message. Note, this value is valid only if the “Constant rate” is selected as the transmit trigger.

Input Signal Source	No configuration … Modbus data … CAN input data … CANopen input data** … Constant data	The data source for this particular message
Input Data Inverted	No, Yes	Inverted data/no data inversion
Modbus Signal Type	Coil status, Input status, Holding reg., Input reg.	If data source is “Modbus Local Data”, this specifies the data source type.
Modbus Address	<uint16>	If data source is “Modbus Local Data”, this specifies the data source address. If data source is “Modbus Remote Data”, this specifies the Modbus Master #” message to use for requesting data.
Modbus Data Resolution	<float>	Data resolution for Modbus Data, used when scaling Modbus data -> CAN data.
Modbus Data Offset	<float>	Data offset for Modbus Data, used when scaling Modbus data -> CAN data.
Modbus Minimum*	<float>	Minimum value for Modbus Data, used when scaling Modbus data -> CAN data
Modbus Maximum*	<float>	Maximum value for Modbus Data, used when scaling Modbus data -> CAN data
CANopen Message Number**	1-16	CANopen message (RPDO) number for data
CANopen Message Subindex**	1-4	CANopen message (RPDO) subindex number for data

Note, that (all) Modbus messages assume the message length equal to 1. This allows writing/reading one coil/register at a time. Accessing multiple coils/registers would require configuring multiple Modbus messages.

*For more detailed explanation for different data scaling variables, please see section 1.5.
**Only available on devices having CANopen® interfaces, such as AX140200&AX140321.

1.9. *J1939-J1939 Low-Level Routing Definition*

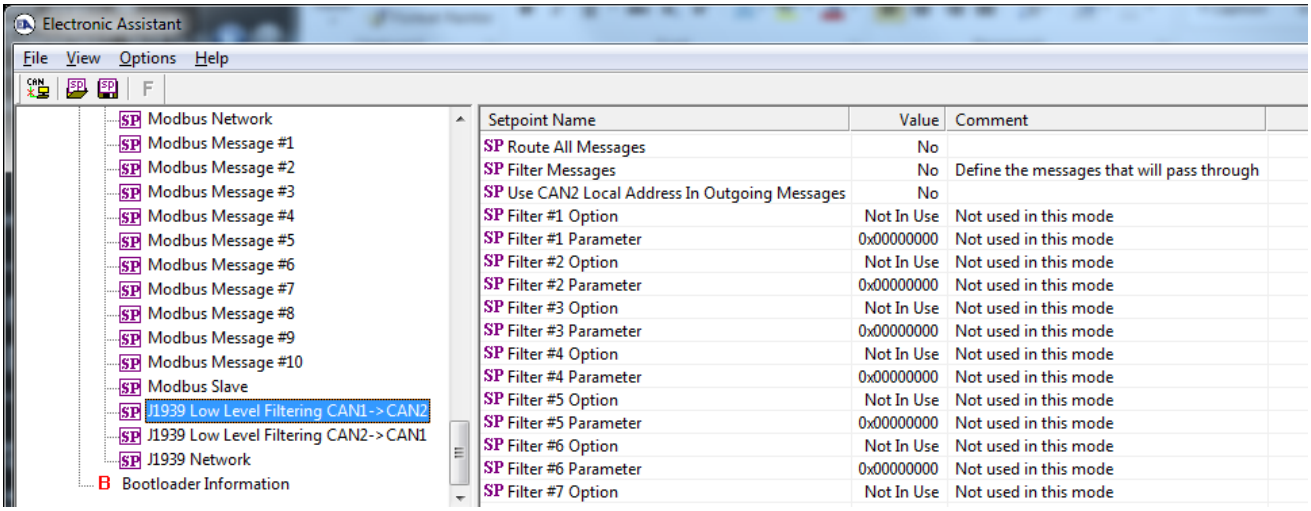


Figure 10: J1939 Low Level Filtering CAN1->CAN2 definition

Parameter name	***Value range***	***Comment***
Route All Messages	No, Yes	Selects if no messages / all messages are routed from CAN1 -> CAN2.
Filter Messages	No, Yes	Whether to filter messages. Setting this to 'Yes' enables the list of filters below. Note, if 'Route all messages' is set to 'No', the filters below will select the messages that will not get routed.
Use CAN2(1) Local Address in Outgoing Messages	No, Yes	Sets the source address field of the outgoing messages to the claimed address of the protocol converter device.
Filter Option #1, #2, #3, … #20	See Table 2	Sets the filtering option.
Filter Parameter #1, #2, #3 … #20	0x00000000 – 0xFFFFFFFF	Defines the PGN, ID, priority or a custom value to match with. Note, the right shifting of this value is taken care by the application.

The low-level routing feature will handle the received CAN frames before the frames are passed over to J1939/CANopen stack(s). This way, all received messages can be routed, independent of the higher-level device configuration.

NOTE: In case a Custom ID filter is used and the ID specified is equal to or below 0x7FF, the ID is assumed to be received in an 11 bit ID CAN frame. For proper initialization of the CAN filters, a power cycle is needed after configuring Custom IDs with values equal to or below 0x7FF (2047 dec).

The low level routing feature is capable of parsing the configured PGNs (Match PGN filter configuration) from TP frames and forward the necessary set of frames to the other CAN interface.

Table 2: Low level filter options

Low level filter option	Description
No configuration	No configuration – no routing
Match PGN	Route all J1939 messages with the specified PGN
Match ID	Route all J1939 messages with the specified SA
Match Priority	Route all J1939 messages with the specified priority
Custom Match	Route all 29bit/11bit ID frames that have an equal CAN Frame ID
Priority OR	If two routing rules are set to 'Priority OR' configuration, only the one with higher priority will be forwarded when frames matching both rules are on the bus.
All ExtID Frames	Route all frames that have a 29bit ID
All StdID Frames	Route all frames that have an 11bit ID

1.10. *J1939 Diagnostics To Monitor, CANx*

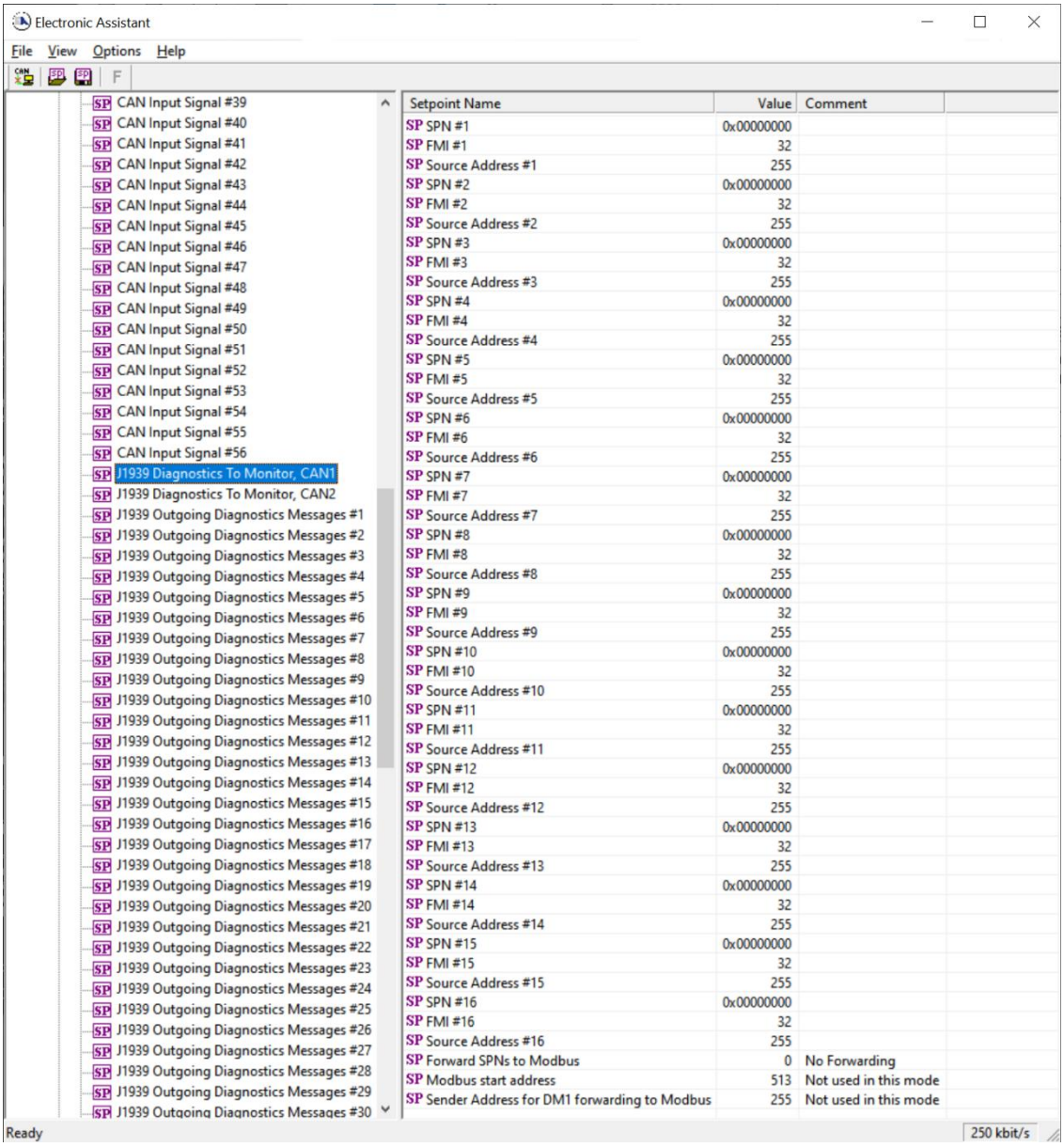


Figure 11: J1939 Diagnostics to monitor, CAN1&CAN2 definition

Parameter name	***Value range***	***Comment***
SPN #x	0 … 0x7FFFF	Defined the SPN to read in from received DM1 messages.
FMI #x	0 … 32***	The FMI for the DM1 with the configured SPN to listed
Source Address #x	0 … 255**	The source address for the DM1 with the configured SPN to listen.
Forward SPNs to Modbus*	0 – No forwarding 1 – Forward SPNs as defined in the list 2 – Forward SPNs based on	Defines the SPN forward into Modbus registers.

	the source address	
Modbus start address*	0…1024	The Modbus slave address starting from which the SPNs are forwarded to.
Sender Address for DM1 forwarding to Modbus*	0 … 255**	In case Source Address based SPN forwarding is in use, this defines the source address from which the SPNs are listened.

* These setpoints are available only in the Modbus versions of the Protocol Converter (AX140100, AX140200 and AX140320, AX140321).

** In case address 255 is specified as Source address, received DM1 messages from all nodes are forwarded.

*** In case FMI 32 is specified as FMI, received DM1 messages with all FMIs are forwarded.

The Modbus registers contain the forwarded SPNs in the following arrangement:

Table 3: J1939 DM1 mapping into Modbus

AX140100, AX140320	
Modbus Address (Holding register)	**Register contents**
Start address + 0	Code 1, SPN LSB (16 bits)
Start address + 1	Code 1, SPN MSB
Start address + 2	Code 1, FMI
Start address + 3	Code 1, SA & OC
Start address + 4	Code 2, SPN LSB (16 bits)
Start address + 5	Code 2, SPN MSB
Start address + 6	Code 2, FMI
Start address + 7	Code 2, SA & OC
Start address + 8	Code 3 …

Table 4: CANopen EMCY mapping into Modbus

AX140200, AX140321	
Modbus Address (Holding register)	**Register contents**
Start address + 0	Code 1, Error Code
Start address + 1	Code 1, Error Register
Start address + 2	Code 1, Manufacturer Code (LSB)
Start address + 3	Code 1, Manufacturer Code (MSB)
Start address + 4	Code 1, Node ID
Start address + 5	Code 2, Error Code
Start address + 6	Code 2, Error Register
Start address + 7	Code 2, Manufacturer Code (LSB)
Start address + 8	Code 2, Manufacturer Code (MSB)
Start address + 9	Code 2, Node ID
Start address + 10	Code 3, …

The SPN list-based forwarding supports the forwarding of 16 SPNs. The Source address-based option supports the forwarding of 32 SPNs.

1.11. *J1939 Outgoing Diagnostics Messages #x*

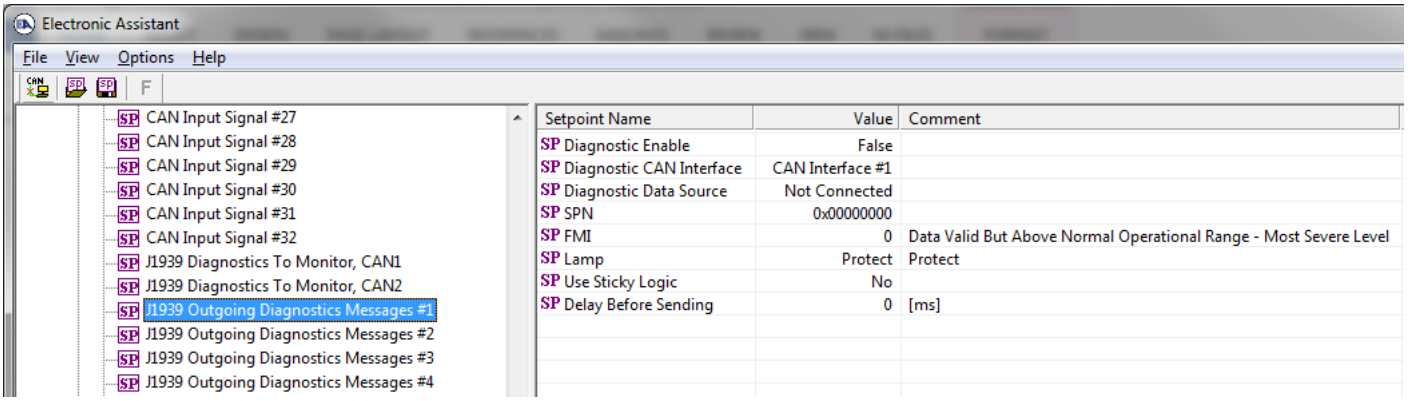


Figure 12: J1939 Outgoing Diagnostics Messages definition

Parameter name	***Value range***	***Comment***
Diagnostic Enable	0…0x7FFFF	Defined the SPN to read in from received DM1 messages.
Diagnostic CAN Interface	CAN Interface #1, CAN Interface #2	This defines the CAN interface to use with the diagnostic message. Please note, this is a **READ-ONLY** parameter. Messages 1…16 are fixed to CAN Interface #1 and messages 17…32 are for CAN Interface #2
Diagnostic Data Source*	Not connected, J1939 Rx Diagnostics 1…32, J1587 Rx Diagnostics 1…16, Modbus Read/Write/Communication error*	The diagnostic data source for this J1939 diagnostic message.
SPN	0…0x7FFFF	The SPN to use in this diagnostic signal. In case this is set to 0, the SPN, FMI and OC values are copied over from Diagnostic Data Source. If this is set to a non-zero value, then the SPN and FMI are set as specified and OC is taken from an internal counter.
FMI	0...31	The FMI to use.
Lamp	0…3	The Lamp to use.
Use Sticky Logic	False, true	If this is set, the Diagnostic Status is not reset automatically. Instead, a DM3 message is needed (or power cycle).
Delay Before Sending	0…60000ms	The delay before the DM1 is sent after the diagnostic condition has become active.

* Depending on the actual device in question, Modbus or J1587 specific diagnostic event sources are available.

1.12. *J1587 Output Messages #x (AX140400/AX140322 only)*

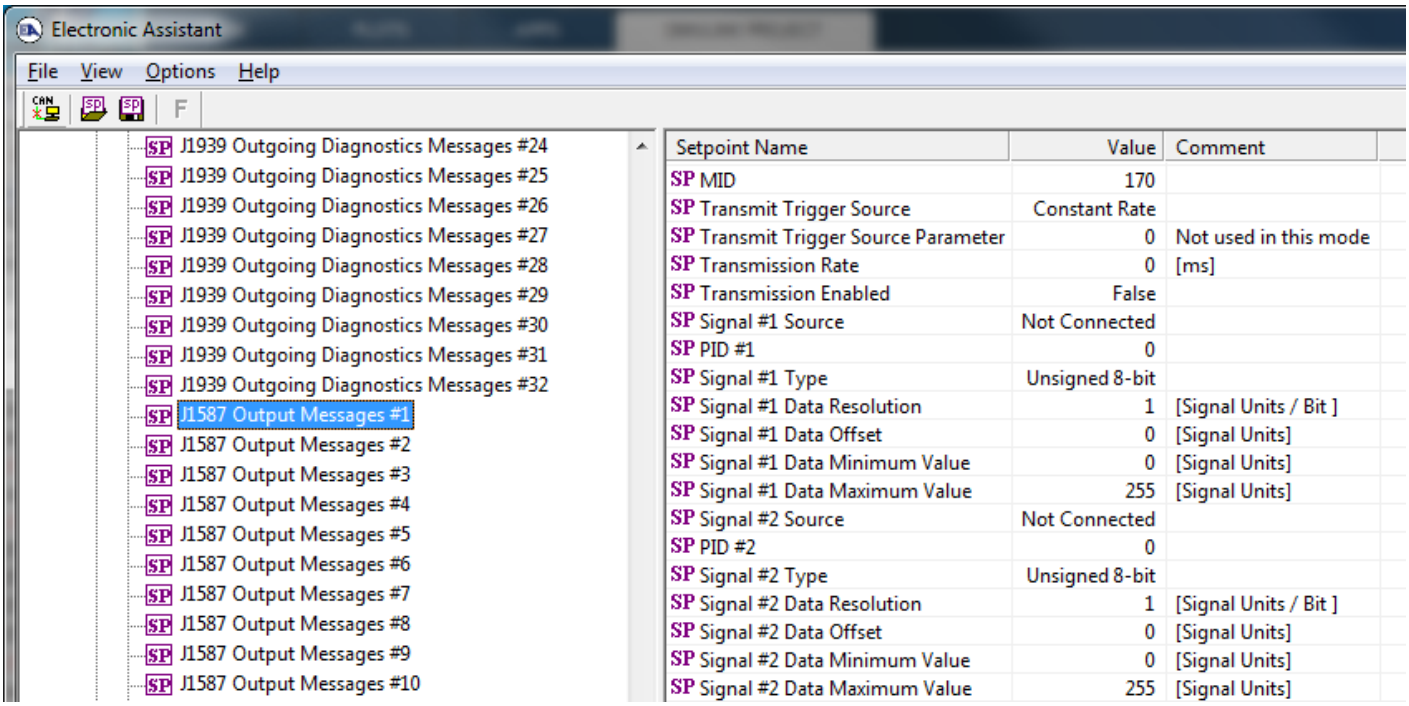


Figure 13: J1587 Output Message definition

Parameter name	***Value range***	***Comment***
MID	0…255	Defined the SPN to read in from received DM1 messages.
Transmit Trigger Source	Constant rate, J1939 message reception, J1587 message reception	The message transmission event trigger source. Constant rate or message reception-based triggering.
Transmit Trigger Source Parameter	1…32	In case message reception-based triggering is selected, this defines the corresponding message number.
Transmission Rate	0…60000	In case constant rate is selected, this defines the message transmission interval in milliseconds.
Transmission Enabled	True, false	Enables the message transmission.
Signal #x Source	Not connected, J1939 Rx message 1…32, J1587 Rx message 1…32	Data source to use for this message. Please also see the J1939 Input Signal based triggering note below.
PID #x	0…1024	The PID to use.
Signal #x Type	undefined (0), uint8 (1), uint16 (2), uint32 (3), sint8 (4), sint16 (5), sint32 (6),	Signal data type to use.

	float16 (7), float32 (8)	
Signal #x Data Resolution	0…(*MAX FLOAT*)	Data resolution to use, how many units / bit.
Signal #x Data Offset	(*MIN FLOAT*) …(*MAX FLOAT*)	Data offset to use.
Signal #x Data Minimum Value	(*MIN FLOAT*) … Signal Data Maximum Value	Data minimum value.
Signal #x Data Maximum Value	Signal Data Minimum Value … (*MAX FLOAT*)	Data maximum value.

A note for CAN message reception based triggering: In case multiple J1939 Input Signals are receiving the same PGN, only the last J1939 Input Signal index can be used as a trigger for a J1587 Output Message.

1.13. *J1587 Input Signals #x (AX140400/AX140322 only)*

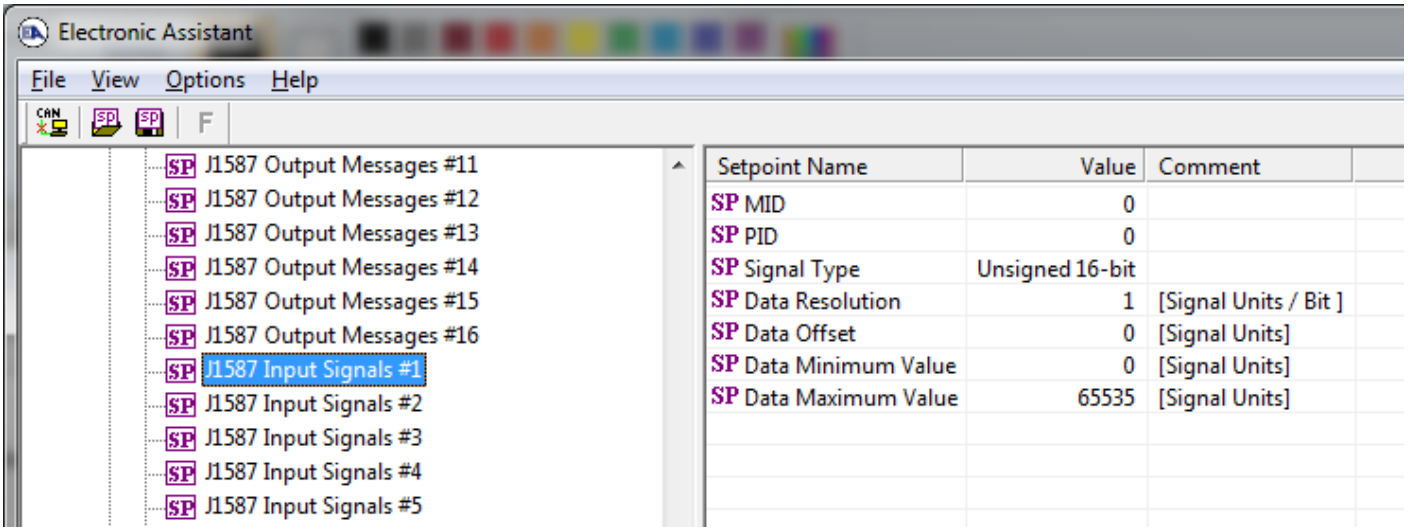


Figure 14: J1587 Input Signals #1 definitions

Parameter name	***Value range***	***Comment***
MID	0…255	Defines the MID to listen.
PID	0…1024	The PID to read in from a message with MID defined above.
Signal Type	undefined (0), uint8 (1), uint16 (2), uint32 (3), sint8 (4), sint16 (5), sint32 (6), float16 (7), float32 (8)	Signal data type to use.
Data Resolution	0…(*MAX FLOAT*)	Data resolution to use, how many units / bit.
Data Offset	(*MIN FLOAT*) …(*MAX FLOAT*)	Data offset to use.
Data Minimum Value	(*MIN FLOAT*) … Signal Data Maximum Value	Data minimum value.
Data Maximum Value	Signal Data Minimum Value … (*MAX FLOAT*)	Data maximum value.

1.14. *J1587 Diagnostics to Monitor (AX140400/AX140322 only)*

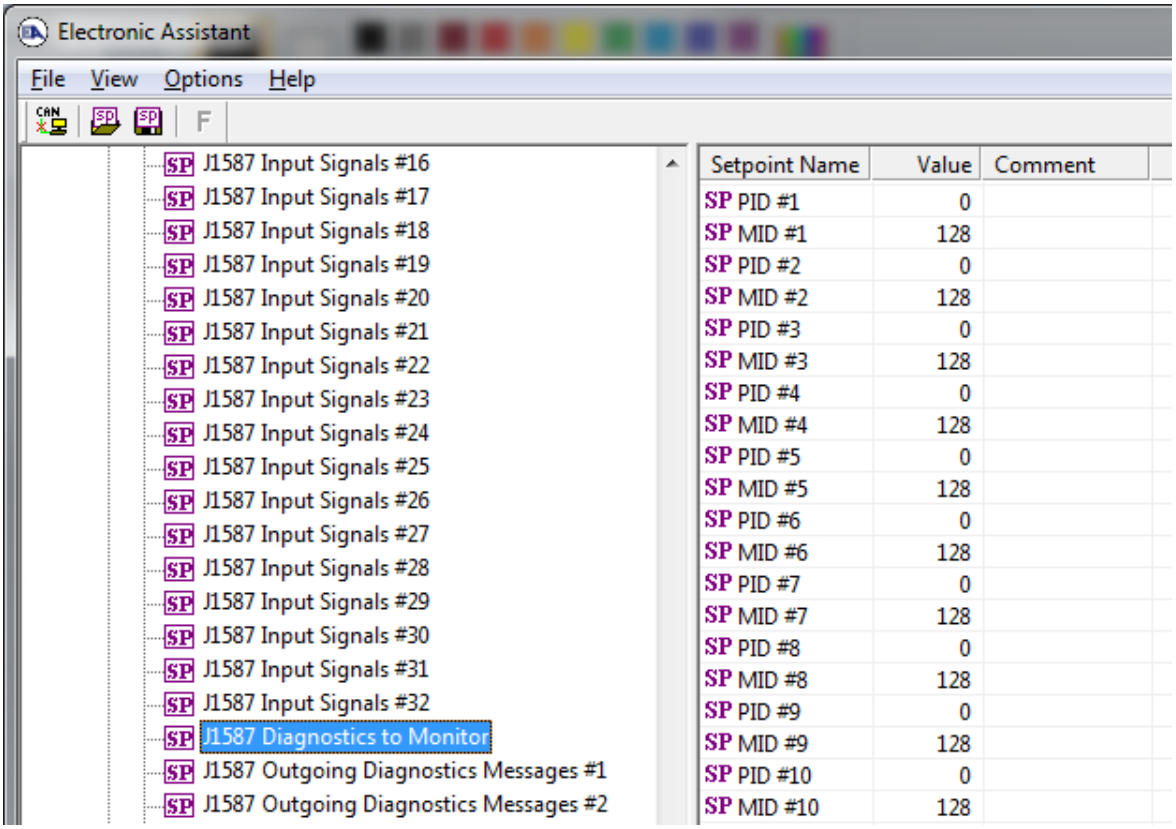


Figure 15: J1587 Diagnostics to Monitor definitions

Parameter name	***Value range***	***Comment***
PID #x	0…1024	Defines the diagnostic PID to listen from received PID194 messages.
MID #x	0…255	Defines the MID for the PID194 messages to listen.

1.15. *J1587 Outgoing Diagnostics Messages #x (AX140400/AX140322 only)*

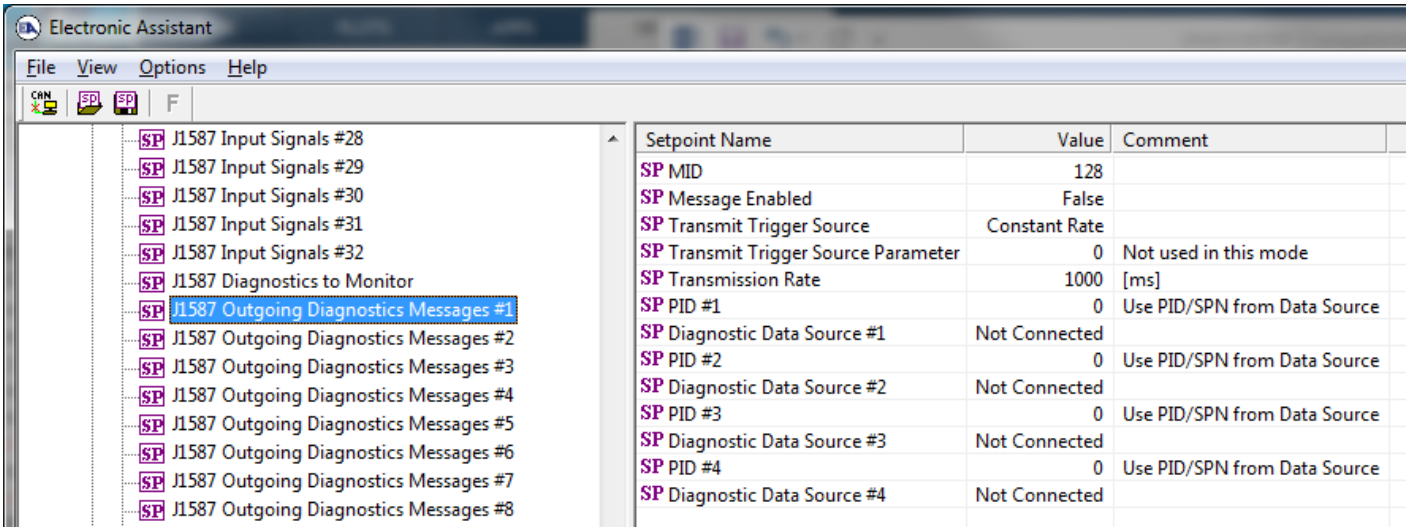


Figure 16: J1587 Outgoing Diagnostics Messages definitions

Parameter name	***Value range***	***Comment***
MID	0…255	Defines the MID to use in the diagnostic message.
Message enabled	True, false	Enables the transmission of the diagnostic message.
Transmit Trigger Source	Constant rate, J1939 message reception, J1587 message reception	The diagnostic message transmission event trigger source. Constant rate or message reception-based triggering.
Transmit Trigger Source Parameter	1…32	In case message reception-based triggering is selected, this defines the corresponding received diagnostic message number.
Transmission Rate	0…60000	In case constant rate is selected, this defines the message transmission interval in milliseconds.
PID #x	0…1024	The PID to include into this diagnostic message.
Diagnostic Data Source #x	Not connected, J1939 Rx Diagnostics 1…32, J1587 Rx Diagnostics 1…16	The diagnostic data source for this J1587 diagnostic message.

1.16. *Build-in J1939-J1587 Data Mappings (AX140400/AX140322 only)*

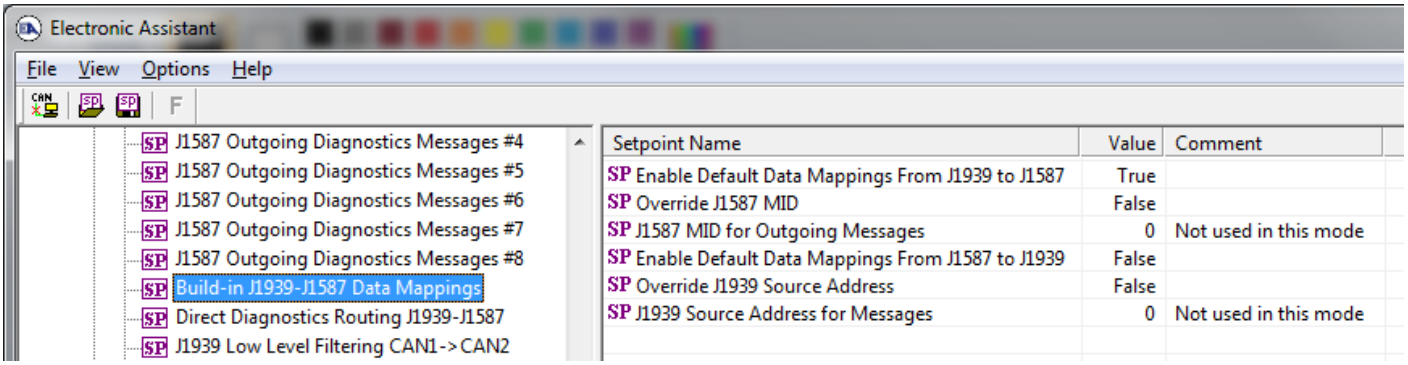


Figure 17: Build-in J1939-J1587 Data Mappings definitions

Parameter name	***Value range***	***Comment***
Enable Default Data Mappings from J1939 to J1587	False, true	Selects whether the default data mappings from J1939 to J1587 are enabled. See section 0 for details.
Override J1587 MID	False, true	Enables the overriding of MID in the forwarded messages.
J1587 MID for Outgoing Messages	0…255	In case MID override is enabled, the MID to use can be specified using this setpoint.
Enable Default Data Mappings from J1587 to J1939	False, true	Selects whether the default data mappings from J1587 to J1939 are enabled. See section 0 for details.
Override J1939 Source Address	False, true	Enables the overriding of Source Address in the forwarded messages.
J1939 Source Address for Messages	0…255	In case Source Address override is enabled, this sets the Source Address to use.

1.17. *Direct Diagnostics Routing J1939-J1587 (AX140400/AX140322 only)*

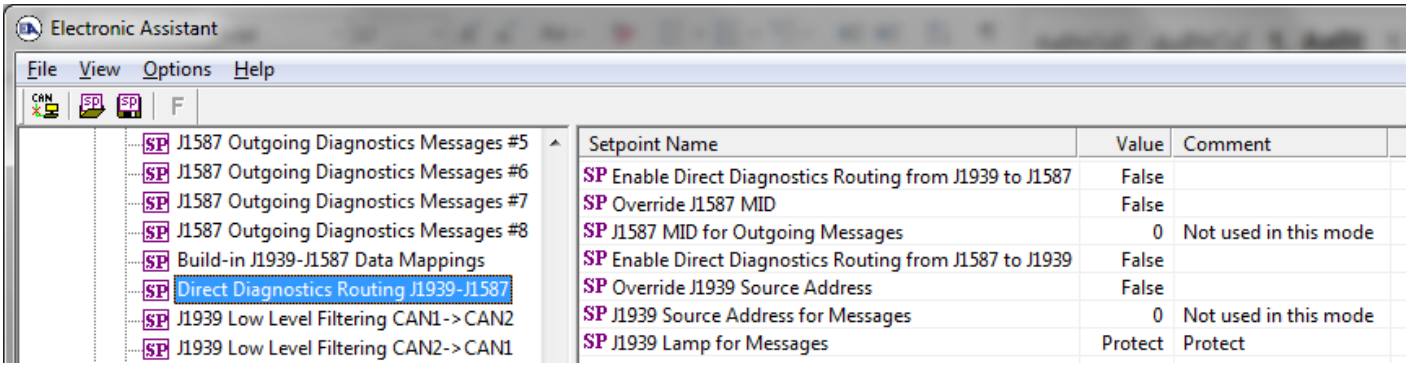


Figure 18: Direct Diagnostics Routing J1939-J1587

Parameter name	***Value range***	***Comment***
Enable Direct Diagnostics Routing from J1939 to J1587	False, true	Selects whether the direct diagnostics routing from J1939 to J1587 is enabled. See section 0 for details.
Override J1587 MID	False, true	Enables the overriding of MID in the forwarded diagnostics messages.
J1587 MID for Outgoing Messages	0…255	In case MID override is enabled, the MID to use can be specified using this setpoint.
Enable Direct Diagnostics Routing from J1587 to J1939	False, true	Selects whether the direct diagnostics routing from J1587 to J1939 are enabled. See section 0 for details.
Override J1939 Source Address	False, true	Enables the overriding of Source Address in the forwarded diagnostics messages.
J1939 Source Address for Messages	0…255	In case Source Address override is enabled, this sets the Source Address to use.
J1939 Lamp for Messages	0=Protect, 1=Amber, 2=Red stop, 3=Malfunction	This defines the Lamp code to use with the J1587 messages that are forwarded into J1939.

1.18. *CANopen RPDO Parameters (AX140200/AX140321 only)*

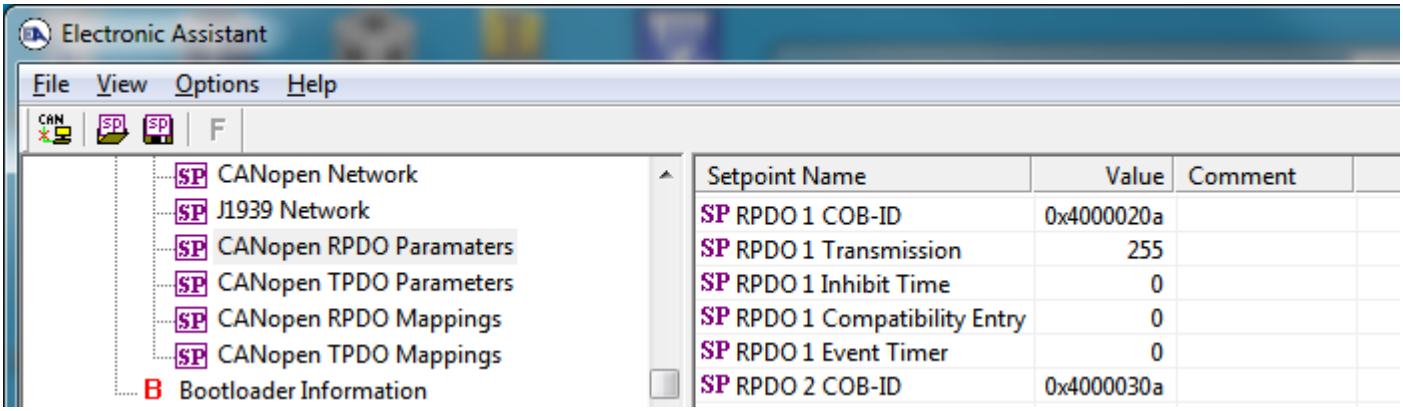


Figure 19: CANopen RPDO Parameters

Parameter name	***Value range***	***Comment***
RPDO x COB-ID	<uint32>	CANopen RPDO COB-ID, refer to CANopen documentation for details.
RPDO x Transmission	<uint8>	CANopen RPDO Transmission Parameter, refer to CANopen documentation for details.
RPDO x Inhibit Time	<uint16>	CANopen RPDO Inhibit Time, refer to CANopen documentation for details.
RPDO x Compatibility Entry	<uint8>	CANopen RPDO Compatibilty parameter, refer to CANopen documentation for details.
RPDO x Event Timer	<uint16>	CANopen RPDO Event Timer, refer to CANopen documentation for details.

Devices having a CANopen interface, such as the AX140200/AX140321 support basic configuration of the CANopen parameters using EA. The PDO Parameters for the first four receive PDOs can be configured using this setpoint group.

Please note, the CANopen RPDO Parameter settings are only applied at device boot up. After changing these parameters, a power cycle will be needed for taking the new settings into use!

1.19. *CANopen TPDO Parameters (AX140200/AX140321 only)*

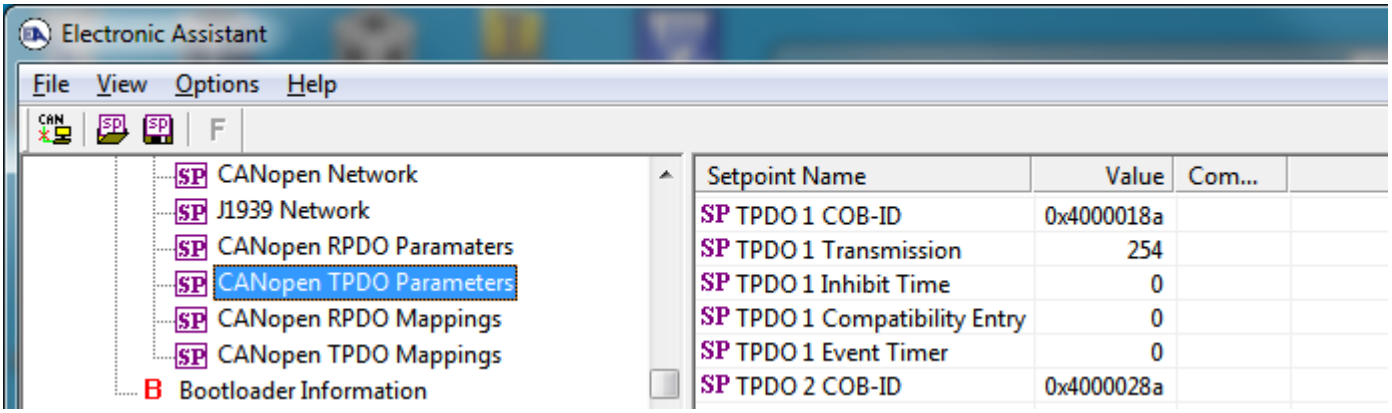


Figure 20: CANopen TPDO Parameters

Parameter name	***Value range***	***Comment***
TPDO x COB-ID	<uint32>	CANopen TPDO COB-ID, refer to CANopen documentation for details.
TPDO x Transmission	<uint8>	CANopen TPDO Transmission Parameter, refer to CANopen documentation for details.
TPDO x Inhibit Time	<uint16>	CANopen TPDO Inhibit Time, refer to CANopen documentation for details.
TPDO x Compatibility Entry	<uint8>	CANopen TPDO Compatibilty parameter, refer to CANopen documentation for details.
TPDO x Event Timer	<uint16>	CANopen TPDO Event Timer, refer to CANopen documentation for details.

Devices having a CANopen interface, such as the AX140200/AX140321 support basic configuration of the CANopen parameters using EA. The PDO Parameters for the first four transmit PDOs can be configured using this setpoint group.

Please note, the CANopen TPDO Parameter settings are only applied at device boot up. After changing these parameters, a power cycle will be needed for taking the new settings into use!

1.20. *CANopen RPDO Mappings (AX140200/AX140321 only)*

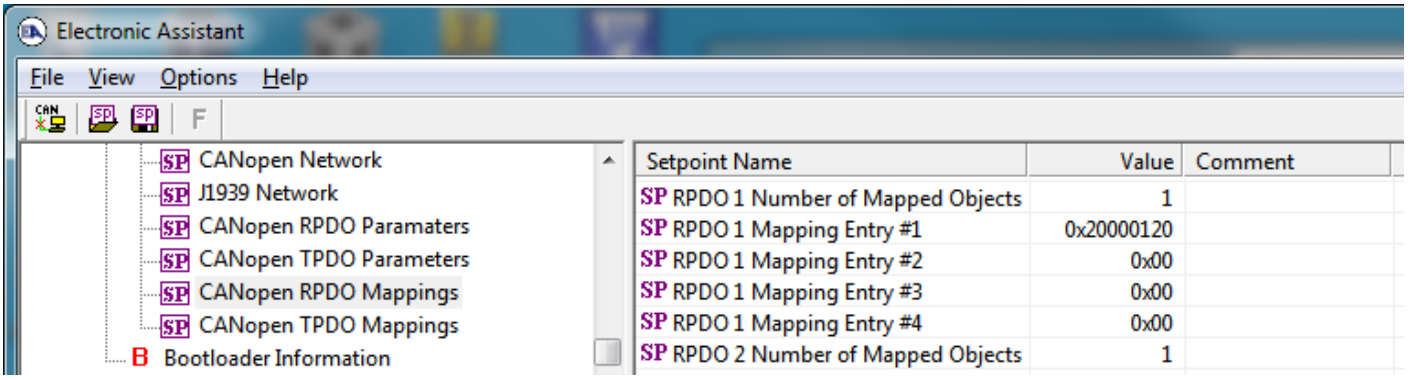


Figure 21: CANopen RPDO Mappings

Parameter name	***Value range***	***Comment***
RPDO x Number of Mapped Objects	0 … 4	CANopen RPDO number of mapped objects, refer to CANopen documentation for details.
RPDO x Mapping Entry #1	<uint32>	CANopen RPDO mapping entry, refer to CANopen documentation for details.
RPDO x Mapping Entry #2	<uint32>	CANopen RPDO mapping entry, refer to CANopen documentation for details.
RPDO x Mapping Entry #3	<uint32>	CANopen RPDO mapping entry, refer to CANopen documentation for details.
RPDO x Mapping Entry #4	<uint32>	CANopen RPDO mapping entry, refer to CANopen documentation for details.

Devices having a CANopen interface, such as the AX140200/AX140321 support basic configuration of the CANopen parameters using EA. The PDO Data Mappings for the first four receive PDOs can be configured using this setpoint group.

Please note, the CANopen RPDO Mapping settings are only applied at device boot up. After changing these parameters, a power cycle will be needed for taking the new settings into use!

1.21. *CANopen TPDO Mappings (AX140200/AX140321 only)*

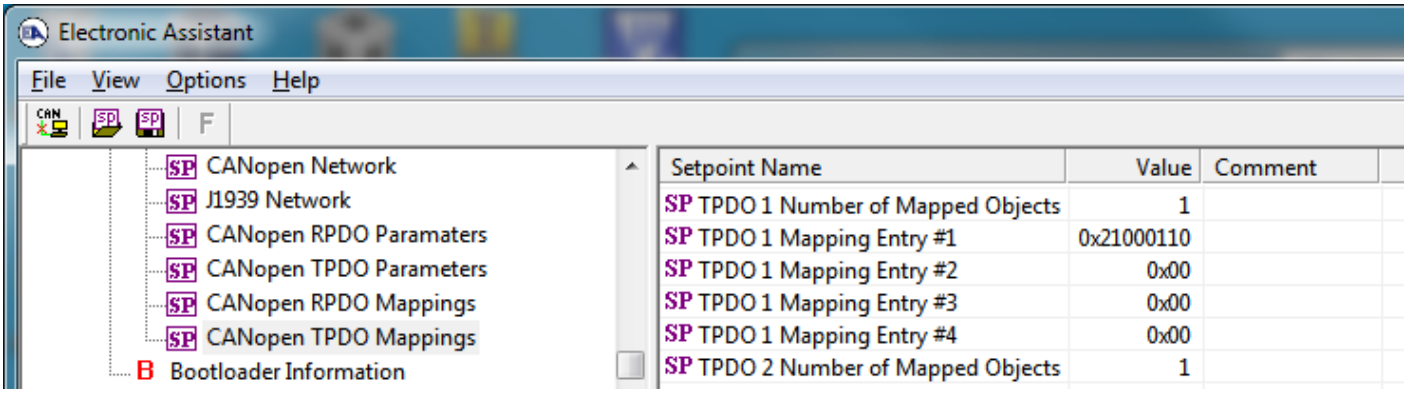


Figure 22: CANopen TPDO Mappings

Parameter name	***Value range***	***Comment***
TPDO x Number of Mapped Objects	0 … 4	CANopen TPDO number of mapped objects, refer to CANopen documentation for details.
TPDO x Mapping Entry #1	<uint32>	CANopen TPDO mapping entry, refer to CANopen documentation for details.
TPDO x Mapping Entry #2	<uint32>	CANopen TPDO mapping entry, refer to CANopen documentation for details.
TPDO x Mapping Entry #3	<uint32>	CANopen TPDO mapping entry, refer to CANopen documentation for details.
TPDO x Mapping Entry #4	<uint32>	CANopen TPDO mapping entry, refer to CANopen documentation for details.

Devices having a CANopen interface, such as the AX140200/AX140321 support basic configuration of the CANopen parameters using EA. The PDO Data Mappings for the first four transmit PDOs can be configured using this setpoint group.

Please note, the CANopen TPDO Mapping settings are only applied at device boot up. After changing these parameters, a power cycle will be needed for taking the new settings into use!

1.22. *Constant Data List*

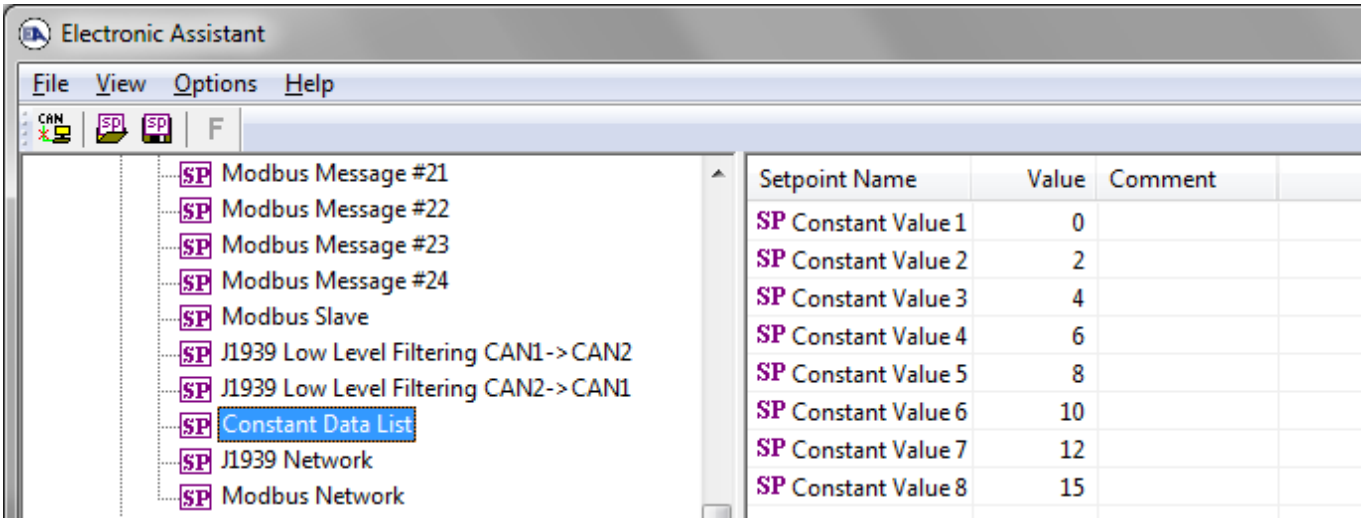


Figure 23: Constant Data List

Parameter name	***Value range***	***Comment***
Constant Value 1	<float>	User configurable constant data
Constant Value 2	<float>	User configurable constant data
Constant Value 3	<float>	User configurable constant data
Constant Value 4	<float>	User configurable constant data
Constant Value 5	<float>	User configurable constant data
Constant Value 6	<float>	User configurable constant data
Constant Value 7	<float>	User configurable constant data
Constant Value 8	<float>	User configurable constant data

The constant data values can be used in CAN transmit message and Modbus Message data sources. The values are user configurable to suit the application in question.

1.23. *Request PGN Configuration*

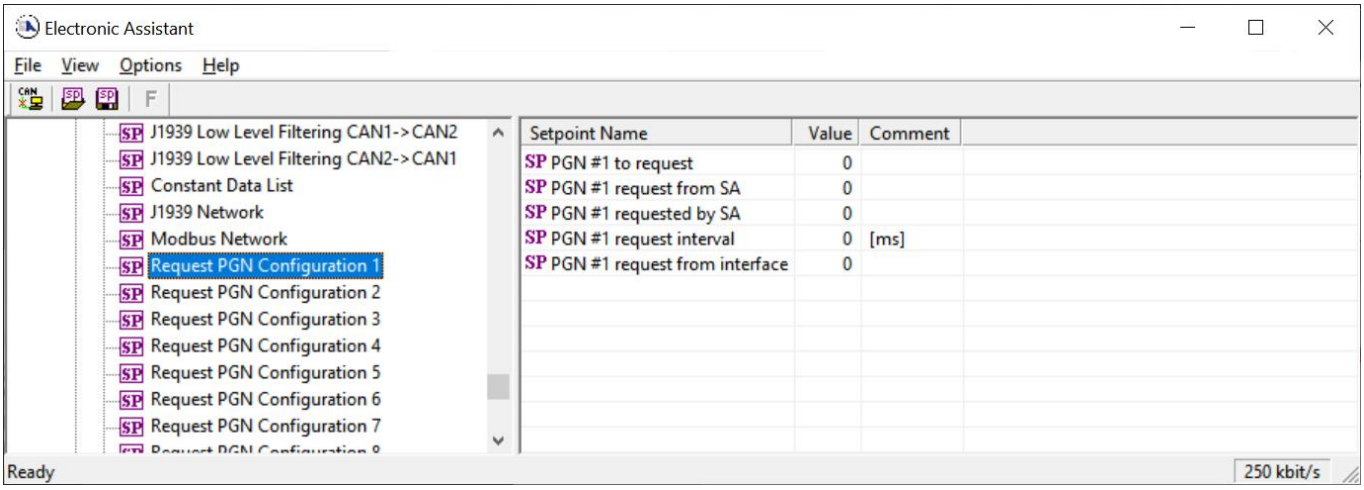


Figure 24: Request PGN configuration setpoints

Parameter name	***Value range***	***Comment***
PGN #x to request	0 … 0x3FFFF	The J1939 PGN to request
PGN #x request from SA	<uint8>	The node address from which to request the PGN (DA byte in outgoing frame)
PGN #x requested by SA	<uint8>	The node address from which the frame is sent (SA byte in the outgoing frame)
PGN #x request interval	0 … 60000ms	Request interval in milliseconds
PGN #x request from interface	0, 1	CAN interface to use

CANopen® CONFIGURATION (only for AX140200&AX140321)

All CANopen® parameters are configured using the CANopen® interface and an eds file supplied with the device. The configuration can be done using CANopen® tools available on the market.

1.1. *Communication objects*

The table below lists all communication objects supported by the Protocol Converter devices having CANopen® interface. The CANopen® stack supports 16 receive and 16 transmit PDOs. (Please note that the list of PDOs is truncated in the table).

Index (hex)	***Object***	***Object Type***	***Data Type***	***Access***	***PDO Mapping***
1000	Device Type	VAR	UNSIGNED32	RO	No
1001	Error Register	VAR	UNSIGNED8	RO	No
1002	Manufacturer Status Register	VAR	UNSIGNED32	RO	No
1003	Pre-Defined Error Field	ARRAY	UNSIGNED32	RO	No
100C	Guard Time	VAR	UNSIGNED16	RW	No
100D	Life Time Factor	VAR	UNSIGNED8	RW	No
1010	Store Parameters	ARRAY	UNSIGNED32	RW	No
1011	Restore Default Parameters	ARRAY	UNSIGNED32	RW	No
1016	Consumer Heartbeat Time	ARRAY	UNSIGNED32	RW	No
1017	Producer Heartbeat Time	VAR	UNSIGNED16	RW	No
1018	Identity Object	RECORD		RO	No
1020	Verify Configuration	ARRAY	UNSIGNED32	RW	No
1029	Error Behaviour	ARRAY	UNSIGNED8	RW	No
1400	RPDO1 Communication Parameter	RECORD		RW	No
1401	RPDO2 Communication Parameter	RECORD		RW	No
…	…	…		…	…
140F	RPDO16 Communication Parameter	RECORD		RW	No
1600	RPDO1 Mapping Parameter	RECORD		RO	No
1601	RPDO2 Mapping Parameter	RECORD		RO	No
…	…	…		…	…
160F	RPDO16 Mapping Parameter	RECORD		RO	No
1800	TPDO1 Communication Parameter	RECORD		RW	No
1801	TPDO2 Communication Parameter	RECORD		RW	No
…	…	…		…	…
180F	TPDO16 Communication Parameter	RECORD		RW	No
1A00	TPDO1 Mapping Parameter	RECORD		RW	No
1A01	TPDO2 Mapping Parameter	RECORD		RW	No
…	…	…		…	…
1A0F	TPDO16 Mapping Parameter	RECORD		RW	No

Per the CANopen ® standard DS-301, the following procedure shall be used for re-mapping, and is the same for both RPDOs and TPDOs.

- Destroy the PDO by setting bit **exists** (most significant bit) of sub-index 01h of the according PDO communication parameter to 1b
- Disable mapping by setting sub-index 00h of the corresponding mapping object to 0
- Modify the mapping by changing the values of the corresponding sub-indices
- Enable mapping by setting sub-index 00h to the number of mapped objects
- Create the PDO by setting bit **exists** (most significant bit) of sub-index 01h of the according PDO communication parameter to 0b

4.1.1. 1000h Device Type

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1000	0	UINT32	RO	No	0x0020019D	0x0020019D	DS-413

4.1.2. 1001h Error Register

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1001	0	UINT8	RO	No	0, 1	0	Error register

4.1.3. 1002h Manufacturer Status Object

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1002	0	UINT32	RO	No	UINT32	0	Manufacturer debug information

4.1.4. 1003h Pre-Defined Error Field

Please note, that the Protocol Converter does not implement any CANopen EMCY for signaling errors in the routing application. The Communication errors flagged by the CANopen stack are the only available EMCY codes.

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1003	0	UINT8	RW	No	15	15	Number of subindexes / reset error codes
	1	UINT32	RO		UINT32	0	EMCY error code #1
	2						EMCY error code #2
	3						EMCY error code #3
	4						EMCY error code #4
	5						EMCY error code #5
	6						EMCY error code #6
	7						EMCY error code #7
	8						EMCY error code #8
	9						EMCY error code #9
	10						EMCY error code #10
	11						EMCY error code #11
	12						EMCY error code #12
	13						EMCY error code #13
	14						EMCY error code #14
	15						EMCY error code #15

4.1.5. 1010h Store Parameters

Please note, that object 1010h saves only CANopen parameters. All J1939 parameters are saved automatically after editing (Axiomatic Electronic Assistant).

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1010	0	UINT8	RO	No	4	4	Number of subindexes
	1	UINT32	RW		save	1	Write 0x65766173 ('e', 'v', 'a', 's') for storing ALL parameters
	2						Write 0x65766173 ('e', 'v', 'a', 's') for storing Communication parameters
	3						Write 0x65766173 ('e', 'v', 'a', 's') for storing Application parameters
	4						Write 0x65766173 ('e', 'v', 'a', 's') for storing Manufacturer parameters

4.1.6. 1011h Restore Parameters

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1011	0	UINT8	RO	No	4	4	Number of subindexes
	1	UINT32	RW		load	1	Write 0x4616F6C ('d', 'a', 'o', 'l') for restoring ALL parameters
	2						Write 0x4616F6C ('d', 'a', 'o', 'l') for restoring Communication parameters
	3						Write 0x4616F6C ('d', 'a', 'o', 'l') for restoring Application parameters
	4						Write 0x4616F6C ('d', 'a', 'o', 'l') for restoring Manufacturer parameters

4.1.7. 1016h Consumer Heartbeat Time

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1016	0	UINT8	RO	No	4	4	Number of subindexes
	1	UINT32	RW		UINT32	0	Consumer heartbeat time bits 31-24: reserved bits 23-16: Node ID bits 15-0: heartbeat time in milliseconds
	2						
	3						
	4						

4.1.8. 1017h Producer Heartbeat Time

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1017	0	UINT16	RW	No	10-65000	0	Producer heartbeat time in milliseconds

4.1.9. 1018h Identity Object

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1018	0	UINT8	RO	No	4	4	Number of subindexes
	1	UINT32			UINT32	0x55	Vendor ID (Axiomatic Technologies)
	2					0xAA140200/ 0xAA140321	Product Code
	3						Revision Number
	4						Serial Number

4.1.10. 1020h Verify Configuration

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1020	0	UINT8	RO	No	4	4	Number of subindexes
	1	UINT32			UINT32		Configuration date: DD-MM-YYYY
	2						Configuration time: HH-MM

4.1.11. 1029h Error Behavior

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1029	0	UINT8	RO	No	6	4	Number of subindexes
	1		RW		0-2	1 (no change)	State transition on Comm. fault
	2						State transition on DI fault
	3						State transition on AI fault
	4						State transition on DO fault
	5						State transition on AO fault
	6						State transition on other faults

4.1.12. 1400h-140Fh RPDO 1-16 Communication Parameters

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1400 … 140F	0	UINT8	RO	No	4	4	Number of subindexes
	1	UINT32	RW		UINT32	0x4000020A 0x4000030A 0x4000040A 0x4000050A 0x4000021A 0x4000022A 0x4000023A 0x4000024A 0x4000025A 0x4000026A 0x4000027A 0x4000029A 0x400002AA 0x400002BA 0x400002CA 0x400002DA	COB-ID (1-16)
	2	UINT8			UINT8	0xFF	Transmission type
	3	UINT16			UINT16	0	Inhibit time
	4	UINT8			UINT8	0	Compatibility entry
	5	UINT16			UINT16	0	Event timer

4.1.13. 1600h-160Fh RPDO 1-16 Mapping Parameters

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1600 … 160F	0	UINT8	RW	No	0-4	1	Number of subindexes
	1	UINT32			UINT32	0x20000120 … 0x200F0120	Receive data object #x
	2					0	Not used by default
	3					0	Not used by default
	4					0	Not used by default

4.1.14. 1800h-180Fh TPDO 1-16 Communication Parameters

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1800 … 180F	0	UINT8	RO	No	4	4	Number of subindexes
	1	UINT32	RW		UINT32	0x4000018A 0x4000028A 0x4000038A 0x4000048A 0x4000039A 0x400003AA 0x400003BA 0x400003CA 0x400003DA 0x400003EA 0x400003FA 0x4000041A 0x4000042A 0x4000043A 0x4000044A 0x4000045A	COB-ID (1-16)
	2	UINT8			UINT8	0xFE	Transmission type
	3	UINT16			UINT16	0	Inhibit time
	4	UINT8			UINT8	0	Compatibility entry
	5	UINT16			UINT16	0	Event timer

4.1.15. 1A00h-1A0Fh TPDO 1-16 Mapping Parameters

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
1A00 … 1A0F	0	UINT8	RW	No	0-4	1	Number of subindexes
	1	UINT32			UINT32	0x21000120 … 0x210F0120	Transmit data object #x
	2					0	Not used by default
	3					0	Not used by default
	4					0	Not used by default

1.2. *Manufacturer objects*

Index (hex)	*Object*	*Object Type*	*Data Type*	*Access*	*PDO Mapping*
2000	CANopen RX data 0	ARRAY	UNSIGNED32	RW	Yes
2001	CANopen RX data 1	ARRAY	UNSIGNED32	RW	Yes
…	…	…	…	…	…
200F	CANopen RX data 15	ARRAY	UNSIGNED32	RW	Yes
2100	CANopen TX data 0	ARRAY	UNSIGNED32	RW	Yes
2101	CANopen TX data 1	ARRAY	UNSIGNED32	RW	Yes
…	…	…	…	…	…
210F	CANopen TX data 15	ARRAY	UNSIGNED32	RW	Yes
2180	CANopen TX bit data 0	VAR	UNSIGNED8	RO	Yes
…	…	…	…	…	…
2183	CANopen TX bit data 3	VAR	UNSIGNED8	RO	Yes
2200	CANopen RX data scaler coefficient 0	ARRAY	REAL32	RW	No
2201	CANopen RX data scaler coefficient 1	ARRAY	REAL32	RW	No
…	…	…	…	…	…
220F	CANopen RX data scaler coefficient 15	ARRAY	REAL32	RW	No
2210	CANopen RX data scaler offset 0	ARRAY	REAL32	RW	No
2211	CANopen RX data scaler offset 1	ARRAY	REAL32	RW	No
…	…	…	…	…	…
221F	CANopen RX data scaler offset 15	ARRAY	REAL32	RW	No
2220	CANopen TX data scaler coefficient 0	ARRAY	REAL32	RW	No
2221	CANopen TX data scaler coefficient 1	ARRAY	REAL32	RW	No
…	…	…	…	…	…
222F	CANopen TX data scaler coefficient 15	ARRAY	REAL32	RW	No
2230	CANopen TX data scaler offset 0	ARRAY	REAL32	RW	No
2231	CANopen TX data scaler offset 1	ARRAY	REAL32	RW	No
…	…	…	…	…	…
223F	CANopen TX data scaler offset 15	ARRAY	REAL32	RW	No
23x0	CANopen RX Message x Modbus Type	ARRAY	UNSIGNED8	RW	No
23x1	CANopen RX Message x Modbus Address	ARRAY	UNSIGNED32	RW	No
23x3	CANopen RX Message x Modbus Maximum	ARRAY	REAL32	RW	No
23x4	CANopen RX Message x Modbus Minimum	ARRAY	REAL32	RW	No
23x5	CANopen RX Message x Modbus Resolution	ARRAY	REAL32	RW	No
23x6	CANopen RX Message x Modbus Offset	ARRAY	REAL32	RW	No
24x0	CANopen TX Message x Modbus Type	ARRAY	UNSIGNED8	RW	No
24x1	CANopen TX Message x Modbus Address	ARRAY	UNSIGNED32	RW	No
24x3	CANopen TX Message x Modbus Maximum	ARRAY	REAL32	RW	No
24x4	CANopen TX Message x Modbus Minimum	ARRAY	REAL32	RW	No
24x5	CANopen TX Message x Modbus Resolution	ARRAY	REAL32	RW	No
24x6	CANopen TX Message x Modbus Offset	ARRAY	REAL32	RW	No
2800	CANopen Master Rx Data	ARRAY	UNSIGNED32	RW	Yes
2900	CANopen Master Tx Data	ARRAY	UNSIGNED32	RW	Yes
2901	CANopen Master Tx Data Mask	ARRAY	UNSIGNED32	RW	No
2902	CANopen Master Tx Data Right Shift	ARRAY	UNSIGNED32	RW	No
2903	CANopen Master Tx Data Left Shift	ARRAY	UNSIGNED32	RW	No
2A00	CANopen EMCY to monitor, Code	ARRAY	UNSIGNED16	RW	No
2A01	CANopen EMCY to monitor, Reg	ARRAY	UNSIGNED8	RW	No
2A02	CANopen EMCY to monitor, Node	ARRAY	UNSIGNED8	RW	No
2A03	Forward CANopen EMCYs to Modbus	VAR	UNSIGNED8	RW	No
2A04	Forward CANopen EMCYs from Node ID	VAR	UNSIGNED8	RW	No

2A05	Forward EMCYs into Modbus address	VAR	UNSIGNED16	RW	No
3000	CANopen RX scaled data 0	ARRAY	UNSIGNED32	RW	Yes
3001	CANopen RX scaled data 1	ARRAY	UNSIGNED32	RW	Yes
…	…	…	…	…	…
300F	CANopen RX scaled data 15	ARRAY	UNSIGNED32	RW	Yes
3100	CANopen TX scaled data 0	ARRAY	UNSIGNED32	RW	Yes
3101	CANopen TX scaled data 1	ARRAY	UNSIGNED32	RW	Yes
…	…	…	…	…	…
310F	CANopen TX scaled data 15	ARRAY	UNSIGNED32	RW	Yes
4000	CANopen TX Message Triggers	ARRAY	UNSIGNED8	RW	No
4001	CANopen TX Message Parameters	ARRAY	UNSIGNED8	RW	No
4010	CANopen RX Message 0 Out Destination	ARRAY	UNSIGNED8	RW	No
4011	CANopen RX Message 1 Out Destination	ARRAY	UNSIGNED8	RW	No
…	…	…	…	…	…
401F	CANopen RX Message 15 Out Destination	ARRAY	UNSIGNED8	RW	No
4020	CANopen TX Message 0 In Source	ARRAY	UNSIGNED8	RW	No
4021	CANopen TX Message 1 In Source	ARRAY	UNSIGNED8	RW	No
…	…	…	…	…	…
402F	CANopen TX Message 15 In Source	ARRAY	UNSIGNED8	RW	No
4100	CANopen EMCY Data Source	ARRAY	UNSIGNED8	RW	No
4101	CANopen EMCY Code/Register	ARRAY	UNSIGNED32	RW	No
4102	CANopen EMCY Manufacturer Code	ARRAY	UNSIGNED32	RW	No
4103	CANopen EMCY Source Address	ARRAY	UNSIGNED8	RW	No
4120	CANopen BitMsg 0 In Source	ARRAY	UNSIGNED8	RW	No
…	…	…	…	…	…
4123	CANopen BitMsg 3 In Source	ARRAY	UNSIGNED8	RW	No
5555	Start in Operational Mode	VAR	BOOLEAN	RW	No
5556	Delay before starting in Operational Mode	VAR	UNSIGNED16	RW	No
5557	RS485 Interface Mode	VAR	UNSIGNED8	RW	No
5600	Set J1939 Baud rate	VAR	UNSIGNED32	RW	No
5601	Current J1939 Baud rate	VAR	UNSIGNED16	RO	No
5610	RS485 Settings	ARRAY	UNSIGNED8	RW	No
5611	RS485 Custom Baud Rate	VAR	UNSIGNED32	RW	No
5A00	CANopen Master Script Settings #0	ARRAY	UNSIGNED32	RW	No
5A01	CANopen Master Script Settings #1	ARRAY	UNSIGNED32	RW	No
…	…	…	…	…	…
5A0F	CANopen Master Script Settings #15	ARRAY	UNSIGNED32	RW	No
5B00	CANopen Master Script Data #0	ARRAY	UNSIGNED32	RW	No
5B01	CANopen Master Script Data #1	ARRAY	UNSIGNED32	RW	No
…	…	…	…	…	…
5B0F	CANopen Master Script Data #15	ARRAY	UNSIGNED32	RW	No
5B50	Set CANopen Baud Rate	VAR	UNSIGNED8	RW	No
5B51	Set CANopen Node ID	VAR	UNSIGNED8	RW	No

4.3.1. 2000h-200Fh CANopen RX data 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2000 … 200F	0	UINT8	RO	No	6	6	Number of subindexes
	1	UINT32	RW	Yes	UINT32	0	CANopen rx data
	2					0	
	3					0	
	4					0	
	5					0x7	RX data type, see Table 13
	6					0xFF	Bitmask for bitfield data type

4.3.2. 2100h-210Fh CANopen TX data 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2100 … 210F	0	UINT8	RO	No	6	6	Number of subindexes
	1	UINT32	RW	Yes	UINT32	0	CANopen tx data
	2					0	
	3					0	
	4					0	
	5					0x7	TX data type, see Table 13
	6					0xFF	Bitmask for bitfield data type

4.3.3. 2180h-2183h CANopen TX bit data 0-3

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2180 … 2183	0	UINT8	RO	Yes	0-255	0	CANopen tx bit data, set by sources defined by 4120h-4123h, see 4.3.41

4.3.4. 2200h-220Fh CANopen RX data scaler coefficient 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2200 … 220F	0	UINT8	RO	No	4	4	Number of subindexes
	1	REAL32	RW		REAL32	1.0 (0x3F800000)	Rx data scaler
	2					1.0 (0x3F800000)	
	3					1.0 (0x3F800000)	
	4					1.0 (0x3F800000)	

4.3.5. 2210h-221Fh CANopen RX data scaler offset 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2210 … 221F	0	UINT8	RO	No	4	4	Number of subindexes
	1	REAL32	RW		REAL32	0.0 (0x0)	Rx data offset
	2					0.0 (0x0)	
	3					0.0 (0x0)	
	4					0.0 (0x0)	

4.3.6. 2220h-222Fh CANopen TX data scaler coefficient 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2220	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	1.0 (0x3F800000)	Tx data scaler
222F	2					1.0 (0x3F800000)	
	3					1.0 (0x3F800000)	
	4					1.0 (0x3F800000)	

4.3.7. 2230h-223Fh CANopen TX data scaler offset 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2230	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0 (0x0)	Tx data offset
223F	2					0.0 (0x0)	
	3					0.0 (0x0)	
	4					0.0 (0x0)	

4.3.8. 23[0]0h-23[F]0h CANopen Rx Message #x Modbus Type

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
23[0]0	0	UINT8	RO	No	4	4	Number of subindexes
…	1		RW		UINT8	0	Modbus type, see Table 8: Modbus (Slave) Types
23[F]0	2					0	
	3					0	
	4					0	

4.3.9. 23[0]1h-23[F]1h CANopen Rx Message #x Modbus Address

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
23[0]1	0	UINT8	RO	No	4	4	Number of subindexes
…	1	UINT32	RW		UINT32	0	Modbus address
23[F]1	2					0	
	3					0	
	4					0	

4.3.10. 23[0]3h-23[F]3h CANopen Rx Message #x Modbus Maximum

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
23[0]3	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0	Modbus maximum value. **Note: a non-zero REAL32 value needs to be programmed to these subindexes for enabling data routing.**
23[F]3	2					0.0	
	3					0.0	
	4					0.0	

4.3.11. 23[0]4h-23[F]4h CANopen Rx Message #x Modbus Minimum

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
23[0]4	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0	Modbus minimum value.
23[F]4	2					0.0	
	3					0.0	
	4					0.0	

4.3.12. 23[0]5h-23[F]5h CANopen Rx Message #x Modbus Resolution

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
23[0]5	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0	Modbus resolution. **Note: a non-zero REAL32 value needs to be programmed to these subindexes for enabling data routing.**
23[F]5	2					0.0	
	3					0.0	
	4					0.0	

4.3.13. 23[0]6h-23[F]6h CANopen Rx Message #x Modbus Offset

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
23[0]6	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0	Modbus offset.
23[F]6	2					0.0	
	3					0.0	
	4					0.0	

4.3.14. 24[0]0h-24[F]0h CANopen Tx Message #x Modbus Type

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
24[0]0	0	UINT8	RO	No	4	4	Number of subindexes
…	1		RW		UINT8	0	Modbus type, see Table 8: Modbus (Slave) Types
24[F]0	2					0	
	3					0	
	4					0	

4.3.15. 24[0]1h-24[F]1h CANopen Tx Message #x Modbus Address

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
24[0]1	0	UINT8	RO	No	4	4	Number of subindexes
…	1	UINT32	RW		UINT32	0	Modbus address
24[F]1	2					0	
	3					0	
	4					0	

4.3.16. 24[0]3h-24[F]3h CANopen Tx Message #x Modbus Maximum

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
24[0]3	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0	Modbus maximum value. **Note: a non-zero REAL32 value needs to be programmed to these subindexes for enabling data routing.**
24[F]3	2					0.0	
	3					0.0	
	4					0.0	

4.3.17. 24[0]4h-24[F]4h CANopen Rx Message #x Modbus Minimum

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
24[0]4	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0	Modbus minimum value.
24[F]4	2					0.0	
	3					0.0	
	4					0.0	

4.3.18. 24[0]5h-24[F]5h CANopen Rx Message #x Modbus Resolution

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
24[0]5	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0	Modbus resolution. **Note: a non-zero REAL32 value needs to be programmed to these subindexes for enabling data routing.**
24[F]5	2					0.0	
	3					0.0	
	4					0.0	

4.3.19. 24[0]6h-24[F]6h CANopen Tx Message #x Modbus Offset

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
24[0]6	0	UINT8	RO	No	4	4	Number of subindexes
…	1	REAL32	RW		REAL32	0.0	Modbus offset.
24[F]6	2					0.0	
	3					0.0	
	4					0.0	

4.3.20. 2800h CANopen Master Rx Data

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2800	0	UINT8	RO	No	16	16	Number of subindexes
	1	UINT32	RW	Yes	UINT32	0	CANopen Master message rx data
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.21. 2900h CANopen Master Tx Data

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2900	0	UINT8	RO	No	16	16	Number of subindexes
	1	UINT32	RW	Yes	UINT32	0	CANopen Master message tx data
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.22. 2901h CANopen Master Tx Data Mask

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2901	0	UINT8	RO	No	16	16	Number of subindexes
	1	UINT32	RW		UINT32	0xFFFFFFFF	CANopen Master message tx data bit mask
	2					0xFFFFFFFF	
	3					0xFFFFFFFF	
	4					0xFFFFFFFF	
	5					0xFFFFFFFF	
	6					0xFFFFFFFF	
	7					0xFFFFFFFF	
	8					0xFFFFFFFF	
	9					0xFFFFFFFF	
	10					0xFFFFFFFF	
	11					0xFFFFFFFF	
	12					0xFFFFFFFF	
	13					0xFFFFFFFF	
	14					0xFFFFFFFF	
	15					0xFFFFFFFF	
	16					0xFFFFFFFF	

4.3.23. 2902h CANopen Master Tx Data Right Shift

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2902	0	UINT8	RO	No	16	16	Number of subindexes
	1	UINT32	RW		UINT32	0	CANopen Master message tx data right bit shift
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.24. 2903h CANopen Master Tx Data Left Shift

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2903	0	UINT8	RO	No	16	16	Number of subindexes
	1	UINT32	RW		UINT32	0	CANopen Master message tx data left bit shift
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.25. 2A00h CANopen EMCY to Monitor, Code

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2A00	0	UINT8	RO	No	16	16	Number of subindexes
	1	UINT16	RW		UINT16	0	EMCY reception acceptance filter. If the Code field of the received EMCY matches this data, a corresponding EMCY index is flagged as active. This reception filtering setting is used together with the configuration in corresponding subindexes of objects 2A01h and 2A02h (all three needs to match for successful reception).
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.26. 2A01h CANopen EMCY to Monitor, Register

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2A01	0	UINT8	RO	No	16	16	Number of subindexes
	1		RW		UINT8	0	EMCY reception acceptance filter. If the Register field of the received EMCY matches this data, a corresponding EMCY index is flagged as active. This reception filtering setting is used together with the configuration in corresponding subindexes of objects 2A00h and 2A02h (all three needs to match for successful reception).
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.27. 2A02h CANopen EMCY to Monitor, Node

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2A02	0	UINT8	RO	No	16	16	Number of subindexes
	1		RW		UINT8	0	EMCY reception acceptance filter. If the sender Node ID of the received EMCY matches this data, a corresponding EMCY index is flagged as active. This reception filtering setting is used together with the configuration in corresponding subindexes of objects 2A00h and 2A01h (all three needs to match for successful reception).
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.28. 2A03h Forward CANopen EMCYs to Modbus

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2A03	0	UINT8	RW	No	0-2	0	0 – No forwarding, 1 – Forward SPNs defined in 2A00h-2A02h 2 – Forward all SPNs from Node ID defined in 2A04h.

4.3.29. 2A04h Forward CANopen EMCYs to Modbus from Node ID

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2A04	0	UINT8	RW	No	0-127	0	Node ID to listen to

4.3.30. 2A05h Forward CANopen EMCYs to Modbus Address

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
2A05	0	UINT16	RW	No	0-1024	0	The Modbus Holding register address starting from which the forwarded EMCYs are available. Please see section 1.10 for more details.

4.3.31. 3000h-300Fh CANopen RX scaled data 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
3000	0	UINT8	RO	No	4	4	Number of subindexes
…	1	UINT32	RW	Yes	UINT32	0	Scaled rx data
300F	2					0	
	3					0	
	4					0	

4.3.32. 3100h-310Fh CANopen TX scaled data 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
3100	0	UINT8	RO	No	4	4	Number of subindexes
…	1	UINT32	RW	Yes	UINT32	0	Scaled tx data
310F	2					0	
	3					0	
	4					0	

4.3.33. 4000h CANopen TX Message Triggers

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4000	0	UINT8	RO	No	16	16	Number of subindexes
	1		RW		enum	0	CANopen TX message triggers, see Table 9: CANopen TX message triggers for details
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.34. 4001h CANopen TX Message Parameters

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4001	0	UINT8	RO	No	16	16	Number of subindexes
	1		RW		enum	0	CANopen TX message trigger parameters, see Table 10: CANopen TX message parameters for details
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.35. 4010h-401Fh CANopen RX Message 0-15 Out Destination

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4010 … 401F	0	UINT8	RO	No	4	4	Number of subindexes
	1		RW		enum	0	CANopen rx message data output destination, see Table 12: CANopen RX messages' out destination configurations
	2					0	
	3					0	
	4					0	

4.3.36. 4020h-402Fh CANopen TX Message 0-15 In Source

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4020 … 402F	0	UINT8	RO	No	4	4	Number of subindexes
	1		RW		enum	5, 9, 13, …	The input source for the data stored in objects 2100h-210Fh. See Table 7: CANopen TX messages' input signal source options
	2					6, 10, 14, …	
	3					7, 11, 15, …	
	4					8, 12, 16, …	

4.3.37. 4100h CANopen EMCY Data Source

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4100	0	UINT8	RO	No	16	16	Number of subindexes
	1		RW		enum	0	CANopen EMCY data source
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.38. 4101h CANopen EMCY Code / Register

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4101	0	UINT8	RO	No	16	16	Number of subindexes
	1	UINT32	RW		UINT32	0	CANopen EMCY messages' Code & Register fields
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.39. 4102h CANopen EMCY Manufacturer Code

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4102	0	UINT8	RO	No	16	16	Number of subindexes
	1	UINT32	RW		UINT32	0	CANopen EMCY messages' Manufacturer code
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.40. 4103h CANopen EMCY Node ID

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4103	0	UINT8	RO	No	16	16	Number of subindexes
	1		RW		UINT8	0	CANopen EMCY messages' node ID
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	
	13					0	
	14					0	
	15					0	
	16					0	

4.3.41. 4120h-4123h CANopen BitMessage 0-3 In Source

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
4120 … 4123	0	UINT8	RO	No	8	8	Number of subindexes
	1		RW		enum	5, 13, …	The input source for the bit messages. Corresponding data objects are 2180h-2183h, see 4.3.3 **The list of input sources include only the J1939 RX messages.** Control source 5 == J1939 CAN Rx 1
	2					6, 14, …	
	3					7, 15, …	
	4					8, 16, …	
	5					9, 17, …	
	6					10, 18, …	
	7					11, 19, …	
	8					12, 20, …	

4.3.42. 5555h Start In Operational Mode

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5555	0	UINT8	RW	No	0-3	0	0 – No action, wait NMT commands 1 – Start directly in operational mode 2 – Start in operational mode and send NMT for starting also other devices

4.3.43. 5556h Start In Operational NMT Delay

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5556	0	UINT16	RW	No	0-65000	1000	Delay in milliseconds before sending the NMT message in case object 5555h is set to ‘2’.

4.3.44. 5557h RS485 Interface Mode

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5557	0	UINT8	RW	No	0, 1	0	0 – Modbus communications 1 – Raw data

4.3.45. 5600h Set J1939 Baud Rate

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5600	0	UINT32	RW	No	set0, set1, set2	0	Write 0x30746573 (‘0’, ‘t’, ‘e’, ‘s’) for 250k baud rate, Write 0x31746573 (‘1’, ‘t’, ‘e’, ‘s’) for 500k baud rate, Write 0x32746573 (‘2’, ‘t’, ‘e’, ‘s’) for 1000k baud rate

4.3.46. 5601h Current J1939 Baud Rate

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5601	0	UINT8	RO	No	250, 500, 1000	250	Current J1939 port baud rate in kbps.

4.3.47. 5610h RS485 Settings

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5610	0	UINT8	RO	No	5	5	Number of subindexes
	1		RW		0-8	3	Baud rate (9600), see Table 17
	2				0, 1	0	Data bits (8 bits), see Table 18
	3				0-2	0	Parity (no parity), see Table 19
	4				0-3	1	Stop bits (1 bit), see Table 20
	5				0, 1	0	Mode: 0 = RTU, 1 = ASCII

4.3.48. 5611h RS485 Custom Baud Rate

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5611	0	UINT32	RW	No	0-460800	0	Custom RS485 baud rate to use in bps

4.3.49. 5A00h-5A0Fh CANopen Master Script Settings 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5A00 … 5A0F	0	UINT8	RO	No	12	12	Number of subindexes
	1	UINT32	RW		UINT32	0	CANopen master script entry settings, see section 1.18
	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	
	9					0	
	10					0	
	11					0	
	12					0	

4.3.50. 5B00h-5B0Fh CANopen Master Script Data 0-15

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5B00	0	UINT8	RO	No	8	8	Number of subindexes
…	1		RW		UINT8	0	CANopen master script entry data, see section 1.18
5B0F	2					0	
	3					0	
	4					0	
	5					0	
	6					0	
	7					0	
	8					0	

4.3.51. 5B50h Set CANopen Baud Rate

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5B50	0	UINT8	RW	No	0-8	3	CANopen baud rate: 0=1000k, 1=800k, 2=500k, 3=250k, 4=125k, 5=100k, 6=50k, 7=20k, 8=10k

4.3.52. 5B51h Set CANopen Node ID

Index	Subindex	Data Type	Access	PDO Mapping	Value Range	Default Value	Description
5B51	0	UINT8	RW	No	1-127	10	CANopen Node ID to use

1.3. *J1939 data scaling*

J1939 data is stored locally in variables (of real32 type). When a J1939 input message is received, the data stored in the message is scaled into the local variable using the following equations:

$$local\ data = \Big(\big((MsgData * A) + B\big) * C\Big) + D$$

in which

$$A = \frac{Resolution}{CAN\ Maximum - CAN\ Minimum}$$

$$B = \frac{Offset - CAN\ Minimum}{CAN\ Maximum - CAN\ Minimum}$$

$$C = CAN\ Maximum - CAN\ Minimum$$

$$D = CAN\ Minimum$$

When building a J1939 output message, the output message functions use the local variables' (J1939/CANopen/Modbus) data. The data from the local storage is scaled into a J1939 output message signal using the following equations:

$$MsgData = \big((local\ data * G) * E\big) + F$$

in which

$$E = \frac{CAN\ Maximum - CAN\ Minimum}{Resolution}$$

$$F = \frac{CAN\ Minimum - Offset}{Resolution}$$

$$G = \frac{1}{CAN\ Maximum - CAN\ Minimum}$$

If no scaling is preferred, use the settings shown in Table 5. With these settings, the local data is copied directly into message data and vice versa.

Table 5: CAN Data scaling values for 1-to-1 scaling between local and message data

Parameter name	***Value***
Resolution	1
Offset	0
CAN Maximum	1
CAN Minimum	0

As an example of J1939-to-J1939 data routing and scaling, consider the following situation in which an incoming message having 4-byte data would be scaled and transmitted out as 2-byte data in another PGN.

J1939 input message:
ID: 18FF00F8, len: 8, data: 00 00 02 00 FF FF FF FF
CAN input signal parameters: signal type: *6* (4 bytes), resolution: *2^(-16)*, CAN min: *0.0*, CAN max: *65535.0*.
CAN output signal parameters: signal type *5* (2 bytes), resolution: *1.0*, CAN min: *0.0*, CAN max: *65535.0*.

The above would yield the following J1939 output message:
ID: 18FF0180, len: 8, data: 02 00 FF FF FF FF FF FF

Note: When setting very small values in EA (like the resolution 2^{-16} in the example above), the value shown on the PC screen stays at **0.000**. The value is still programmed to the protocol converter device, EA just shows the first three decimals on the PC screen.

1.4. *CANopen® data scaling*

The CANopen data can be scaled by setting a scaling factor J and a scaling offset K. The scaling coefficients are of type real32. The scaling itself is done using equation:

$$scaled\ data = (message\ data * J) + K$$

The message data in the equation above are the object dictionary entries 2000h-200Fh (default received CANopen data target objects) for received data and entries 2100h-210Fh (default transmitted CANopen data source objects) for transmitted data. Scaled data can be found from obejcts 3000h-300Fh (scaled received data) and from 3100h-310Fh (scaled data for transmitting). All these obejcts are mappable to be used in receive and transmit PDOs.

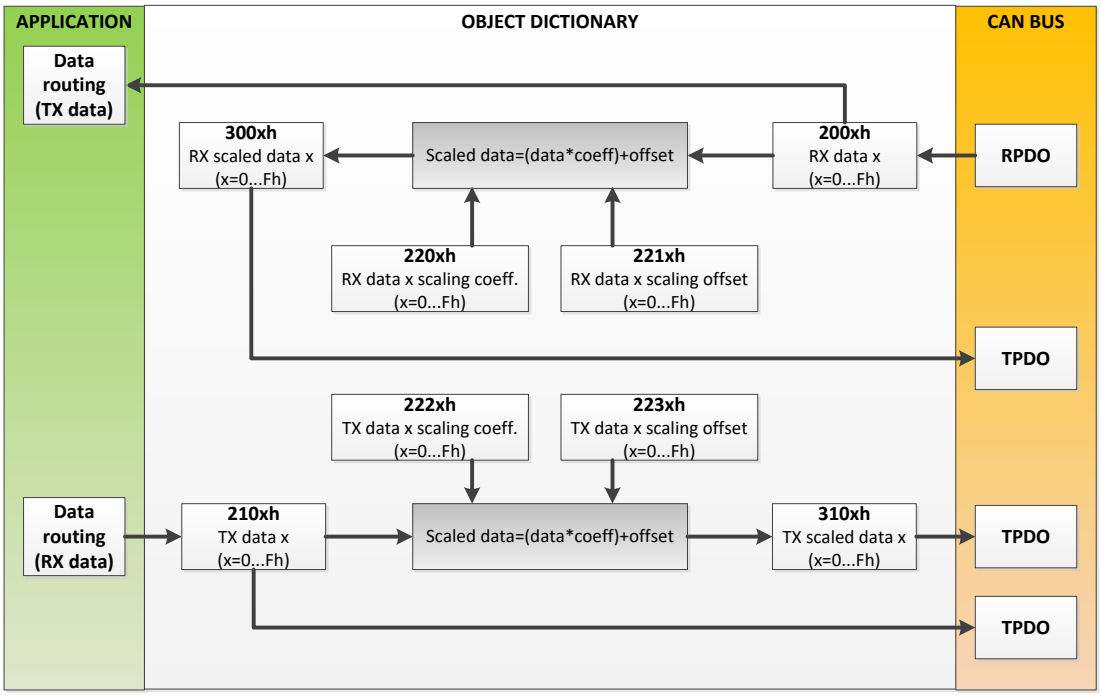


Figure 25: CANopen data scaling

Note, that CANopen RX scaled data can't be used directly in the data routing application, its main use is in transmit PDOs. This is because all the other interfaces will have data scaling possibility for their output data.

1.4.1. CANopen® data types

CANopen objects 2000h-200Fh (see 4.3.1) and 2100h-210Fh (see 4.3.2) contain altogether six sub-indexes, out of which sub-indexes 1-4 are for data and sub-indexes 5 and 6 are for special data type (listed in Table 13) and for bit mask configuration.

The objects 2000h-200Fh and 2100h-210Fh are used in PDO mappings by default. The received and transmitted data is type casted to the specified data type before used in the Protocol Converter data routing and scaling.

1.4.2. CANopen® bit signals

The AX140200&AX140321 support four objects for J1939 bit signals -> CANopen bit signals routing. The objects 4120h-4123h (see 4.3.41) define the data sources for bit signal data objects 2180h-2183h (see 4.3.3).

The data objects 2180h-2183h are CANopen VARs of UNSIGNED8 type. The variables in these objects can be set bitwise on/off using the data sources specified in 4120h-4123h (sub-index 1 = bit 0, sub-index 2 = bit 1, etc.)

1.5. *Modbus data scaling*

The Modbus server data is stored in a local structure containing all coil, input and different registers' latest values. This structure can be updated by a remote Modbus client device accessing Protocol converter or locally by writing from a received CAN message data (see section 1.7) or from a Modbus Master message data (see section 1.8).

The data read from the Modbus server/client data structures is scaled as follows when placed into the Protocol Converter's local variables:

$$local\ data = (Modbus\ data * Modbus\ resolution) + Modbus\ offset$$

After scaling, the data will be limited between the specified Minimum and Maximum values.

$$Data_{Modbus} = \max[local\ data, MIN_{Modbus}]$$
$$Data_{Modbus} = \min[local\ data, MAX_{Modbus}]$$

The data written into Modbus server data structures and into Modbus client messages is scaled as follows:

$$Modbus\ data = \frac{local\ data - Modbus\ offset}{Modbus\ resolution}$$

After scaling, the data will be limited between the specified Minimum and Maximum values.

$$Data_{Modbus} = \max[Modbus\ data, MIN_{Modbus}]$$
$$Data_{Modbus} = \min[Modbus\ data, MAX_{Modbus}]$$

1.6. *J1587 Data scaling (AX140400/AX140322 only)*

J1587 data is scaled using the Resolution and Offset parameters. The Minimum and Maximum setpoints define the minimum and maximum limits for the scaled data. In case the result of data scaling (using resolution and offset parameters) is out of range, the value will saturate either to Minimum or Maximum setpoint value.

$$scaledValue_{J1587} = \frac{fValue}{Resolution_{J1587}} + Offset_{J1587}$$

in which fValue is the internal value of the data (received from another J1587 message or from J1939 bus).

The data that will be sent to J1587 bus is compared with the Minimum and Maximum values and saturate in case it is out of configured range.

$$Data_{J1587} = \max\left[scaledValue_{J1587}, MIN_{J1587}\right]$$
$$Data_{J1587} = \min\left[scaledValue_{J1587}, MAX_{J1587}\right]$$

1.7. *J1939 Output Signal Source*

A J1939 output message signal can have one of the following data sources; Not connected (signal disabled), Modbus Local and Remote Data, J1939 Rx Data and CANopen Rx Data (in devices supporting CANopen®, such as the AX140200/AX140321).

Table 6: J1939 output messages' signal source options

Signal source	***Explanation***
Not connected	No data source, this signal won't be included into the output message
Modbus Local Data	Use Modbus local data specified by the Modbus –parameters for the CAN output signal (Modbus→J1939 routing)
Modbus Remote Data	Use Modbus remote data (requested from a remote Modbus node). With this option, *Modbus Address* parameter is used for specifying the Modbus Master message definition (1-24) to use for requesting the data (Modbus→J1939 routing)
Latest (local) updated Modbus address	Use latest updated (write command from an external Modbus node) address in local Modbus implementation
Latest (local) updated Modbus data	Use latest updated (write command from an external Modbus node) data in local Modbus implementation
CAN Received Signal #1	Use CAN Input message #1 data (J1939→J1939 routing)
CAN Received Signal #2	Use CAN Input message #2 data (J1939→J1939 routing)
CAN Received Signal #3	Use CAN Input message #3 data (J1939→J1939 routing)
CAN Received Signal #4	Use CAN Input message #4 data (J1939→J1939 routing)
CAN Received Signal #5	Use CAN Input message #5 data (J1939→J1939 routing)
CAN Received Signal #6	Use CAN Input message #6 data (J1939→J1939 routing)
CAN Received Signal #7	Use CAN Input message #7 data (J1939→J1939 routing)
CAN Received Signal #8	Use CAN Input message #8 data (J1939→J1939 routing)
CAN Received Signal #9	Use CAN Input message #9 data (J1939→J1939 routing)
CAN Received Signal #10	Use CAN Input message #10 data (J1939→J1939 routing)
. . .	
CAN Received Signal #56 (#32 in CANopen® devices)	Use CAN Input message #56 data (J1939→J1939 routing) (message #32 data in CANopen® devices)
--OR--	
CANopen RPDO	Use CANopen receive PDO #n data (CANopen®→J1939 routing, use "CANopen message number" and "CANopen subindex number" to configure the actual data to use).

Note, that in case of Modbus→J1939 data routing, the message data is first scaled using corresponding Modbus scaling values and then again using CAN output signal scaling values.

If ***Latest local updated address*** or ***Latest local updated data*** is used, the latest address and/or data from Modbus interface shall be included into outgoing CAN message.

For example, Modbus "Write Holding Register" command (15h 06h 00h 01h 11h 23h 96h 97h) would yield latest updated address '1' and latest updated data '1123h'.

In case of J1939→J1939 data routing, the received data is first scaled using the CAN input message data scaling values and then using the CAN output signal scaling values before it is sent out as the CAN message.

In case of CANopen→J1939 data routing, the additional coefficients "Input #n CANopen Message Number" and "Input #n CANopen Message Subindex" (in which n=1…5) define the CANopen RPDO number and it's data source variable's subindex to use when adding data into the outgoing J1939 message.

1.8. *CANopen TX Messages' In Source*

The input source for the CANopen PDOs (for data that is placed into objects 2100h-210Fh by the protocol converter's background task) can be defined using objects 4020h-402Fh. The available data source options are shown in the table below.

Table 7: CANopen TX messages' input signal source options

Value	***Signal source***	***Explanation***
0	Default CANopen data (objects 3000h-300Fh)	Default CANopen data source, no routing
1	Modbus Local Data	Use Modbus local data specified by the Modbus –parameters for the CAN output signal (Modbus→CANopen routing)
2	Modbus Remote Data	Use Modbus remote data (requested from a remote Modbus node). With this option, *Modbus Address* parameter is used for specifying the Modbus Master message definition (1-24) to use for requesting the data (Modbus→CANopen routing) **Note that you also need to define the Modbus message data type (object 24x0h) for correct scaling!**
3	*No source*	
4	*No source*	
5	CAN Received Signal #1	Use CAN Input message #1 data (J1939→CANopen routing)
6	CAN Received Signal #2	Use CAN Input message #2 data (J1939→CANopen routing)
7	CAN Received Signal #3	Use CAN Input message #3 data (J1939→CANopen routing)
8	CAN Received Signal #4	Use CAN Input message #4 data (J1939→CANopen routing)
9	CAN Received Signal #5	Use CAN Input message #5 data (J1939→CANopen routing)
10	CAN Received Signal #6	Use CAN Input message #6 data (J1939→CANopen routing)
11	CAN Received Signal #7	Use CAN Input message #7 data (J1939→CANopen routing)
12	CAN Received Signal #8	Use CAN Input message #8 data (J1939→CANopen routing)
13	CAN Received Signal #9	Use CAN Input message #9 data (J1939→CANopen routing)
14	CAN Received Signal #10	Use CAN Input message #10 data (J1939→CANopen routing)
…	. . .	
36	CAN Received Signal #32	Use CAN Input message #32 data (J1939→CANopen routing)

In case Modbus slave database is used as a source (or data target in PDO reception), the table below lists the Modbus Types that can be programmed into objects 24x0h (in which x=0…15, the number of CANopen PDO configuration).

Table 8: Modbus (Slave) Types

Objects 23x0h, 24x0h – Modbus Types	
Value	*Type*
0	Coil (1bit)
1	Input (1bit)
2	Holding register (16bits)
3	Input register (16bits)
4	Holding register (32bits)
5	Input register (32bits)
6	Holding register (FLOAT, 32bits)
7	Input register (FLOAT, 32bits)

1.9. *CANopen TX Messages' Additional Triggering*

The CANopen PDOs are sent out by the programmed transmission timer interval, as specified in PDO parameter objects 1800h-180Fh. In case additional on reception of different messages type of triggering is needed, the objects 4000h and 4001h can be used for defining this functionality.

Table 9: CANopen TX message triggers

Object 4000h – CANopen TX Message Triggers	
Value	*Action*
0	No action (PDO timers only)
1	CANopen PDO reception
2	J1939 PGN reception
3	Modbus Message reception

Table 10: CANopen TX message parameters

Object 4001h – CANopen TX Message Parameters	
4000h value	*Action*
0	No action
1	Received CANopen PDO number (1…16)
2	Received J1939 PGN number (1…16)
3	Received Modbus Function Code to match

1.9.1. CANopen TX message configuration example

For sending out a CANopen TPDO upon reception of a Modbus write command and use data from a specified Modbus (local) holding register, the following CANopen configuration steps needs to be taken:

1. CANopen TX PDO #1 triggering set to reception of a Modbus message (**4000h**, subindex 1 set to '3', see also Table 9)
2. CANopen TX PDO #1 triggering parameter set to write holding reg command (**4001h**, subindex 1 set to '6', write holding register command).
3. CANopen TX PDO #1 data source to local modbus data (**4020h** subindex 1 set to '1', see also Table 7)
4. CANopen TX PDO #1 Modbus type to Holding Register (**2400h**, subindex 1 set to '2', see also Table 8)
5. CANopen TX PDO #1 Modbus address to register #1 (**2401h**, subindex 1 set to '1').
6. CANopen TX PDO #1 Modbus maximum to 65535.0 (**2403h**, subindex 1 set to ´0x477FFF00')
7. CANopen TX PDO #1 Modbus resolution to 1.0 (**2405h**, subindex 1 set to ´0x3F800000')
8. Transmit PDO tx interval to 0ms (**1800h**, subindex 5 set to '0')

1.10. *CAN Input Data Destination*

A J1939 input message can have one of the following data destinations specified; Default CAN (stored only into a local variable) or Modbus Local Data Destination. Note, that independent of the selected input data destination, received data will always be stored into the local CAN variables for other routing functions to use.

Table 11: J1939 input data destination

Signal destination	***Explanation***
Default CAN	Store data only into a local variable.
Modbus Local Destination	Store data into Modbus local data structure with type, address and scaling specified by the Modbus –parameters for the CAN input signal (J1939→Modbus routing). Data is stored also into a local variable for other routing functions to use.

1.11. *CANopen RX Messages' Data Destination*

A CANopen receive PDO data can have one of the following data destinations specified; Default CAN (stored only into a local variable, 2000h-200Fh) and Modbus Local Data Destination. Note, that independent of the selected input data destination, received data will always be stored into the local CAN variables for other routing functions to use.

In case local Modbus slave database is used as a destination, please see Table 8 for Modbus Types that can be programmed into objects 23x0h (in which x=0…15, the number of CANopen PDO configuration).

Also, the corresponding data scaling objects need to be properly set. Please note, that the values of the critical **Maximum** (23x3h, see also 4.3.10) and **Resolution** (23x5h, see also 4.3.12) objects are 0.0 by default. Non-zero values need to be configured to proper subindexes of objects 23x3h and 23x5h. These objects need to be programmed with floating point values.

Table 12: CANopen RX messages' out destination configurations

Value	***Signal destination***	***Explanation***
0	Default CAN	Store data only into a local variable (objects 2000h-200Fh)
1	Modbus Local Destination	Store data into Modbus local data structure with type, address and scaling specified by the Modbus –parameters for the CAN input signal (CANopen→Modbus routing). Data is stored also into a local variable for other routing functions to use.
2	Direct RS485 Forwarding	The received data is forwarded as is into the RS485 interface, in case the RS485 interface is set to operate in Raw Data mode. See also object 0x5557.

Table 13: CANopen RX and TX messages' data types

Value	***Data type***
0	NULL
1	Boolean
2	Integer8
3	Integer16
4	Integer32
5	Unsigned8
6	Unsigned16
7	Unsigned32
8	Real32
9	Bitfield

1.12. *J1939 <-> Modbus data transfer in general*

The AX140100, AX140200, AX140320 and AX140321 support data transfer between J1939 and Modbus in two ways. The Protocol Converter can work as a Modbus master that actively reads from and writes data to other Modbus devices. This configuration supports up to 24 Modbus parameters (see also section 1.13).

The other method for transferring data is that the Protocol Converter works as a Modbus slave, and just makes the received J1939 data available in its Modbus registers that can be then read by other Modbus devices. This approach supports 56 Modbus registers in AX140100 and AX140320, and 32 in AX140200 and AX140321. There is also the special CAN to Modbus slave version AX140100-100 that supports 120 Modbus registers. Practically the number of signals that can be routed from CAN into Modbus is limited by these numbers, because the Modbus slave implementation supports up to 1024 Holding registers into which the data can be forwarded.

1.13. *Modbus Master Messages*

Protocol Converter can be set up for having Modbus client functionality in addition to the default Modbus server implementation.

Further, these Modbus Master message definitions can be used for setting up Modbus→Modbus data routing. Note, that the Protocol Converter has only one Modbus interface, so this Modbus data routing is between the local Modbus server and the Modbus client implementation sharing the same Modbus interface.

Table 14: Modbus commands for master messages

Modbus command type	***Explanation***
No operation	Message disabled
Read remote coils	Read coil status information from a remote node
Read remote input bits	Read input bit status information from a remote node
Read remote holding registers	Read holding register value from a remote node
Read remote input registers	Read input register value from a remote node
Read remote UINT32 holding registers	Read 32bit wide unsigned integer value from a holding register of a remote node*
Read remote UINT32 input registers	Read 32bit wide unsigned integer value from an input register of a remote node*
Read remote FLOAT32 holding registers	Read 32bit wide IEEE floating point value from a holding register of a remote node*
Read remote FLOAT32 input registers	Read 32bit wide IEEE floating point value from an input register of a remote node*
Write remote coil	Write single coil status into a remote node
Write remote holding register	Write single holding register value into a remote node
Write remote UINT32 holding registers	Write 32bit wide unsigned integer value to a holding register of a remote node*
Write remote FLOAT32 holding registers	Write 32bit wide unsigned integer value to an input register of a remote node*

* These commands access two consequent 16-bit wide holding/input registers.

Note that the **Modbus Length** (in both remote and local definitions for reading and writing) **is '1' by default** (and it is not user configurable). **The only exception is the custom UINT32 and FLOAT32 messages that access two consequent registers.** In practice this limits the Modbus data (read or write) access using a single master message definition to one coil/register. To access multiple Modbus addresses, multiple Modbus messages needs to be defined.

In case **Read Remote UINT32 Holding / Input Registers** command is defined with **negative Modbus Minimum** value, the read data is handled as a **signed number**.

1.14. *J1939 -> J1939 Low Level Filtering and Routing (AX140100, AX140320, AX140400 and AX140322 only)*

AX140100, AX140320, AX140400 and AX140322 support direct routing and filtering of CAN messages between the CAN interfaces. The routing & filtering function contains altogether 40 filters (20 filters for CAN1->CAN2 filtering and 20 filters for CAN2->CAN1 filtering).

The router has three main configuration options, **Route all messages**, **Filter messages** and **Use local claimed address in outgoing messages**. The local address is the claimed address of CAN1 or CAN2 interface, depending on which one is the interface sending the routed messages out.

Route all messages specify whether all received messages should be routed from one CAN interface into another by default. If this is set to *yes*, the AX140100/AX140320 & AX140400/AX140322 will forward all received CAN frames from one CAN interface into the other. Note, that the low-level routing feature handles the received frames on a lower level than the rest of the data routing between interfaces, thus all messages get routed independent of whether they are configured as input messages or not.

Filter messages specify if the 20 filter definitions are enabled or not. If **Route all messages** is set to *no*, the user configurable filters will define which of the received messages get routed. On the other hand, if **Route all messages** is set to *yes*, the 20 configurable filters will define the messages that are not allowed to pass.

The use local CAN claimed address option will force the source address of the outgoing messages into the local claimed address of the sending interface. Otherwise, the CAN frame's ID field is left as is.

The filter consists of two parameters, namely the filter option and the filter parameter. The filter option specifies the four possible filtering methods, *Match PGN*, *Match ID*, *Match Priority, Custom Match*, *Priority OR*, *Route all ExtdID* and *Route All StdID* (see also section 1.9, Table 2). When defining the filtering, configure filters starting from the first available filter.

NOTE: In case a Custom ID filter is used, and the ID specified is equal to or below 0x7FF, the ID is assumed to be received in an 11-bit ID CAN frame. For proper initialization of the CAN filters, a power cycle is needed after configuring Custom IDs with values equal to or below 0x7FF (2047 dec).

NOTE: First filter having *Not configured* as an option will break the filter chain handling procedure. All filters defined after the first *Not configured* filter will not be handled, i.e. incoming CAN frames won't be compared against these definitions.

1.14.1. PGN filtering example

Consider the following setup:

Route all messages	**No**
Filter messages	**Yes**
Filter option	**Match PGN**
Filter parameter	**0xFF01**

Received frames:

```
Id: 18FF0080 len: 8  data: 00 00 11 11 00 00 11 11   → DROPPED
Id: 18FF0180 len: 8  data: 00 00 11 11 00 00 11 11   → PASS
```

1.14.2. ID filtering example

Consider the following setup:

Route all messages	**No**
Filter messages	**Yes**
Filter option	**Match ID**
Filter parameter	**0x80**

Received frames:

```
Id: 18FF0083 len: 8  data: 00 00 11 11 00 00 11 11   → DROPPED
Id: 18FF0080 len: 8  data: 00 00 11 11 00 00 11 11   → PASS
```

1.14.3. Priority filtering example

Consider the following setup:

Route all messages	**No**
Filter messages	**Yes**
Filter option	**Match Priority**
Filter parameter	**6**

Received frames:

```
Id: 0CFF0080 len: 8  data: 00 00 11 11 00 00 11 11   → DROPPED
Id: 18FF0080 len: 8  data: 00 00 11 11 00 00 11 11   → PASS
```

1.14.4. Custom filtering example

Consider the following setup:

Route all messages	**No**
Filter messages	**Yes**
Filter option	**Custom Match**
Filter parameter	**0x18000180**

Received frames:

```
Id: 18FF0080 len: 8  data: 00 00 11 11 00 00 11 11    → DROPPED
Id: 18000180 len: 8  data: 00 00 11 11 00 00 11 11    → PASS
```

1.15. *Build-in J1939-J1587 Data Mappings (AX140400/AX140322 only)*

The built-in, fixed data mappings as described in Table 15: Built-in data mappings between J1939 and J1587 are available in AX140400/AX140322. To enable these mappings, the corresponding setpoint needs to be set to ‘true’ in “Build-in J1939-J1587 Data Mappings” setpoint group. Please see section 1.16 for setpoint details.

Please note, that all built-in J1939 messages expect source address 0x00 in the messages for data mapping functionality.

It is possible to override the J1939 Source Address and J1587 MID for the forwarded messages with a user specified value.

Table 15: Built-in data mappings between J1939 and J1587

Name	PGN	SPN	Width	Pos.	Res.	Offset	Min	Max	PID	Res.	Offset	Min	Max
Throttle Position	0xFEF2	51	8bits	6	0.4	0.0	0.0	100.0	51	0.4	0.0	0.0	102.0
Engine Oil Pressure	0xFEEF	100	8bits	3	4.0	0.0	0.0	1000.0	100	3.45	0.0	0.0	879.0
Engine Coolant Pressure	0xFEEF	109	8bits	6	2.0	0.0	0.0	500.0	109	0.862	0.0	0.0	219.8
Engine Boost Pressure	0xFEF6	102	8bits	1	2.0	0.0	0.0	500.0	102	0.862	0.0	0.0	219.8
*Intake Manifold Temperature**	0xFEF6	105	8bits	2	1.0	-40.0	-40.0	210.0	105	1.0	0.0	0.0	255.0
Barometric Pressure	0xFEF5	108	8bits	0	0.5	0.0	0.0	125.0	108	0.431	0.0	0.0	109.9
*Engine Coolant Temperature**	0xFEEE	110	8bits	0	1.0	-40.0	-40.0	210.0	110	1.0	0.0	0.0	255.0
*Engine Oil Temperature**	0xFEEE	175	16bits	2	0.03125	-273.0	-273.0	1734.96875	175	0.25	0.0	-8192.0	8191.75
Injector Timing Rail Pressure	0xFEDB	156	16bits	4	1.0/256.0	0.0	0.0	250.996	156	0.689	0.0	0.0	45153.6
Injector Metering Rail Pressure	0xFEDB	157	16bits	2	1.0/256.0	0.0	0.0	250.996	157	0.689	0.0	0.0	45153.6
Battery Voltage	0xFEF7	168	16bits	4	0.05	0.0	0.0	3212.75	168	0.05	0.0	0.0	3276.75
Fuel Rate	0xFEF2	183	16bits	0	0.05	0.0	0.0	3212.75	183	0.000016428	0.0	0.0	1.07665
Engine Speed	0xF004	190	16bits	3	0.125	0.0	0.0	8031.875	190	0.25	0.0	0.0	16383.75
Percent Engine Torque	0xF004	513	8bits	2	1.0	-125.0	-125.0	125.0	513	0.25	0.0	0.0	16383.75
Total Engine Hours	0xFEE5	247	32bits	0	0.05	0.0	0.0	210554060.75	247	0.05	0.0	0.0	214748364.8
Total Engine Revolutions	0xFEE5	249	32bits	4	0.05	0.0	0.0	210554060.75	249	1000	0.0	0.0	$4.29*10^{12}$
Time, Seconds	0xFEE6	959	8bits	0	0.25	0.0	0.0	62.5	251	0.25	0.0	0.0	62.5
Time, Minutes	0xFEE6	960	8bits	1	1.0	0.0	0.0	250.0	251	1.0	0.0	0.0	250.0
Time, Hours	0xFEE6	961	8bits	2	1.0	0.0	0.0	250.0	251	1.0	0.0	0.0	250.0
Date, Day	0xFEE6	962	8bits	4	0.25	0.0	0.0	62.5	252	0.25	0.0	0.0	62.5
Date, Month	0xFEE6	963	8bits	3	1.0	0.0	0.0	250.0	252	1.0	0.0	0.0	250.0
Date, Year	0xFEE6	964	8bits	5	1.0	0.0	0.0	250.0	252	1.0	0.0	0.0	250.0

**Includes °C↔°F conversion*

1.16. *Direct Diagnostics Routing J1939-J1587 (AX140400/AX140322 only)*

The AX140400/AX140322 can route diagnostics messages between J1939 and J1587. When forwarding diagnostics messages from J1939 to J1587, the forward function uses J1939 SPN as J1587 diagnostic code (PID) in diagnostics message (PID 194). To enable the diagnostic routing functionality, the corresponding setpoint needs to be set to ‘true’ in “Direct Diagnostics Routing J1939-J1587” setpoint group. Please see section 1.17 for setpoint details.

In the other direction, the J1587 diagnostic code PID is used as J1939 SPN.

The FMI and OC values are forwarded as received.

It is possible to override the J1939 Source Address and J1587 MID for the forwarded diagnostics messages with a user specified value.

1.17. *CANopen EMCY forwarding to J1939 (AX140200/AX140321 only)*

AX140200/AX140321 supports the sending and receiving of CANopen EMCY messages. The received EMCY messages can be used for triggering the sending of J1939 DM1 messages. There are 16 EMCY transmit and 16 EMCY receive configurations that are currently supported.

The EMCY transmission can be triggered by receiving a specified J1939 DM1 message. Also, Modbus network errors (data read error, data write error, communication error) can be forwarded to CANopen bus as EMCY messages.

The objects 4100h, 4101h, 4102h and 4103h can be used for configuring the outgoing EMCY messages. The subindex used for configuration defines the EMCY index, the configuration must use the same subindex in all four objects.

For example, to send the following EMCY on reception of EA configured DM1 reception:

ID: 0x089, data: 0x04 0x10, 0x01, 0x00, 0x22, 0x11, 0x00, 0x00

The configuration needs to be set as follows (subindex 1 used for all four objects). Object 4100h: 0x1 (this defines that the transmit trigger is the first J1939 DM1 definition), object 4101h: 0x1004, object 4102h: 0x1122, object 4103h: 0x9. This configuration would then send EMCY #1 on reception of J1939 SPN from the specified address (J1939 diagnostics to monitor, please see 1.10 for more info).

The objects 2A00h, 2A01h and 2A02h can be used for configuring the receive filter for EMCY messages generated by other CANopen nodes in the network. All three filtering criteria need to match before an EMCY message is flagged as received. The subindex used for configuration defines the EMCY index, that can be then used in the EA based J1939 configuration.

For example, to receive the following EMCY:

ID: 0x083, data: 0x04 0x10, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00

The configuration needs to be set as follows (subindex 1 used for all three objects). Object 2A00h: 0x1004, object 2A01h: 0x1, object 2A02h: 0x3. This configuration would then flag EMCY #1 as active on reception of the above EMCY.

1.18. *CANopen Master Scripts (AX140200/AX140321 only)*

The AX140200/AX140321 supports simple CANopen master scripts, which can be used for reading and writing SDO data to remote CANopen nodes.

Objects 5A00h…5A0Fh specify the commands and 5B00h…5B0Fh store the initial data for the commands. Once configured, the AX140200/AX140321 will start to execute the script as soon as the mode of operation is changed to OPERATIONAL.

Table 16: CANopen Master Script Configuration and allowed value ranges

Subindex	Name	Value range	Details
0	Number of subindexes in this object	Read only	CANopen specific read-only value (default: 12).
1	Message ID	0x600 … 0x67F	The message ID to use when sending messages.
2	Message length	1 – 8	The message length in bytes.
3	Delay	0 … 60000	Delay in milliseconds before sending the message.
4	Status	0, 1	Configuration entry status, valid values: 0 – Entry not in use 1 – Entry configured and in use
5	Data mapping	0, 1, 2	Data mapping to use with this script entry: 0 – No data mapping 1 – Only outgoing data 2 – Incoming data expected The outgoing and incoming data is read and written to Master Tx (2900h, see 0) and Rx (2800h, see 4.3.21) data objects or from Modbus slave interface. The script entry number determines the subindex used. For example, script entry 0x5A00 uses data from subindex 1 of 2900h.
6	Data mapping start byte	1 – 8	Message data start position. In case of a SDO messaging, this should be set to 4.
7	Data mapping length	1 – 8	The length of the mapped data in bytes. Note, that ‘Data mapping start byte’ + ‘Data mapping length’ can’t exceed ‘8’ (which is the maximum data length for a CAN frame).
8	Script entry type	1, 2, 4	Script entry configuration type: 0 – Not configured 1 – SDO 2 – PDO 3 – Proprietary
9	Transmit count	UINT32	The number of TX/RX rounds for this script entry. In case continuous transmission/reception is preferred, write -1 to this entry (0xFFFFFFFF as uint32).
10	Transmit reload value	UINT32	If a specific number of TX/RX rounds is specified, this value defines the Transmit count reload value. The Transmit counter is reloaded using this value on every mode transition to OPERATIONAL. In case continuous transmission/reception is preferred, write -1 to this entry (0xFFFFFFFF as uint32).

11	Data destination / source	UINT32	Data destination/source, flags and addresses: 0x80000000=Data destination/source: UINT32 Holding register, 0x40000000=Data destination/source: UINT32 Input register Mask: 0x000FFFFF=Data destination address mask
12	RX autoreset in milliseconds	0 … 60000	In case a SDO read is not answered before autoreset time elapses, the rx data is set to 0xFFFFFFFF.

1.18.1. CANopen Master Script example for SDO write

This example shows how to configure a Master Script entry for SDO write procedure. The data written to the remote node is read from corresponding CAN Master Tx object. In this example, the configuration is written to object 5A00h and the data is read from subindex 1 of 2900h.

The message to send will have an ID 0x60C (subindex 1) with a data length of 8 (subindex 2). There will be a 1000ms delay before sending out this message (subindex 3). The entry is configured to be in use (subindex 4) and have only outgoing data (subindex 5). The data mapping starts from byte 4 and is 4 bytes long (subindexes 6 and 7, corresponds to a SDO write message). Subindex 8 specifies this to be a SDO message and subindexes 9 and 10 are set to -1 for continuous transmission (at 1000ms intervals, because the delay before sending is set to 1000ms). Subindex 11 specifies the data source. If left to default 0x0, the data is read from object 2900h. The data can be also read from local Modbus slave interface, please see Table 16.

sub	size	Hex	long unsigned	long signed	hi uns	low uns	hi sig	low sig	FLOAT	Visible String$
0:	1	0000000C	12	12	0	12	0	12		.
1:	4	0000060C	1548	1548	0	1548	0	1548	0.000000	....
2:	4	00000008	8	8	0	8	0	8	0.000000	....
3:	4	000003E8	1000	1000	0	1000	0	1000	0.000000	....
4:	4	00000001	1	1	0	1	0	1	0.000000	....
5:	4	00000001	1	1	0	1	0	1	0.000000	....
6:	4	00000004	4	4	0	4	0	4	0.000000	....
7:	4	00000004	4	4	0	4	0	4	0.000000	....
8:	4	00000001	1	1	0	1	0	1	0.000000	....
9:	4	FFFFFFFF	4294967295	-1	65535	65535	-1	-1	-1.#QNAN0	....
10:	4	FFFFFFFF	4294967295	-1	65535	65535	-1	-1	-1.#QNAN0	....
11:	4	00000000	0	0	0	0	0	0	0.000000	....
12:	4	00000000	0	0	0	0	0	0	0.000000	....

Figure 26: Example of CANopen Master script configuration for SDO write

The above example specifies that the mapped data for the message starts from byte 4. We will still need to specify the first four static data bytes. This can be done with object 5B00h (for script entry in 5A00h). The setup shown in Figure 27 configures the SDO write to subindex 1 of object 2101h. In this setup, the subindexes 5 to 8 contain only FFhs. These subindexes are ‘don’t care’ values because the Master Script engine will replace this data with the data read from object 2900h or from Modbus slave interface.

sub	size	Hex	long unsigned	long signed	hi uns	low uns	hi sig	low sig	FLOAT	Visible String$
0:	1	00000008	8	8	0	8	0	8		.
1:	1	0000002B	43	43	0	43	0	43		+
2:	1	00000001	1	1	0	1	0	1		.
3:	1	00000021	33	33	0	33	0	33		!
4:	1	00000001	1	1	0	1	0	1		.
5:	1	000000FF	255	255	0	255	0	255		.
6:	1	000000FF	255	255	0	255	0	255		.
7:	1	000000FF	255	255	0	255	0	255		.
8:	1	000000FF	255	255	0	255	0	255		.

Figure 27: Example of static data configuration for the Master script for SDO write

The above configuration will send following messages to CAN #2 interface of AX140200/AX140321 at 1000ms intervals:

ID: 0x60C, len: 8, data: 0x2B 0x01 0x21 0x01 <d1> <d2> <d3> <d3>

in which <d1>…<d3> are read from subindex 1 of 2900h or from Modbus Holding/Input registers, depending on configuration.

1.18.2. CANopen Master Script example for SDO read

This example shows how to configure a Master Script entry for SDO read procedure. The data read from the remote node is written into the corresponding CAN Master Rx object. In this example, the configuration is written to object 5A01h and the received data is written to subindex 2 of 2800h. Please note that this configuration is written in addition to the SDO write configuration shown in 0.

The message to send for requesting the SDO read reply will have an ID 0x60C with a data length of 8 defined in subindex 2. Although subindex 3 is set to 1000, this message will be sent at 2000ms intervals. The delay of 1000ms is the delay between this message and the first script entry (which was configured in 0). The entry is configured to be in use (subindex 4 set to ‘1’) and has incoming data (subindex 5). The data mapping starts from byte 4 and is 4 bytes long (subindexes 6 and 7, corresponds to a SDO read message). Subindex 8 specifies this to be a SDO message and subindexes 9 and 10 are set to -1 for continuous transmission (at 2000ms intervals, because the delay before sending the first script entry is 1000ms). Subindex 11 specifies the received data destination, in this case the received data is forwarded to local Modbus slave interface, to UINT32 Holding Register @ address 10. Subindex 12 specifies the timeout in milliseconds. If no response is received before the timeout, the received data is set to 0xFFFFFFFF. Please also see Table 16.

sub	size	Hex	long unsigned	long signed	hi uns	low uns	hi sig	low sig	FLOAT	Visible String$
0:	1	0000000C	12	12	0	12	0	12		.
1:	4	0000060C	1548	1548	0	1548	0	1548	0.000000	....
2:	4	00000008	8	8	0	8	0	8	0.000000	....
3:	4	000003E8	1000	1000	0	1000	0	1000	0.000000	....
4:	4	00000001	1	1	0	1	0	1	0.000000	....
5:	4	00000002	2	2	0	2	0	2	0.000000	....
6:	4	00000004	4	4	0	4	0	4	0.000000	....
7:	4	00000004	4	4	0	4	0	4	0.000000	....
8:	4	00000001	1	1	0	1	0	1	0.000000	....
9:	4	FFFFFFFF	4294967295	-1	65535	65535	-1	-1	-1.#QNAN0	....
10:	4	FFFFFFFF	4294967295	-1	65535	65535	-1	-1	-1.#QNAN0	....
11:	4	8000000A	2147483658	-2147483638	32768	10	-32768	10	-0.000000	....
12:	4	000007D0	2000	2000	0	2000	0	2000	0.000000	....

Figure 28: Example of CANopen Master script configuration for SDO read

The above example specifies that the mapped data for the message starts from byte 4. We will still need to specify the first four static data bytes. This can be done with object 5B01h (for script entry in 5A01h). The setup shown in Figure 29 configures the SDO read from subindex 2 of object 1018h.

sub	size	Hex	long unsigned	long signed	hi uns	low uns	hi sig	low sig	FLOAT	Visible String$
0:	1	00000008	8	8	0	8	0	8		.
1:	1	00000040	64	64	0	64	0	64		@
2:	1	00000018	24	24	0	24	0	24		.
3:	1	00000010	16	16	0	16	0	16		.
4:	1	00000002	2	2	0	2	0	2		.
5:	1	000000FF	255	255	0	255	0	255		.
6:	1	000000FF	255	255	0	255	0	255		.
7:	1	000000FF	255	255	0	255	0	255		.
8:	1	000000FF	255	255	0	255	0	255		.

Figure 29: Example of static data configuration for the Master script for SDO read

The above configuration will send following messages to CAN #2 interface of AX140200/AX140321 at 2000ms intervals:

ID: 0x60C, len: 8, data: 0x40 0x18 0x10 0x02 0xFF 0xFF 0xFF 0xFF

The replies from the remote node with the SDO read data are expected to be:

ID: 0x58C, len: 8, data: <d.c.> 0x18 0x10 0x02 <d1> <d2> <d3> <d3>

from which <d.c.> is a ‘don’t care’ and <d1>…<d3> are stored to subindex 2 of 2800h.

1.19. *CAN <-> RS485 (RS422) direct data routing mode*

The AX140100, AX140200, AX140320 and AX140321 firmware supports low level CAN <-> RS485 direct data forwarding. The **RS485 mode** setpoint (see section 1.3) controls whether this type of data forwarding is active.

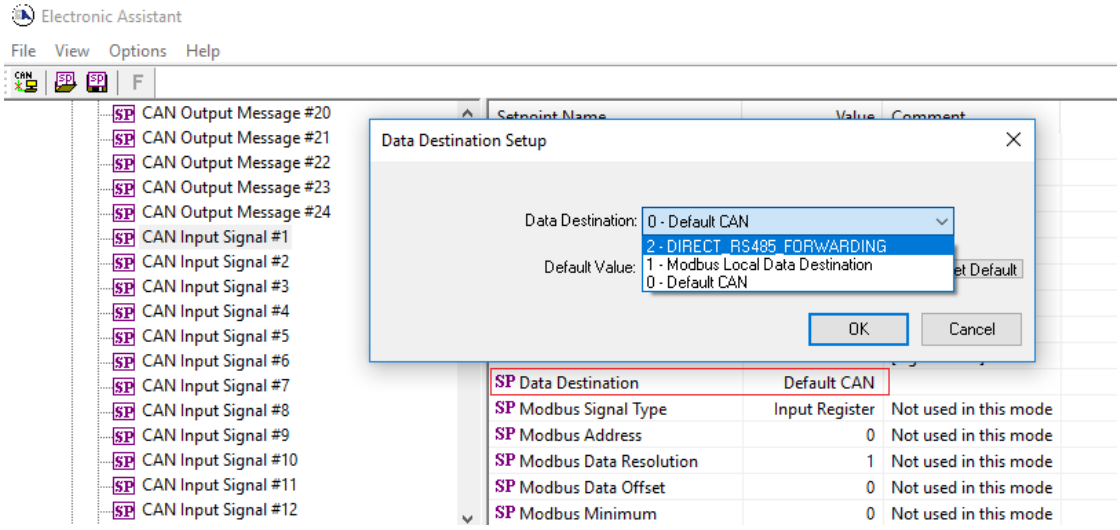


Figure 30: CAN Input messages' configuration for direct RS485 data forwarding

To forward received CAN data into RS485 as is, the CAN Input Signals' *Data Destination* setpoint needs to be set as *2 – DIRECT RS485 FORWARDING*.

The other way around, the configuration of RS485 data forwarding into CAN is done using the *Start char #1*, *Start char #2*, *End char #1* and *End char #2* setpoints. Also, *Forward raw RS485 to CAN* needs to be set to "*Yes"*.

The start and end characters define the one or two char sequences for parsing the RS485 message stream and generating the CAN messages from that. For not using a start or end characters, those setpoints need to be set to '*-1*'. In case no start and end characters are configured, all received RS485 traffic will be forwarded.

The *CAN ID for recvd RS485 msgs* and *Use 11bit ID* setpoints define the CAN ID to use when forwarding the received RS485 / RS422 data.

Table 17: RS485 (RS422 on AX14032X) baud rates

Value	***Baud rate (bps)***
0	1200
1	2400
2	4800
3	9600
4	19200
5	38400
6	57600
7	115200
8	Use custom baud rate (5611h)

Table 18: RS485 (RS422 on AX14032X) data bits

Value	***Baud rate (bps)***
0	8 bits
1	9 bits

Table 19: RS485 (RS422 on AX14032X) parity options

Value	***Baud rate (bps)***
0	No parity
1	Odd parity
2	Even parity

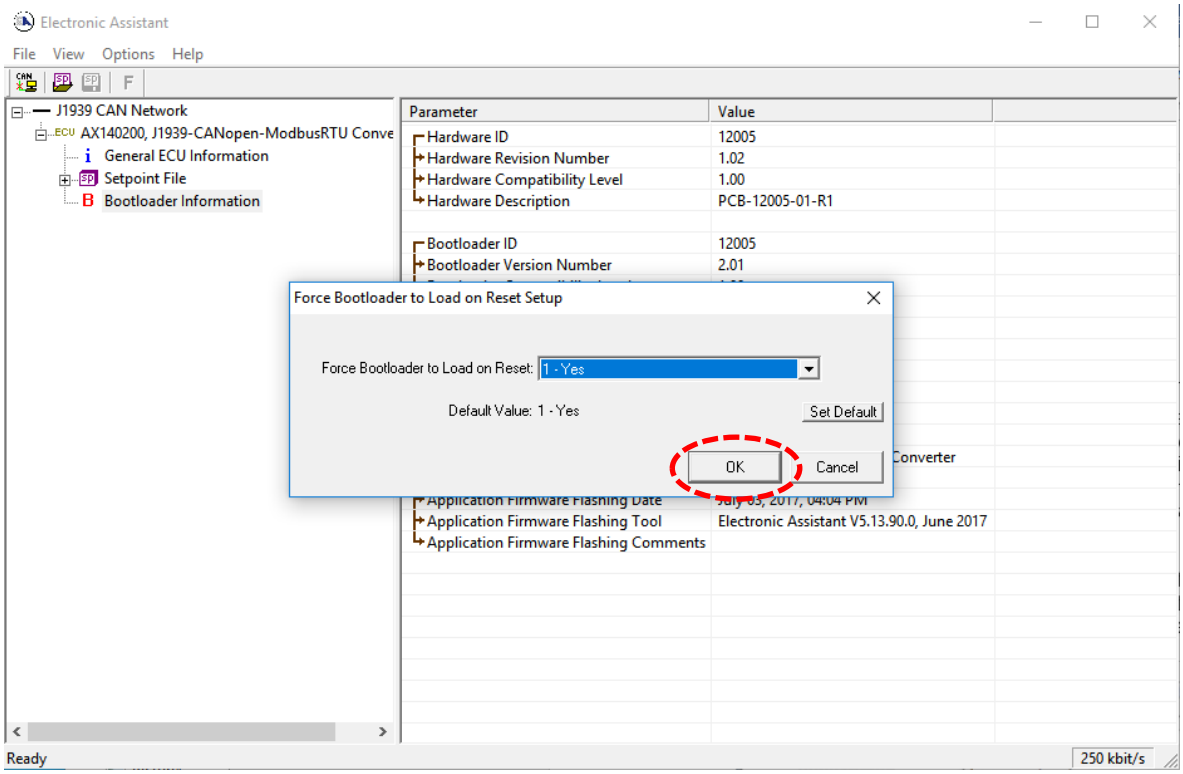
Table 20: RS485 (RS422 on AX14032X) stop bits

Value	***Baud rate (bps)***
0	0.5 bits
1	1 bit
2	1.5 bits
3	2 bits

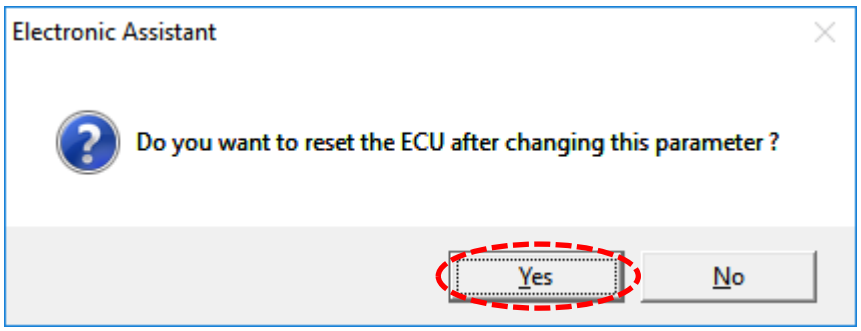
REFLASHING INSTRUCTIONS

The AX140100/AX140200/AX140400 and AX140320/AX140321/AX140322 devices can be upgraded with new application firmware using the Bootloader Information section. This section details the simple step-by-step instructions to upload new firmware provided by Axiomatic onto the unit via CAN, without requiring it to be disconnected from the J1939 network. Please note that the term “AX140x00/AX14032x” in the instructions below refers to all variations of the Protocol Converter devices, such as the AX140100, AX140200, AX140400 and the AX140320, AX140321, AX140322.

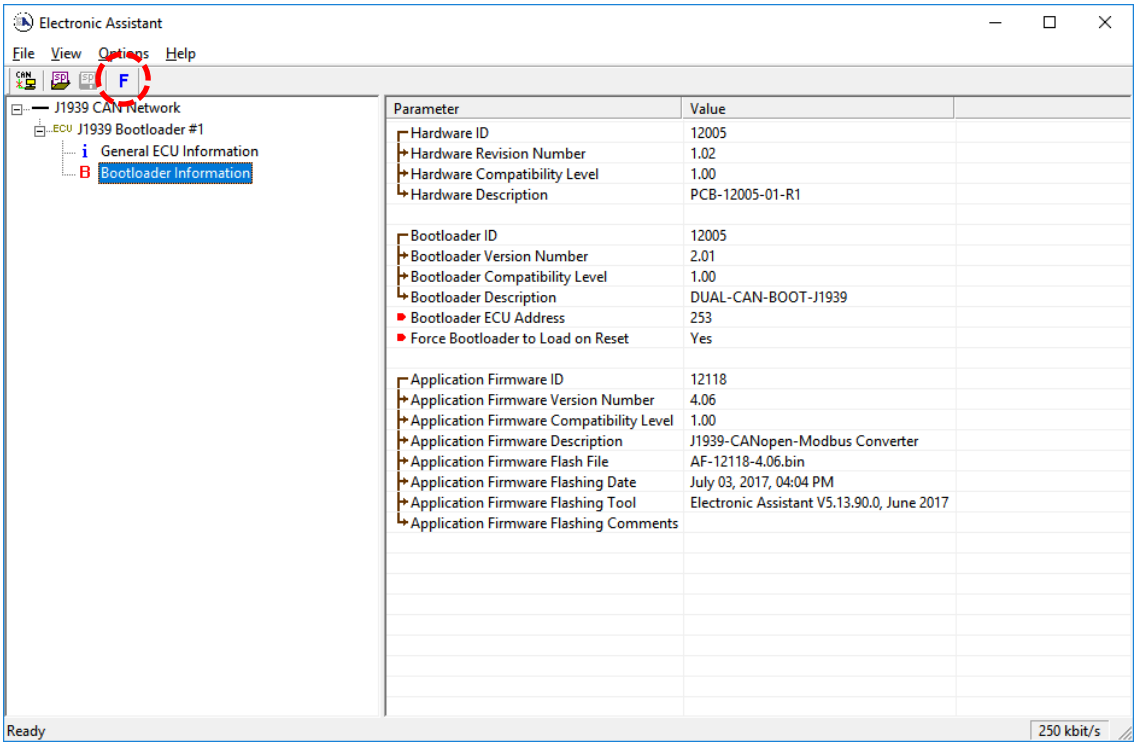
1. When EA first connects to the ECU, the Bootloader Information section will display the following information. To use the bootloader to upgrade the firmware running on the ECU, change the variable “Force Bootloader To Load on Reset” to Yes.



2. When the prompt box asks if you want to reset the ECU, select Yes.

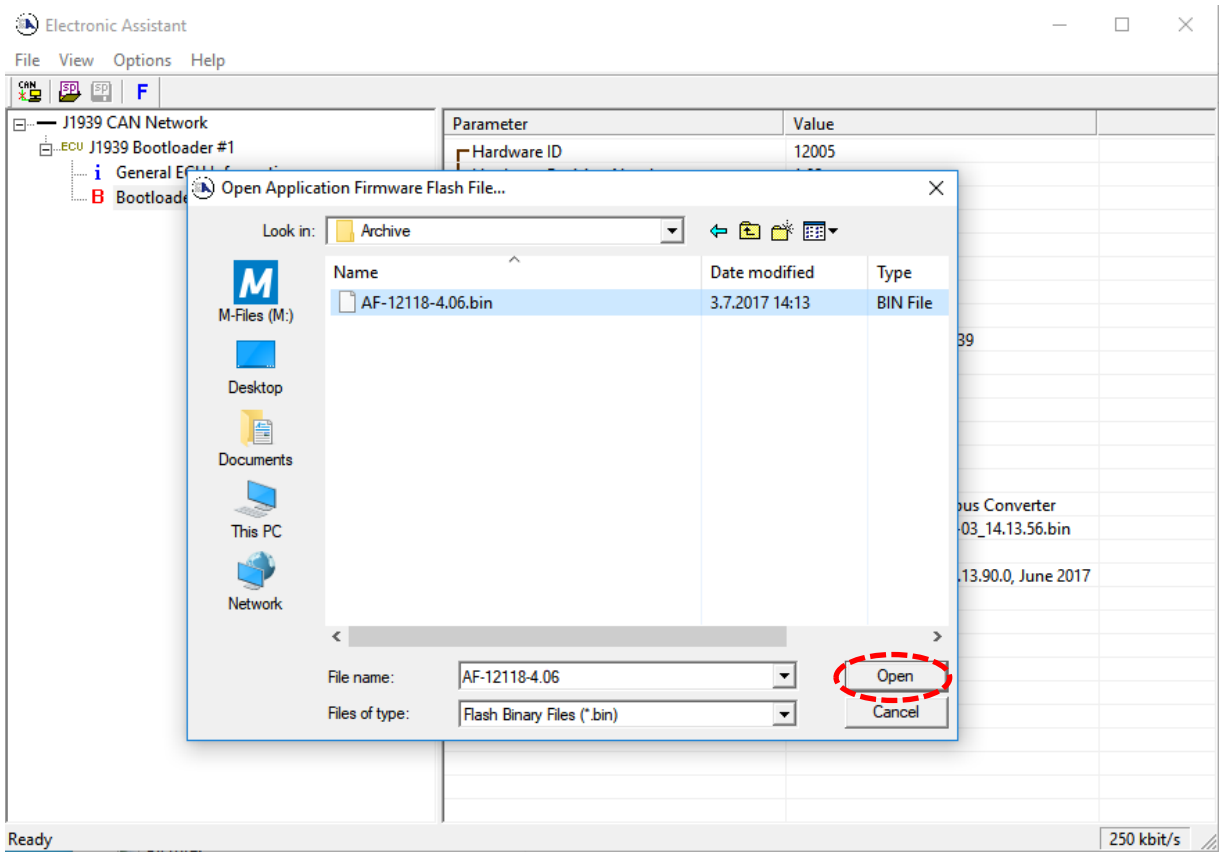


3. Upon reset, the ECU will no longer show up on the J1939 network as an AX140x00/AX14032x device but rather as J1939 Bootloader #1.



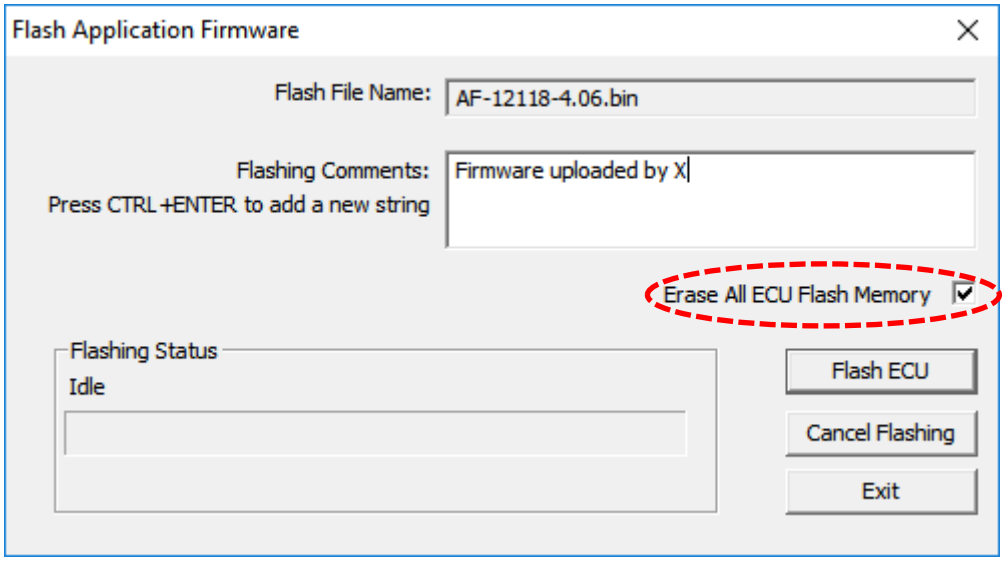
4. When the Bootloader Information section is selected, the same information is shown as when it was running the AX140200 firmware (Application Firmware ID 12118), but in this case the Flashing feature has been enabled.

5. Select the Flashing button and navigate to where you had saved the .bin file sent from Axiomatic. (Note: only binary (.bin) files can be flashed using the EA tool).

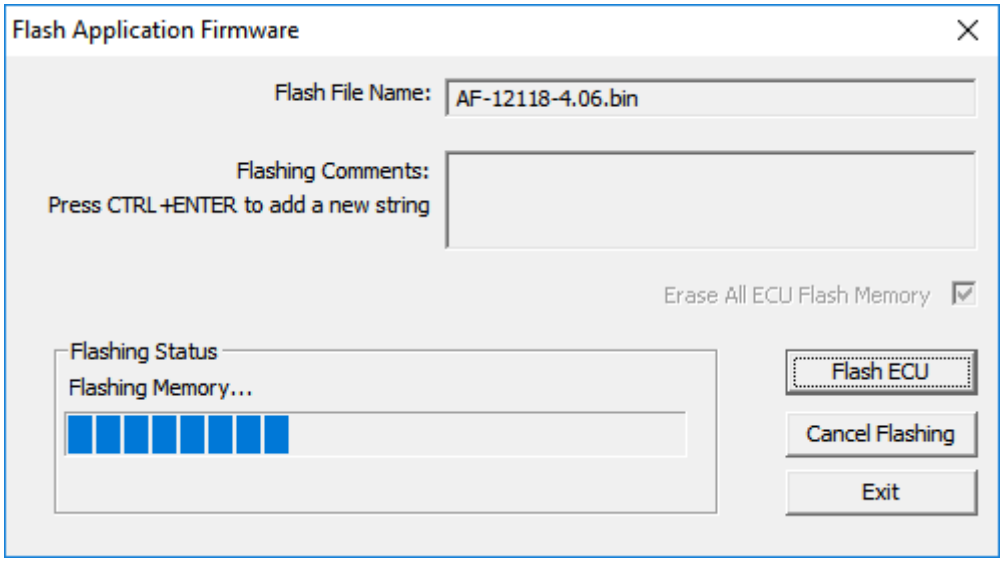


6. Once the Flash Application Firmware window opens, you can enter comments such as “Firmware uploaded by [Name]” if you so desire. This is not required, and you can leave the field blank if you do not want to use it.

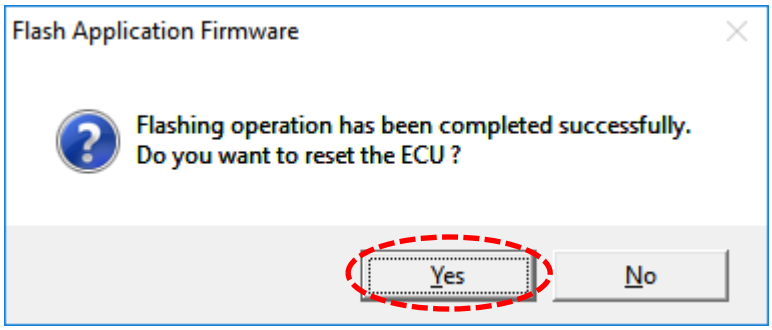
   Note: You do not have to date/timestamp the file, as this is done automatically by the EA tool when you upload the new firmware. **Please note, that it is always good practice to select the “Erase All ECU Flash Memory” option. This will make sure that the whole application Flash is erased and proper default settings are written to Flash. Please also note, that selecting this option will erase all settings previously programmed to the Protocol Converter. Thus, the settings should be saved to PC before reprogramming the hardware.**



7. A progress bar will show how much of the firmware has been sent as the upload progresses. The more traffic there is on the J1939 network, the longer the upload process will take.



8. Once the firmware has finished uploading, a message will pop up indicating the successful operation. If you select to reset the ECU, the new version of the AX140x00/AX14032x application will start running, and the ECU will be identified as such by EA. The next time the ECU is power-cycled, the AX140x00/AX14032x application will run rather than the bootloader function.



VERSION HISTORY

Revision	Date	Author	Changes
Rev. A	29.Oct.2012	Antti Keränen	Initial version.
Rev. B	6.Nov.2012	Antti Keränen	CAN message count increased to 40 INs and 40 OUTs. Routing configuration examples added. Some of the images updated to reflect changes in latest EA.
Rev. C	22.Mar.2013	Antti Keränen	Document updated to consist all device flavors, J1939, CANopen, ModbusRTU/J1587 (all AX140xxx devices).
-	27. Mar. 2013	Amanda Wilkins	Added Technical Specifications as an Appendix
Rev. D	13. June 2013	Antti Keränen	Added reflashing instructions
Rev. E	2. July 2013	Antti Keränen	Table 7 and some CANopen object default values updated.
Rev. F	2. April 2015	Antti Keränen	Description of AX140400 functionality added.
Rev. G	25. June 2015	Antti Keränen	AX140100 part updated.
Rev. H	17. July 2015	Antti Keränen	AX140200 part updated, CANopen Master Script configuration example added. CANopen EMCY description added.
Rev. I	14. Oct. 2015	Antti Keränen	A lot of updates throughout the document for describing the new features.
--	28. Oct. 2015	Amanda Wilkins	Added pin out information for Model AX140400 and further information about the Frame GND.
--	28. Jan. 2016	Antti Keränen	Updated the built in data mappings table (J1939 <-> J1587).
--	14. Mar. 2016	Antti Keränen	CANopen TX PDO data source table updated.
Rev. J	7. Jun. 2016	Antti Keränen	Constant Data block description added.
--	8. June 2016	Amanda Wilkins	Added in CE marking and marine type approvals (DNV-GL and BV) – to Technical Specifications
Rev. K	15. Dec 2016	Antti Keränen	New functionality (Byte order in CAN messages, Modbus special commands) description added. New AX140100-100 converter added. Missing CAN Data Size setpoint description added and CAN Data Type setpoint description corrected.
Rev. L	16. Mar 2017	Antti Keränen	Control Software description in Technical Specifications (Appendix A) corrected.
Rev. M	3. July 2017	Antti Keränen	Section 6 updated. Footer updated to better describe the Protocol Converter versions covered by this user manual.
-	16 November 2017	Amanda Wilkins	Added information on AX140400 to Technical Spec
-	08 January 2018	Amanda Wilkins	Add vibration compliance, updated dimensional drawing and grounding information to Technical Spec
Rev. N	6. February 2018	Antti Keränen	CANopen master script description updated.
Rev. O	5. December 2018	Antti Keränen	Protocol Converter pin description updated. Modbus RTU network parameter description updated and low level CAN <-> RS485 data forwarding functionality description added both for J1939 and CANopen interfaces.
Rev P	15. July 2020	Antti Keränen	Updated Section 3 and Section 6 to reflect that only CAN1 is used for re-flashing application firmware.
Rev Q	28. March 2022	Antti Keränen	Various updates describing the latest features and updates done to the firmware. CANopen section rewritten. AX140320 features and TD section added.
-	29 May 2023	M Ejaz	Marketing review Copied specifications of AX140320 from datasheet into user manual
-	22 November, 2023	M Ejaz	Updated the technical specifications for AX140320 and added those for AX140321 and AX140322 Added P/Ns: AX140321 and AX140322 to the title

Rev R	27 November, 2023	Antti Keränen	Added references to the AX14032x versions of the firmware.
-	23 January, 2024	M Ejaz	Corrected pin out of AX140320, AX140321, and AX140322

APPENDIX A – TECHNICAL SPECIFICATIONS (AX140100, AX140200, AX140400)

Specifications are indicative and subject to change. Actual performance will vary depending on the application and operating conditions. Users should satisfy themselves that the product is suitable for use in the intended application.
All our products carry a limited warranty against defects in material and workmanship. Please refer to our Warranty, Application Approvals/ Limitations and Return Materials Process as described on https://www.axiomatic.com/service/.

Power

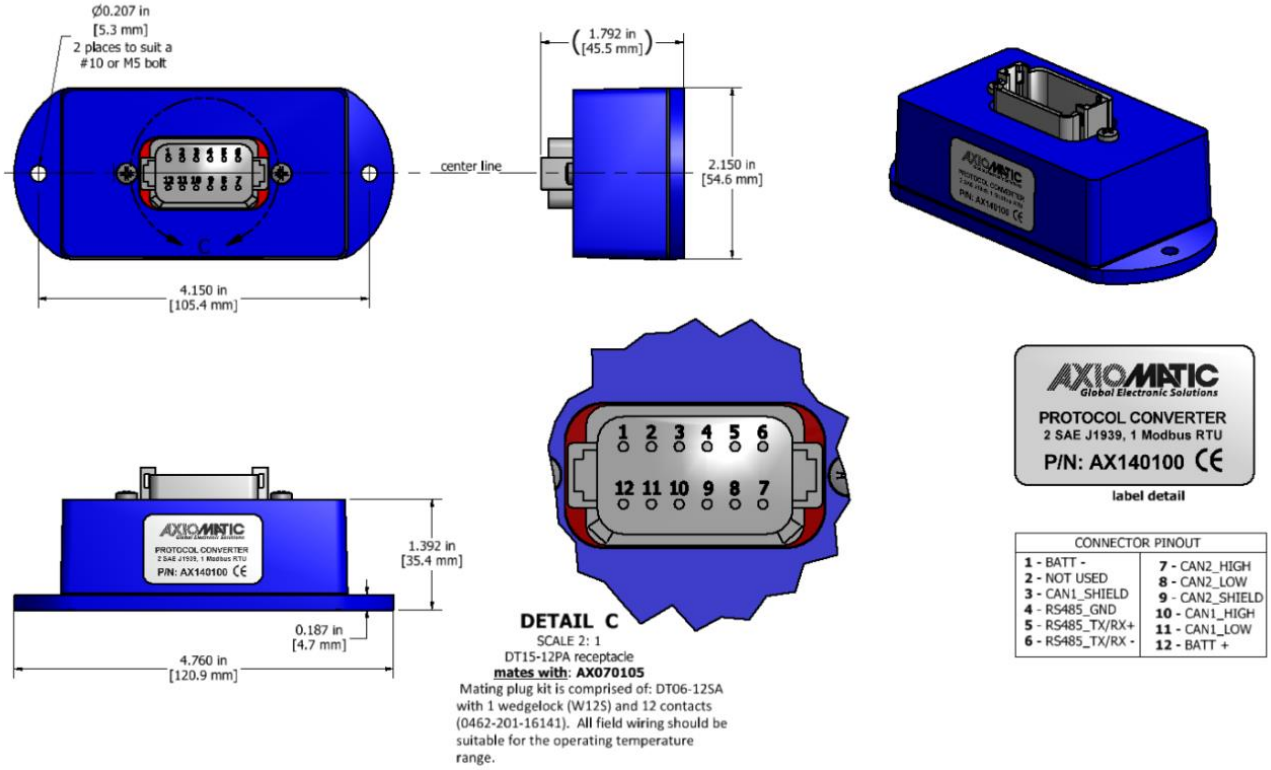
Power Supply Input - Nominal	12 V or 24 Vdc nominal; 9…36 Vdc The minimum allowable supply voltage for the power pin is 8 Vdc.
Surge Protection	95 Vdc
Reverse Polarity Protection	Provided

Control Software

Software Platform	The Protocol Converter comes pre-programmed with standard protocol conversion logic for data exchange between 2 CAN networks and RS-485. The following protocols are available in the standard control logic of model **AX140100:** • SAE J1939 • SAE J1939 • Modbus RTU The following protocols are available in the standard control logic of model **AX140200:** • SAE J1939 (CAN 1 port) • CANopen® (CAN 2 port) • Modbus RTU The following protocols are available in the standard control logic of model **AX140400**. • SAE J1939 (CAN 1 port) • SAE J1939 (CAN 2 port) • SAE J1587 (RS_485 port) Other models are available with CANopen® and J1587.

General Specifications

Memory	STM32F205 32-bit, 512 Kbytes Flash Program Memory
RS-485 Port	1 Isolated RS-485
CAN Ports	2 Isolated CAN 2.0B
Isolation	300 Vrms
Quiescent Current Draw	36 mA @12 V; 19 mA @24 V
Operating Conditions	-40 to 75°C (-40 to 167°F)
Storage Temperature	-55 to 85°C (-67 to 185°F)
Enclosure and Dimensions	Aluminum enclosure, Integral Deutsch IPD connector, Encapsulation



<table>
<tr><td>Electrical Connections</td><td>12 pin Deutsch IPD connector P/N: DT15-12PA
A mating plug kit is available as Axiomatic P/N: AX070105.

<table>
<tr><th colspan="2">CAN and I/O Connector</th></tr>
<tr><th>Pin #</th><th>Description</th></tr>
<tr><td>1</td><td>BATT-</td></tr>
<tr><td>2</td><td>NOT USED</td></tr>
<tr><td>3</td><td>CAN1_SH</td></tr>
<tr><td>4</td><td>RS485_GND (J1587_GND in Model AX140400)</td></tr>
<tr><td>5</td><td>RS485_TX/RX+ (J1587 + in Model AX140400)</td></tr>
<tr><td>6</td><td>RS485_TX/RX- (J1587 – in Model AX140400)</td></tr>
<tr><td>7</td><td>CAN2_H</td></tr>
<tr><td>8</td><td>CAN2_L</td></tr>
<tr><td>9</td><td>CAN2_SH</td></tr>
<tr><td>10</td><td>CAN1_H</td></tr>
<tr><td>11</td><td>CAN1_L</td></tr>
<tr><td>12</td><td>BATT+</td></tr>
</table>

Notes: In the protocol converter all circuits are isolated. BATT-, ground for CAN1 port, ground for CAN2 port, and RS485_GND, are isolated from one another. BATT- (and the other circuit GNDs) is fully isolated from the aluminum housing of the converter. So, the 24VDC supply to the converter is floating. If the housing is taken to EARTH it will not affect the supply to the converter.</td></tr>
<tr><td>Weight</td><td>0.70 lbs. (0.32 kg)</td></tr>
<tr><td>Protection Rating</td><td>IP67; Unit is encapsulated within the housing.</td></tr>
<tr><td>EMC Compliance</td><td>CE marking</td></tr>
<tr><td>Vibration</td><td>4 g
IEC publication 60068-2-6, Test Fc</td></tr>
<tr><td>Marine Type Approvals</td><td>DNV-GL and Bureau Veritas (BV)</td></tr>
<tr><td>Installation</td><td>Mounting holes sized for #10 or M4.5 bolts. The bolt length will be determined by the end-user’s mounting plate thickness. The mounting flange of the controller is 0.19 inches (4.75 mm) thick.

If the module is mounted without an enclosure, it should be mounted to reduce the likelihood of moisture entry. Install the unit with appropriate space available for servicing and for adequate wire harness access (6 inches or 15 cm) and strain relief (12 inches or 30 cm).

The CAN wiring is considered intrinsically safe. The power wires are not considered intrinsically safe and so in hazardous locations, they need to be located in conduit or conduit trays at all times. The module must be mounted in an enclosure in hazardous locations for this purpose.

All field wiring should be suitable for the operating temperature range of the module.

All chassis grounding should go to a single ground point designated for the machine and all related equipment.</td></tr>
<tr><td>User Interface – SAE J1939 models</td><td>For SAE J1939 models, parameters are configurable using the Axiomatic Electronic Assistant.

Axiomatic Electronic Assistant KIT P/N: AX070502, AX070505K, or AX070506K
The Axiomatic Electronic Assistant for Windows operating systems comes with a royalty-free license for use on multiple computers. It requires an Axiomatic USB-CAN converter to link the device’s CAN port to a Windows-based PC.

The functionality of the Axiomatic Electronic Assistant includes but is not limited to the following.
Specify CAN message filters
Allow J1939 PGN’s to be transmitted over CANopen
Link J1587 bus to J1939
Link Modbus to CAN bus
Link CANopen to J1939
Define CANnode ID, and baud rate
Facilitate dynamic decoupling of 2 CAN networks
Monitor CAN data</td></tr>
<tr><td>User Interface – CANopen® models</td><td>.EDS provided to interface to standard CANopen ® tools</td></tr>
</table>

Notes:
CANopen® is a registered community trademark of CAN in Automation e.V.

APPENDIX B – TECHNICAL SPECIFICATIONS (AX140320)

Typical at nominal input voltage and 25 degrees C unless otherwise specified. Specifications are indicative and subject to change. Actual performance will vary depending on the application and operating conditions. Users should satisfy themselves that the product is suitable for use in the intended application.
All our products carry a limited warranty against defects in material and workmanship. Please refer to our Warranty, Application Approvals/ Limitations and Return Materials Process as described on https://www.axiomatic.com/service/.

Power

Power Supply Input	12 or 24 VDC nominal; 9 to 36 VDC The minimum allowable supply voltage for the power pin is 8 VDC.
Quiescent Current Draw	36 mA @12 V; 19 mA @24 V
Surge Protection	95 VDC
Reverse Polarity Protection	Provided

Control Software

Software Platform	The Protocol Converter comes pre-programmed with standard protocol conversion logic for data exchange between 2 CAN networks and RS-422. The following protocols are available in the standard control logic of model. **AX140320**. • SAE J1939 (CAN 1 port) • SAE J1939 (CAN 2 port) • Modbus RTU (RS-422/ RS-485 port) **Custom programming for other applications is available on request.**

General Specifications

Microcontroller	STM32F767 32-bit, 1024 KB Flash Program Memory
RS-422 / RS-485 Port	1 non-isolated RS-422 AX140320 supports RS-485 if • TX+ and RX+ (pins 10 & 4) are connected together (RS485 D+), and • TX- and RX- (pins 9 & 8) are connected together (RS485 D-).
CAN Ports	2 Isolated CAN 2.0B: 2 SAE J1939 (10 kbit/s, 50 kbit/s, 100 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s, 667 kbit/s, 1 Mbit/s. auto-baud-rate detection)
Isolation	300 Vrms
User Interface – SAE J1939 Models	**For SAE J1939 models, parameters are configurable using the Axiomatic Electronic Assistant.** (P/Ns: AX070502 or AX070506K) The Axiomatic Electronic Assistant for *Windows* operating systems comes with a royalty-free license for use on multiple computers. It requires an Axiomatic USB-CAN converter to link the device's CAN port to a *Windows*-based PC. The functionality of the Axiomatic Electronic Assistant includes but is not limited to the following. • Specify CAN message filters • Allow J1939 PGN's to be transmitted over CANopen® • Link Modbus to CAN bus • Define CANnode ID, and baud rate • Facilitate dynamic decoupling of 2 CAN networks • Monitor CAN data
CE/UKCA Marking	Compliant to the EMC Directive Compliant to the RoHS Directive
Vibration	4 g IEC publication 60068-2-6, Test Fc
Protection Rating	IP67
Operating Conditions	-40°C to 70°C (-40°F to 158°F)
Storage Temperature	-55°C to 85°C (-67°F to 185°F)
Weight	0.15 lb. (0.068 kg)
Enclosure and Dimensions	Molded Enclosure, integral connector Nylon 6/6, 30% glass Ultrasonically welded 3.54 in x 2.75 in x 1.31 in (90.09 mm x 70.00 mm x 33.35 mm) L x W x H including integral connector Refer to dimensional drawing.
Electrical Connections	12-pin connector (equivalent TE Deutsch P/N: DT15-12PA) A mating plug kit is available as Axiomatic P/N: **AX070105.**

	CAN and I/O Connector **Pin # / Description** 1 BATT- 2 CAN2_H 3 CAN2_L 4 RS-422_TX- 5 CAN_SH 6 CAN_H 7 CAN_L 8 RS-422_TX+ 9 RS-422_RX- 10 RS-422_RX+ 11 CAN2_SH 12 BATT+
Mating Plug Kit	**PL-DTM06-12SA** Mating Plug Kit : 1 Plug (DTM06-12SA), 1 Wedgelock (WM-12S), 12 Contacts (0462-201-20141), 6 Sealing Plugs (0413-204-2005)
Mounting	Mounting holes are sized for #8 or M4 bolts. The bolt length will be determined by the end-user's mounting plate thickness. The mounting flange of the controller is 0.425 inches (10.8 mm) thick. If the module is mounted without an enclosure, it should be mounted vertically with connectors facing left or right to reduce likelihood of moisture entry. The CAN wiring is considered intrinsically safe. The power wires are not considered intrinsically safe and so in hazardous locations, they need to be located in conduit or conduit trays at all times. The module must be mounted in an enclosure in hazardous locations for this purpose. No wire or cable harness should exceed 30 meters in length. The power input wiring should be limited to 10 meters. All field wiring should be suitable for the operating temperature range. Install the unit with appropriate space available for servicing and for adequate wire harness access (6 inches or 15 cm) and strain relief (12 inches or 30 cm).

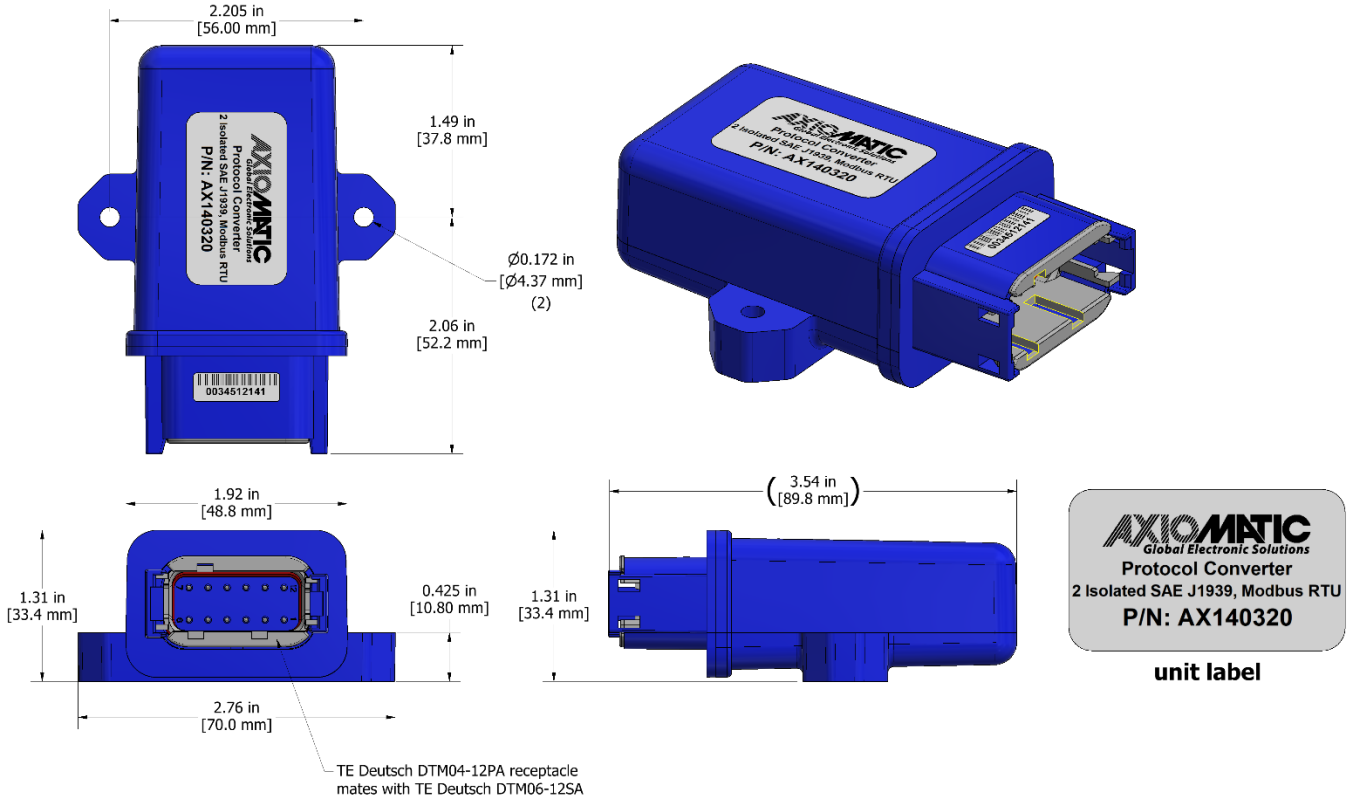


Figure 1.0 – Dimensional Drawing

CANopen® is a registered community trademark of CAN in Automation e.V.

APPENDIX C – TECHNICAL SPECIFICATIONS (AX140321)

Typical at nominal input voltage and 25 degrees C unless otherwise specified. Specifications are indicative and subject to change. Actual performance will vary depending on the application and operating conditions. Users should satisfy themselves that the product is suitable for use in the intended application.
All our products carry a limited warranty against defects in material and workmanship. Please refer to our Warranty, Application Approvals/ Limitations and Return Materials Process as described on https://www.axiomatic.com/service/.

Power

Power Supply Input	12 or 24 VDC nominal; 9 to 36 VDC The minimum allowable supply voltage for the power pin is 8 VDC.
Quiescent Current Draw	36 mA @12 V; 19 mA @24 V
Surge Protection	95 VDC
Reverse Polarity Protection	Provided

Control Software

Software Platform	The Protocol Converter comes pre-programmed with standard protocol conversion logic for data exchange between 2 CAN networks and RS-422. The following protocols are available in the standard control logic of model **AX140321**. • CANopen® (CAN 2 port) • SAE J1939 (CAN 1 port) • Modbus RTU (RS-422/ RS-485 port) **Custom programming for other applications is available on request.**

General Specifications

Microcontroller	STM32F767 32-bit, 1024 KB Flash Program Memory
RS-422 / RS-485 Port	1 non-isolated RS-422 AX140321 supports RS-485 if • TX+ and RX+ (pins 10 & 4) are connected together (RS485 D+), and • TX- and RX- (pins 9 & 8) are connected together (RS485 D-).
CAN Ports	2 Isolated CAN 2.0B: 1 SAE J1939 (10 kbit/s, 50 kbit/s, 100 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s, 667 kbit/s, 1 Mbit/s auto-baud-rate detection) 1 CANopen®
Isolation	300 Vrms
User Interface – SAE J1939	**For SAE J1939 models, parameters are configurable using the Axiomatic Electronic Assistant (**P/Ns: AX070502 or AX070506K). The Axiomatic Electronic Assistant for *Windows* operating systems comes with a royalty-free license for use on multiple computers. It requires an Axiomatic USB-CAN converter to link the device's CAN port to a *Windows*-based PC. The functionality of the Axiomatic Electronic Assistant includes but is not limited to the following. • Specify CAN message filters • Allow J1939 PGN's to be transmitted over CANopen® • Link Modbus to CAN bus • Link CANopen® to J1939 • Define CANnode ID, and baud rate • Facilitate dynamic decoupling of 2 CAN networks • Monitor CAN data
User Interface – CANopen®	EDS provided to interface to standard CANopen® tools
CE/UKCA Marking	Compliant to the EMC Directive Compliant to the RoHS Directive
Vibration	4 g IEC publication 60068-2-6, Test Fc
Protection Rating	IP67
Operating Conditions	-40°C to 70°C (-40°F to 158°F)
Storage Temperature	-55°C to 85°C (-67°F to 185°F)
Weight	0.15 lb. (0.068 kg)
Enclosure and Dimensions	Molded Enclosure, integral connector Nylon 6/6, 30% glass Ultrasonically welded 3.54 in x 2.75 in x 1.31 in (90.09 mm x 70.00 mm x 33.35 mm) L x W x H including integral connector Refer to dimensional drawing.
Electrical Connections	12-pin connector (equivalent TE Deutsch P/N: DT15-12PA) A mating plug kit is available as Axiomatic P/N: **AX070105.**

	CAN and I/O Connector **Pin # / Description** 1 BATT- 2 CAN2_H 3 CAN2_L 4 RS-422_TX- 5 CAN_SH 6 CAN_H 7 CAN_L 8 RS-422_TX+ 9 RS-422_RX- 10 RS-422_RX+ 11 CAN2_SH 12 BATT+
Mating Plug Kit	**PL-DTM06-12SA** Mating Plug Kit : 1 Plug (DTM06-12SA), 1 Wedgelock (WM-12S), 12 Contacts (0462-201-20141), 6 Sealing Plugs (0413-204-2005)
Mounting	Mounting holes are sized for #8 or M4 bolts. The bolt length will be determined by the end-user's mounting plate thickness. The mounting flange of the controller is 0.425 inches (10.8 mm) thick. If the module is mounted without an enclosure, it should be mounted vertically with connectors facing left or right to reduce the likelihood of moisture entry. The CAN wiring is considered intrinsically safe. The power wires are not considered intrinsically safe and so in hazardous locations, they need to be located in conduit or conduit trays at all times. The module must be mounted in an enclosure in hazardous locations for this purpose. No wire or cable harness should exceed 30 meters in length. The power input wiring should be limited to 10 meters. All field wiring should be suitable for the operating temperature range. Install the unit with appropriate space available for servicing and for adequate wire harness access (6 inches or 15 cm) and strain relief (12 inches or 30 cm).

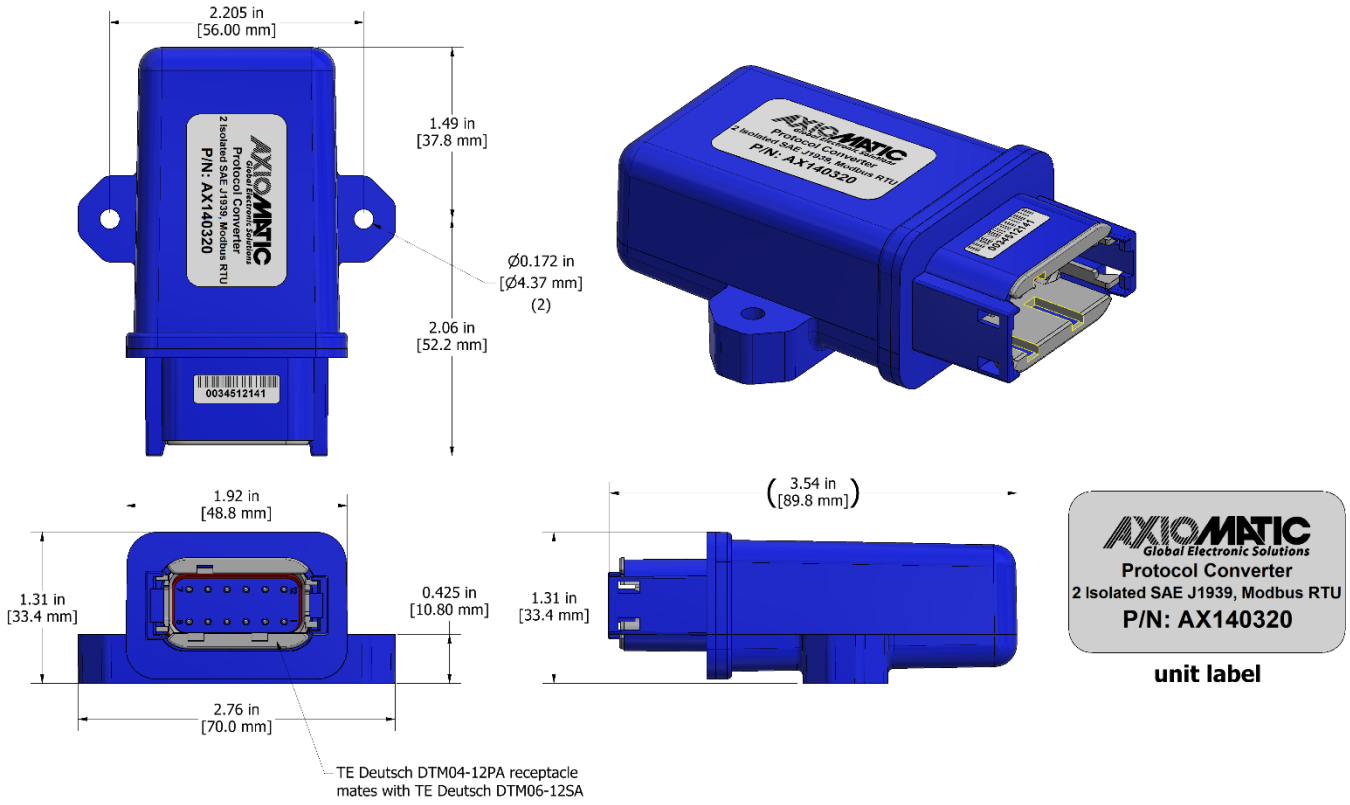


Figure 1.0 – Dimensional Drawing (AX140320 and AX140321 have the same dimensions.)

CANopen® is a registered community trademark of CAN in Automation e.V.

APPENDIX D – TECHNICAL SPECIFICATIONS (AX140322)

Typical at nominal input voltage and 25 degrees C unless otherwise specified. Specifications are indicative and subject to change. Actual performance will vary depending on the application and operating conditions. Users should satisfy themselves that the product is suitable for use in the intended application.
All our products carry a limited warranty against defects in material and workmanship. Please refer to our Warranty, Application Approvals/ Limitations and Return Materials Process as described on https://www.axiomatic.com/service/.

Power

Power Supply Input	12 or 24 VDC nominal; 9 to 36 VDC The minimum allowable supply voltage for the power pin is 8 VDC.
Quiescent Current Draw	36 mA @12 V; 19 mA @24 V
Surge Protection	95 VDC
Reverse Polarity Protection	Provided

Control Software

Software Platform	The Protocol Converter comes pre-programmed with standard protocol conversion logic for data exchange between 2 CAN networks and RS-422. The following protocols are available in the standard control logic of model **AX140322**. • SAE J1939 (CAN 1 port) • SAE J1939 (CAN 2 port) • SAE J1587 (RS-422/ RS-485 port) **Custom programming for other applications is available on request.**

General Specifications

Microcontroller	STM32F767 32-bit, 1024 KB Flash Program Memory
RS-422 / RS-485 Port	1 non-isolated RS-422 AX140322 supports RS-485 if • TX+ and RX+ (pins 10 & 4) are connected together (RS485 D+), and • TX- and RX- (pins 9 & 8) are connected together (RS485 D-).
CAN Ports	2 Isolated CAN 2.0B: 2 SAE J1939 (10 kbit/s, 50 kbit/s, 100 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s, 667 kbit/s, 1 Mbit/s. auto-baud-rate detection)
Isolation	300 Vrms
User Interface – SAE J1939	**For SAE J1939 models, parameters are configurable using the Axiomatic Electronic Assistant.** (P/Ns: AX070502 or AX070506K) The Axiomatic Electronic Assistant for *Windows* operating systems comes with a royalty-free license for use on multiple computers. It requires an Axiomatic USB-CAN converter to link the device's CAN port to a *Windows*-based PC. The functionality of the Axiomatic Electronic Assistant includes but is not limited to the following. • Specify CAN message filters • Allow J1939 PGN's to be transmitted over CANopen® • Link J1587 bus to J1939 • Link Modbus to CAN bus • Define CANnode ID, and baud rate • Facilitate dynamic decoupling of 2 CAN networks • Monitor CAN data
CE/UKCA Marking	Compliant to the EMC Directive Compliant to the RoHS Directive
Vibration	4 g IEC publication 60068-2-6, Test Fc
Protection Rating	IP67
Operating Conditions	-40°C to 70°C (-40°F to 158°F)
Storage Temperature	-55°C to 85°C (-67°F to 185°F)
Weight	0.15 lb. (0.068 kg)
Enclosure and Dimensions	Molded Enclosure, integral connector Nylon 6/6, 30% glass Ultrasonically welded 3.54 in x 2.75 in x 1.31 in (90.09 mm x 70.00 mm x 33.35 mm) L x W x H including integral connector Refer to dimensional drawing.
Electrical Connections	12-pin connector (equivalent TE Deutsch P/N: DT15-12PA) A mating plug kit is available as Axiomatic P/N: **AX070105**.

	CAN and I/O Connector **Pin # / Description** 1 BATT- 2 CAN2_H 3 CAN2_L 4 RS-422_TX- 5 CAN_SH 6 CAN_H 7 CAN_L 8 RS-422_TX+ 9 RS-422_RX- 10 RS-422_RX+ 11 CAN2_SH 12 BATT+
Mating Plug Kit	**PL-DTM06-12SA** Mating Plug Kit : 1 Plug (DTM06-12SA), 1 Wedgelock (WM-12S), 12 Contacts (0462-201-20141), 6 Sealing Plugs (0413-204-2005)
Mounting	Mounting holes are sized for #8 or M4 bolts. The bolt length will be determined by the end-user's mounting plate thickness. The mounting flange of the controller is 0.425 inches (10.8 mm) thick. If the module is mounted without an enclosure, it should be mounted vertically with connectors facing left or right to reduce likelihood of moisture entry. The CAN wiring is considered intrinsically safe. The power wires are not considered intrinsically safe and so in hazardous locations, they need to be located in conduit or conduit trays at all times. The module must be mounted in an enclosure in hazardous locations for this purpose. No wire or cable harness should exceed 30 meters in length. The power input wiring should be limited to 10 meters. All field wiring should be suitable for the operating temperature range. Install the unit with appropriate space available for servicing and for adequate wire harness access (6 inches or 15 cm) and strain relief (12 inches or 30 cm).

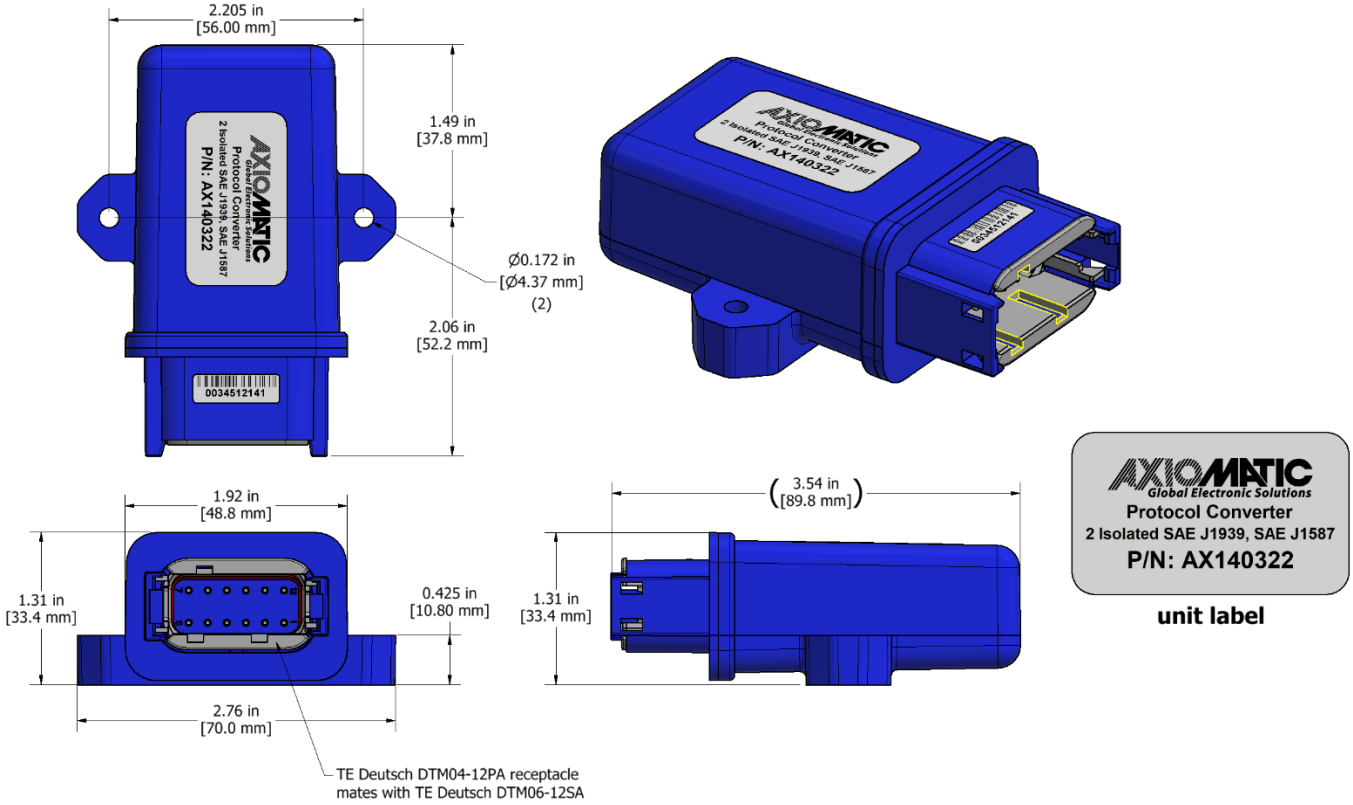


Figure 1.0 – Dimensional Drawing

CANopen® is a registered community trademark of CAN in Automation e.V.

OUR PRODUCTS

AC/DC Power Supplies

Actuator Controls/Interfaces

Automotive Ethernet Interfaces

Battery Chargers

CAN Controls, Routers, Repeaters

CAN/WiFi, CAN/Bluetooth, Routers

Current/Voltage/PWM Converters

DC/DC Power Converters

Engine Temperature Scanners

Ethernet/CAN Converters, Gateways, Switches

Fan Drive Controllers

Gateways, CAN/Modbus, RS-232

Gyroscopes, Inclinometers

Hydraulic Valve Controllers

Inclinometers, Triaxial

I/O Controls

LVDT Signal Converters

Machine Controls

Modbus, RS-422, RS-485 Controls

Motor Controls, Inverters

Power Supplies, DC/DC, AC/DC

PWM Signal Converters/Isolators

Resolver Signal Conditioners

Service Tools

Signal Conditioners, Converters

Strain Gauge CAN Controls

Surge Suppressors

OUR COMPANY

Axiomatic provides electronic machine control components to the off-highway, commercial vehicle, electric vehicle, power generator set, material handling, renewable energy and industrial OEM markets. ***We innovate with engineered and off-the-shelf machine controls that add value for our customers.***

QUALITY DESIGN AND MANUFACTURING

We have an ISO9001:2015 registered design/manufacturing facility in Canada.

WARRANTY, APPLICATION APPROVALS/LIMITATIONS

Axiomatic Technologies Corporation reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. Users should satisfy themselves that the product is suitable for use in the intended application. All our products carry a limited warranty against defects in material and workmanship. Please refer to our Warranty, Application Approvals/Limitations and Return Materials Process at https://www.axiomatic.com/service/.

COMPLIANCE

Product compliance details can be found in the product literature and/or on axiomatic.com. Any inquiries should be sent to sales@axiomatic.com.

SAFE USE

All products should be serviced by Axiomatic. Do not open the product and perform the service yourself.

This product can expose you to chemicals which are known in the State of California, USA to cause cancer and reproductive harm. For more information go to www.P65Warnings.ca.gov.

SERVICE

All products to be returned to Axiomatic require a Return Materials Authorization Number (RMA#) from rma@axiomatic.com. Please provide the following information when requesting an RMA number:

- Serial number, part number
- Runtime hours, description of problem
- Wiring set up diagram, application and other comments as needed

DISPOSAL

Axiomatic products are electronic waste. Please follow your local environmental waste and recycling laws, regulations and policies for safe disposal or recycling of electronic waste.

CONTACTS

Axiomatic Technologies Corporation
1445 Courtneypark Drive E.
Mississauga, ON
CANADA L5T 2E3
TEL: +1 905 602 9270
FAX: +1 905 602 9279
www.axiomatic.com
sales@axiomatic.com

Axiomatic Technologies Oy
Höytämöntie 6
33880 Lempäälä
FINLAND
TEL: +358 103 375 750
www.axiomatic.com
salesfinland@axiomatic.com