USER MANUAL UMAX142100A
Version 2.0.1

RS232-RS232-RS422 ROUTER WITH ETHERNET AND CAN

USER MANUAL

P/N: AX142100A

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

VERSION HISTORY

Version	Date	Author	Modification
1.0.0.	May 9, 2023	Antti Keränen	Initial Draft
2.0.0.	September 19, 2024	Antti Keränen	Descriptions added for Receive Message Configuration and Transmit Message Configuration. Configuration web pages' screenshots updated.
2.0.1	September 19, 2024	M Ejaz	Marketing review Updated power protections and quiescent current as per validation results Updated dimensional drawing & storage temperature

ℹ The default password: '**AX142100A**'

ACCRONYMS

ACK	Positive Acknowledgement (from SAE J1939 standard)
BATT +/-	Battery positive (a.k.a. Vps) or Battery Negative (a.k.a. GND)
DM	Diagnostic Message (from SAE J1939 standard)
DTC	Diagnostic Trouble Code (from SAE J1939 standard)
EA	Axiomatic Electronic Assistant (A Service Tool for Axiomatic ECUs)
ECU	Electronic Control Unit (from SAE J1939 standard)
GND	Ground reference (a.k.a. BATT-)
I/O	Inputs and Outputs
IP	Internet Protocol
MAC	Media Access Control
MAP	Memory Access Protocol
NAK	Negative Acknowledgement (from SAE J1939 standard)
PDU1	A format for messages that are to be sent to a destination address, either specific or global (from SAE J1939 standard)
PDU2	A format used to send information that has been labeled using the Group Extension technique, and does not contain a destination address.
PGN	Parameter Group Number (from SAE J1939 standard)
PropA	Message that uses the Proprietary A PGN for peer-to-peer communication
PropB	Message that uses a Proprietary B PGN for broadcast communication
SPN	Suspect Parameter Number (from SAE J1939 standard)
TCP/IP	Transmission Control Protocol / Internet Protocol
TP	Transport Protocol
Vps	Voltage Power Supply (a.k.a. BATT+)

TABLE OF CONTENTS

List of Figures

List of Tables

REFERENCES

J1939	Recommended Practice for a Serial Control and Communications Vehicle Network, SAE, April 2011
J1939/21	Data Link Layer, SAE, December 2010
J1939/71	Vehicle Application Layer, SAE, March 2011
J1939/73	Application Layer-Diagnostics, SAE, February 2010
J1939/81	Network Management, SAE, May 2003
TDAX142100A	Technical Datasheet, RS232-RS232-RS422-ENET-CAN Converter, Axiomatic Technologies
UMAX07050x	User Manual, Axiomatic Electronic Assistant and USB-CAN, Axiomatic Technologies

This document assumes the reader is familiar with the SAE J1939 standard. Terminology from the standard is used, but not described in this document.

NOTE: This product is supported by Axiomatic Electronic Assistant V5.15.129.0 and higher

1. OVERVIEW OF CONTROLLER

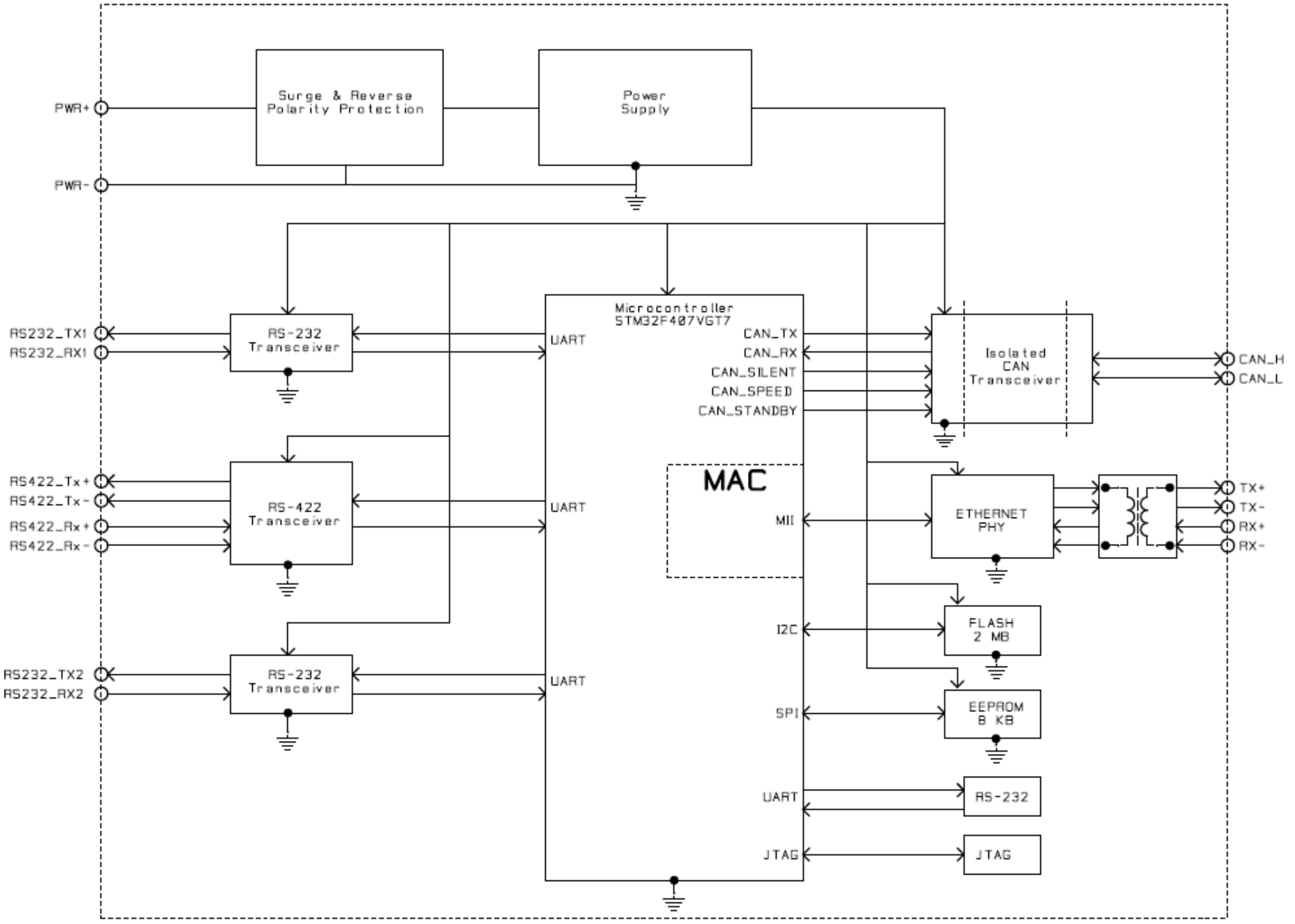


Figure 1 – Block diagram of the RS232-RS232-RS422 Router with Ethernet and CAN

The RS232-RS232-RS422 Router with Ethernet and CAN (later 3RS-ENET-CAN) is a device that forwards serial port messages between the three serial ports, CAN and Ethernet based on a custom routing configuration. The configuration can be done using a web browser and the built-in web server running on the 3RS-ENET-CAN device.

The Axiomatic Electronic Assistant can be used to configure the network parameters of the 3RS-ENET-CAN converter. The configuration of all parameters can be done via the web browser interface (port 80).

2. INSTALLATION INSTRUCTIONS

2.1. Dimensions and Pinout

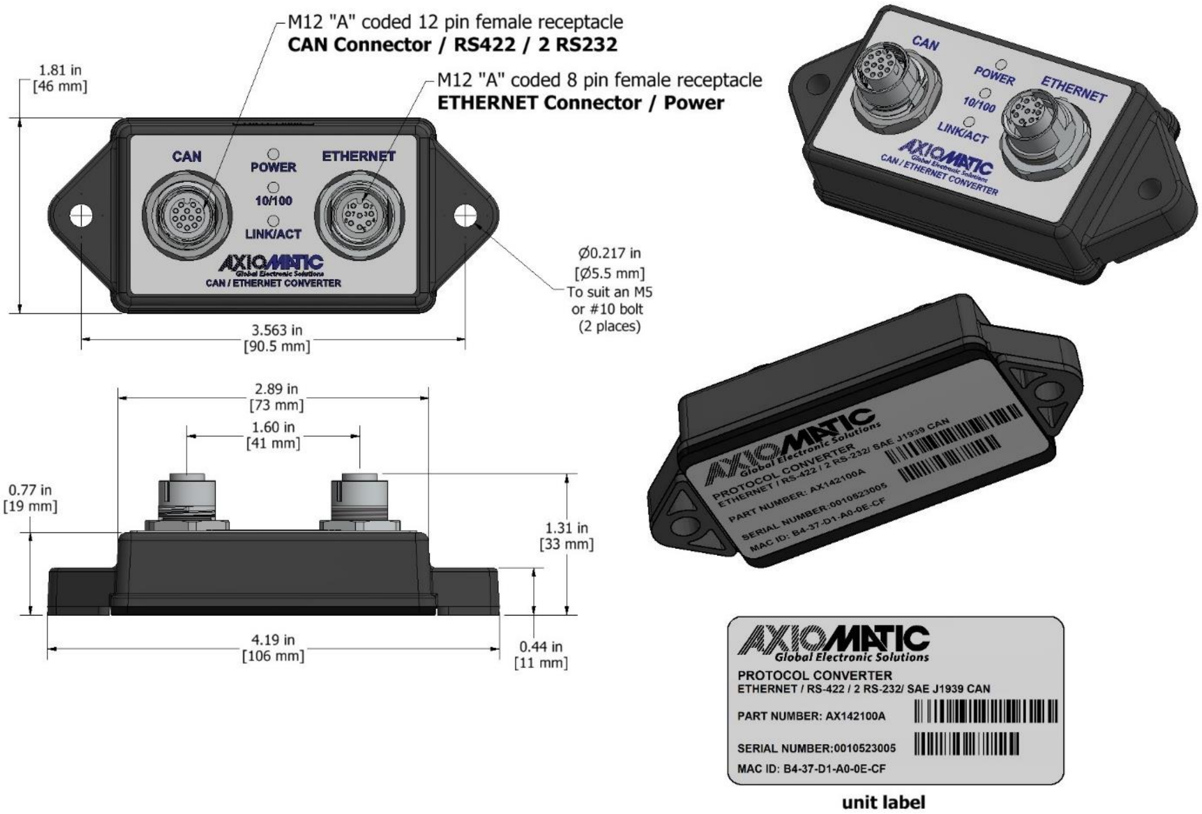


Figure 2 – Controller Dimensions and Label

CAN connector / 2xRS232		Ethernet connector / RS422	
Pin #	**Function**	**Pin #**	**Function**
1	RS-422 RX+	1	Power +
2	RS-422 TX+	2	Power -
3	RS-422 RX-	3	Power -
4	RS-232 TX 2	4	Ethernet TX -
5	RS-232 RX 2	5	Ethernet RX +
6	CAN_L	6	Ethernet TX +
7	CAN_H	7	Power +
8	RS-232 TX 1	8	Ethernet RX -
9	RS-232 RX 1		
10	RS-422 TX-		
11	Ground		
12	Ground		

Table 1 – AX142100A Connector Pinout

3. OVERVIEW OF J1939 FEATURES

The software was designed to provide flexibility to the user with respect to messages sent from the ECU by providing:

- Configurable ECU Instance in the NAME (to allow multiple ECUs on the same network)
- Configurable PGN and Data Parameters
- Configurable Diagnostic Messaging Parameters, as required

3.1. Introduction to Supported Messages

The ECU is compliant with the standard SAE J1939, and supports following PGNs from the standard.

From J1939-21 – Data Link Layer

Request		59904	0x00EA00
Acknowledgement		59392	0x00E800
Transport Protocol – Connection Management		60416	0x00EC00
Transport Protocol – Data Transfer Message		60160	0x00EB00
Proprietary B	from	65280	0x00FF00
	to	65535	0x00FFFF

From J1939-73 – Diagnostics

DM1 – Active Diagnostic Trouble Codes	65226	0x00FECA
DM2 – Previously Active Diagnostic Trouble Codes	65227	0x00FECB
DM3 – Diagnostic Data Clear/Reset for Previously Active DTCs	65228	0x00FECC
DM11 – Diagnostic Data Clear/Reset for Active DTCs	65235	0x00FED3

From J1939-81 – Network Management

Address Claimed/Cannot Claim	60928	0x00EE00
Commanded Address	65240	0x00FED8

From J1939-71 – Vehicle Application Layer

ECU Identification Information	64965	0x00FDC5
Software Identification	65242	0x00FEDA
Component Identification	65259	0x00FEEB

None of the application layer PGNs are supported as part of the default configurations, but they can be selected as desired for transmit function blocks.

Setpoints are accessed using standard Memory Access Protocol (MAP) with proprietary addresses. The Axiomatic Electronic Assistant (EA) allows for quick and easy configuration of the unit over CAN network.

3.2. NAME, Address and Identification Information

The 3RS-ENET-CAN ECU has the following default for the J1939 NAME. The user should refer to the SAE J1939/81 standard for more information on these parameters and their ranges.

Arbitrary Address Capable	Yes
Industry Group	0, Global
Vehicle System Instance	0
Vehicle System	0, Non-specific system
Function	25, Axiomatic Protocol Converter
Function Instance	21, Axiomatic AX142100A
ECU Instance	0, First Instance
Manufacture Code	162, Axiomatic Technologies
Identity Number	Variable, uniquely assigned during factory programming for each ECU

The ECU Instance is a configurable setpoint associated with the NAME. Changing this value will allow multiple ECUs of this type to be distinguishable from one another when they are connected on the same network.

The default value of the “ECU Address” setpoint is 128 (0x80), which is the preferred starting address for self-configurable ECUs as set by the SAE in J1939 tables B3 and B7. The EA will allow the selection of any address between 0 and 253. ***It is the user’s responsibility to select an address that complies with the standard***. The user must also be aware that since the unit is arbitrary address capable, if another ECU with a higher priority NAME contends for the selected address, the 10 Analog input will continue select the next highest address until it finds one that it can claim. See J1939/81 for more details about address claiming.

ECU Identification Information

PGN 64965 ECU Identification Information -ECUID

Transmission Repetition Rate:	On request
Data Length:	Variable
Extended Data Page:	0
Data Page:	0
PDU Format:	253
PDU Specific:	197 PGN Supporting Information:
Default Priority:	6
Parameter Group Number:	64965 (0x00FDC5)

Start Position	Length	Parameter Name	SPN
a	Variable	ECU Part Number, Delimiter (ASCII "*")	2901
b	Variable	ECU Serial Number, Delimiter (ASCII "*")	2902
c	Variable	ECU Location, Delimiter (ASCII "*")	2903
d	Variable	ECU Type, Delimiter (ASCII "*")	2904
e	Variable	ECU Manufacturer Name, Delimiter (ASCII "*")	4304

(a)*(b)*(c)*(d)*(e)*

Software Identifier

PGN 65242 Software Identification -SOFT

Transmission Repetition Rate:	On request
Data Length:	Variable
Extended Data Page:	0
Data Page:	0
PDU Format:	254
PDU Specific:	218 PGN Supporting Information:
Default Priority:	6
Parameter Group Number:	65242 (0x00FEDA)

Start Position	Length	Parameter Name	SPN
1	1 Byte	Number of software identification fields	965
2-n	Variable	Software identification(s), Delimiter (ASCII "*")	234

Byte 1 is set to 5, and the identification fields are as follows.

(Part Number)*(Version)*(Date)*(Owner)*(Description)

The EA shows all this information in its "General ECU Information" page. *Note: The information provided in the Software ID is available for any J1939 service tool which supports the PGN -SOFT*

Component Identification

PGN 65259	Component Identification	-CI

Transmission Repetition Rate:	On request
Data Length:	Variable
Extended Data Page:	0
Data Page:	0
PDU Format:	254
PDU Specific:	235 PGN Supporting Information:
Default Priority:	6
Parameter Group Number:	65259 (0x00FEEB)

Start Position	Length	Parameter Name	SPN
a	1-5 Byte	Make, Delimiter (ASCII “*”)	586
b	Variable	Model, Delimiter (ASCII “*”)	587
c	Variable	Serial Number, Delimiter (ASCII “*”)	588
d	Variable	Unit Number (Power Unit), Delimiter (ASCII “*”)	233

(a)*(b)*(c)*(d)*(e)*

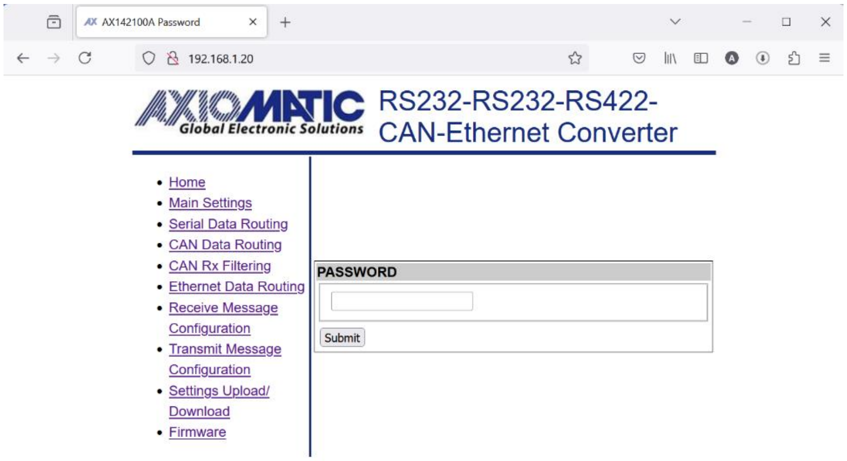
4. WEB BROWSER BASED CONTROLLER CONFIGURATION

The 3RS-ENET-CAN controller supports configuration of the data routing parameters from Ethernet port using a standard web browser.

The default password: ‘**AX142100A**’

4.1. Parameter Editing

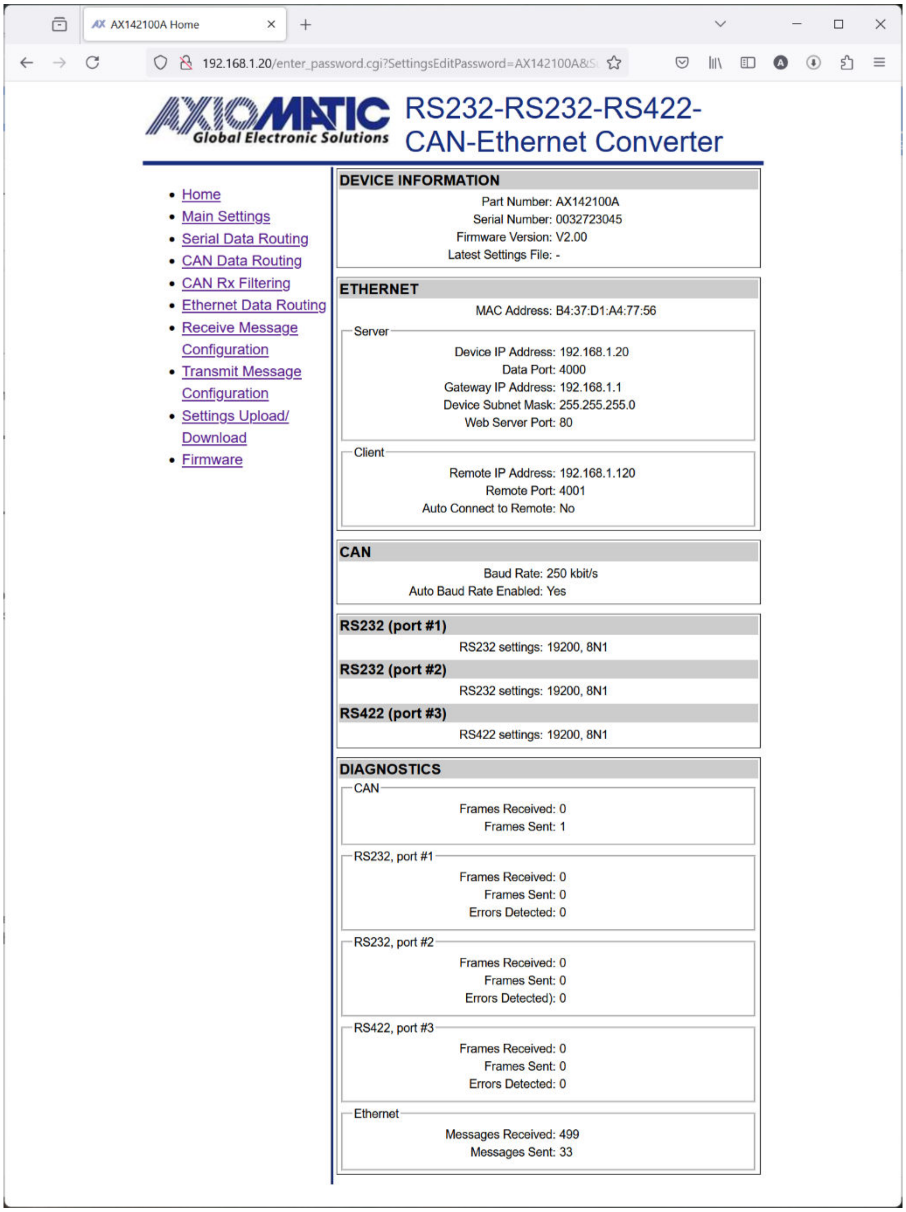
The 3RS-ENET-CAN has a web server running on TCP port 80. The web server asks for a password before the configuration pages can be accessed. The default password is ‘**AX142100A**’ (this is case sensitive).



When the correct password is entered, the configuration page is opened. The settings can be applied by clicking the button at the top of the page. In case the user doesn’t want to change settings, the connection can be closed.

<configured ip>

<configured ip>/index.shtml


AX142100A Home
192.168.1.20/enter_password.cgi?SettingsEditPassword=AX142100A&S
AXIOMATIC Global Electronic Solutions

RS232-RS232-RS422-
CAN-Ethernet Converter

- Home
- Main Settings
- Serial Data Routing
- CAN Data Routing
- CAN Rx Filtering
- Ethernet Data Routing
- Receive Message Configuration
- Transmit Message Configuration
- Settings Upload/ Download
- Firmware

DEVICE INFORMATION

Part Number: AX142100A
Serial Number: 0032723045
Firmware Version: V2.00
Latest Settings File: -

ETHERNET

MAC Address: B4:37:D1:A4:77:56

Server
Device IP Address: 192.168.1.20
Data Port: 4000
Gateway IP Address: 192.168.1.1
Device Subnet Mask: 255.255.255.0
Web Server Port: 80

Client
Remote IP Address: 192.168.1.120
Remote Port: 4001
Auto Connect to Remote: No

CAN

Baud Rate: 250 kbit/s
Auto Baud Rate Enabled: Yes

RS232 (port #1)

RS232 settings: 19200, 8N1

RS232 (port #2)

RS232 settings: 19200, 8N1

RS422 (port #3)

RS422 settings: 19200, 8N1

DIAGNOSTICS

CAN
Frames Received: 0
Frames Sent: 1

RS232, port #1
Frames Received: 0
Frames Sent: 0
Errors Detected: 0

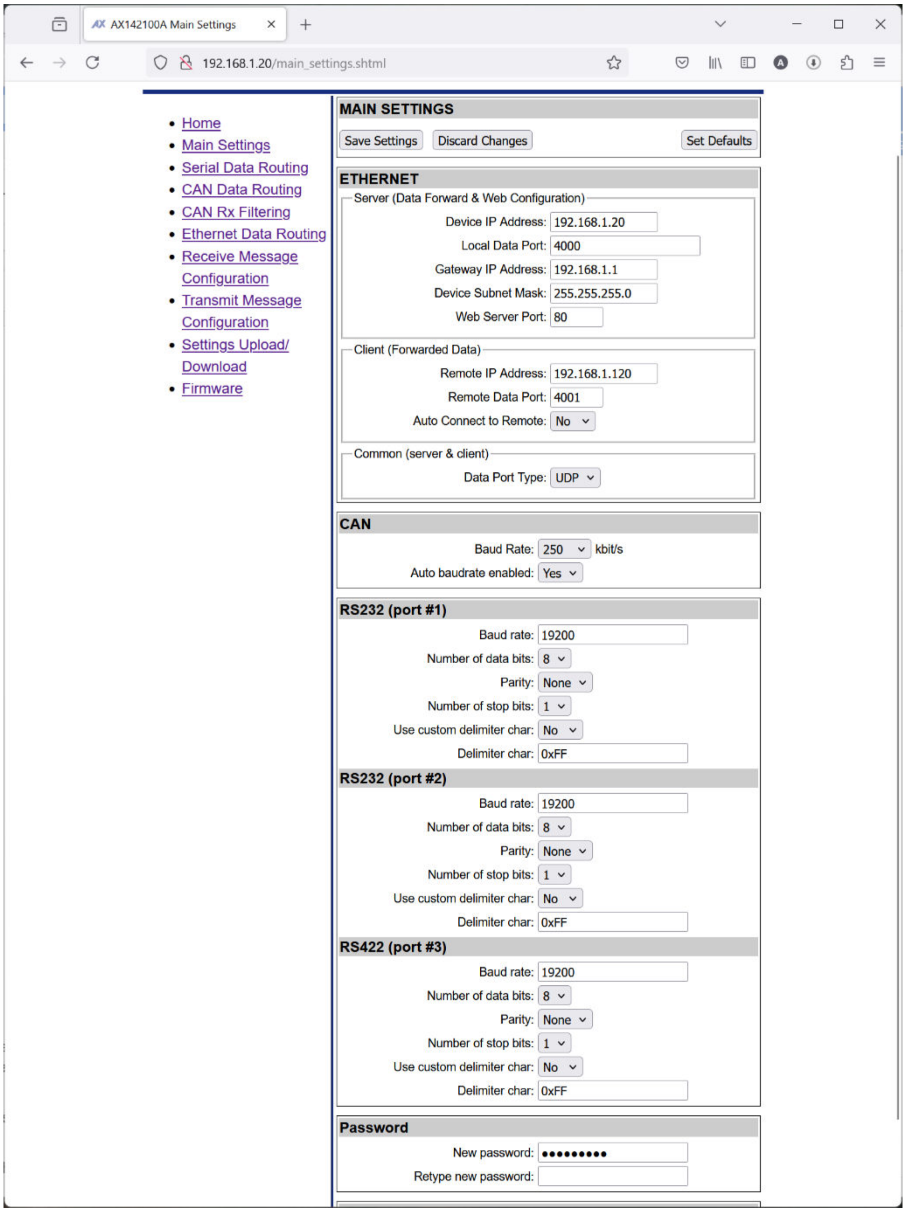
RS232, port #2
Frames Received: 0
Frames Sent: 0
Errors Detected): 0

RS422, port #3
Frames Received: 0
Frames Sent: 0
Errors Detected: 0

Ethernet
Messages Received: 499
Messages Sent: 33

The Home page gives an overview of the main settings and device status information. This page contains no editable settings.

<configured ip>/main_settings.shtml



MAIN SETTINGS

ETHERNET

Server (Data Forward & Web Configuration)

Device IP Address: 192.168.1.20
Local Data Port: 4000
Gateway IP Address: 192.168.1.1
Device Subnet Mask: 255.255.255.0
Web Server Port: 80

Client (Forwarded Data)

Remote IP Address: 192.168.1.120
Remote Data Port: 4001
Auto Connect to Remote: No

Common (server & client)

Data Port Type: UDP

CAN

Baud Rate: 250 kbit/s
Auto baudrate enabled: Yes

RS232 (port #1)

Baud rate: 19200
Number of data bits: 8
Parity: None
Number of stop bits: 1
Use custom delimiter char: No
Delimiter char: 0xFF

RS232 (port #2)

Baud rate: 19200
Number of data bits: 8
Parity: None
Number of stop bits: 1
Use custom delimiter char: No
Delimiter char: 0xFF

RS422 (port #3)

Baud rate: 19200
Number of data bits: 8
Parity: None
Number of stop bits: 1
Use custom delimiter char: No
Delimiter char: 0xFF

Password

New password: •••••••••
Retype new password:

The Main Settings page allows the user to modify the device’s IP address, netmask and the main configuration parameters for the communication interfaces. The CAN configuration parameters include the default baud rate to use and the auto-baud rate capability.

The serial port configuration contains, baud rate (freely settable, allowed range: 1200bps … 256kbps), number of data, start and stop bits and parity.

The serial port configuration also supports custom message delimiter character. By default, only the detected idle condition on the serial interface is considered as a message delimiter. By configuring a customer message delimiter character, messages can be picked up from a continuous serial data stream.

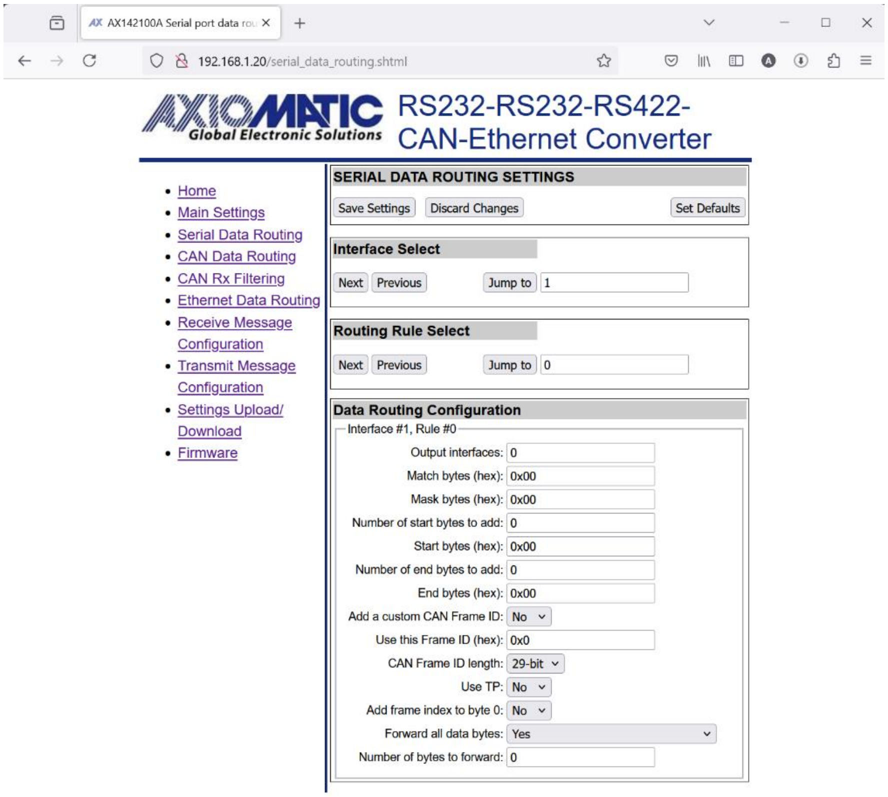
In the settings (see also Table 1 – AX142100A Connector Pinout)

RS Port 1 == RS232, pins 8 & 9 of the CAN / RS232 / RS422 Connector

RS Port 2 == RS232, pins 4 & 5 of the CAN / RS232 / RS422 Connector

RS Port 3 == RS422, pins 1, 2, 3 & 10 of the CAN / RS232 / RS422 Connector

<configured ip>/serial_data_routing.shtml



Interface Select: Jump to 1

Routing Rule Select: Jump to 0

Interface #1, Rule #0

Output interfaces: 0
Match bytes (hex): 0x00
Mask bytes (hex): 0x00
Number of start bytes to add: 0
Start bytes (hex): 0x00
Number of end bytes to add: 0
End bytes (hex): 0x00
Add a custom CAN Frame ID: No
Use this Frame ID (hex): 0x0
CAN Frame ID length: 29-bit
Use TP: No
Add frame index to byte 0: No
Forward all data bytes: Yes
Number of bytes to forward: 0

The data routing configuration is done for each interface separately. The routing is done for all frames received from the three serial ports. Each serial interface supports 3 routing rules.

Each of the rules have a list of output interfaces, match bytes and mask bytes (software filter), add start and end bytes and CAN options, such as add a custom CAN frame ID, use Ext/Std ID and whether to use TP or not. For data forwarded to CAN interface, it is also possible to use CAN message byte 0 as an index.

The **Output interfaces** should be entered as comma separated list with no spaces. Match and Mask Bytes define a software filter for selecting the frames that will be routed to the configured output interfaces.

Start bytes (hex) and **Number of start bytes to add** define the bytes that should be added to the beginning of the forwarded frame.

End bytes (hex) and **Number of end bytes to add** define the bytes that should be added to the end of the forwarded frame.

In case the CAN Interface (interface #4) is among the Output interfaces, the forwarded frames that end up to CAN bus can be configured to have a specific CAN frame ID. In case a CAN frame ID is not defined, the first 29/11 bits (depending on the CAN ID type) will be used as the CAN frame ID.

In case **Use TP** is selected, the forwarded frame will be wrapped to TP frames in case the length exceeds 8 bytes. In case TP is not used, the frame will be sent as multiple single CAN frames. The option to add frame index to byte 0 has an effect only if TP is not used.

The Match and Mask Bytes are applied like this on the received serial port data. In case comparison is true, the data is forwarded:

```
"RX data & mask" == match
```

The **Match bytes (hex)**, **Mask bytes (hex)**, **Number of start bytes to add**, **Start bytes (hex)**, **Number of end bytes to add** and **End bytes (hex)** are applied to the received serial data.

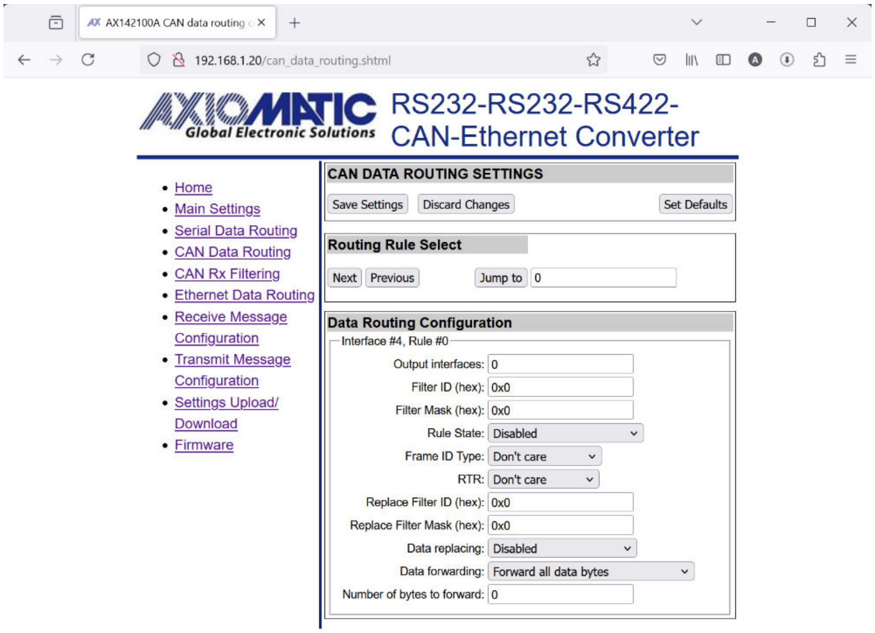
Add a custom CAN Frame ID, **Use this Frame ID (hex)**, **CAN Frame ID length**, **Use TP** and **Add frame index to byte 0** are applied to data that is forwarded to the CAN interface (#4).

Forward all data bytes / **Number of bytes to forward** are applied to all forwarded data.

Please note, that the TP messaging is used only when 29bit CAN frame ID is specified. In case TP is in use, the PGN wrapped inside the TP frame is specified using the **Use this Frame ID (hex)** option.

The **Add frame index to byte 0** option can be used with 11bit frame IDs. This implements “TP like” CAN output.

<configured ip>/can_data_routing.shtml



CAN DATA ROUTING SETTINGS

Routing Rule Select

Jump to: 0

Data Routing Configuration

Interface #4, Rule #0

Output interfaces: 0
Filter ID (hex): 0x0
Filter Mask (hex): 0x0
Rule State: Disabled
Frame ID Type: Don't care
RTR: Don't care
Replace Filter ID (hex): 0x0
Replace Filter Mask (hex): 0x0
Data replacing: Disabled
Data forwarding: Forward all data bytes
Number of bytes to forward: 0

CAN interface supports 16 data routing rules. Each one of the rules has a list of **Output interfaces**, **Filter ID (hex)** and **Filter Mask (hex)** (software filter) and Data replacing options. The data replacing is supported for the CAN Frame ID bits. Also, the number of data bytes to forward can be specified.

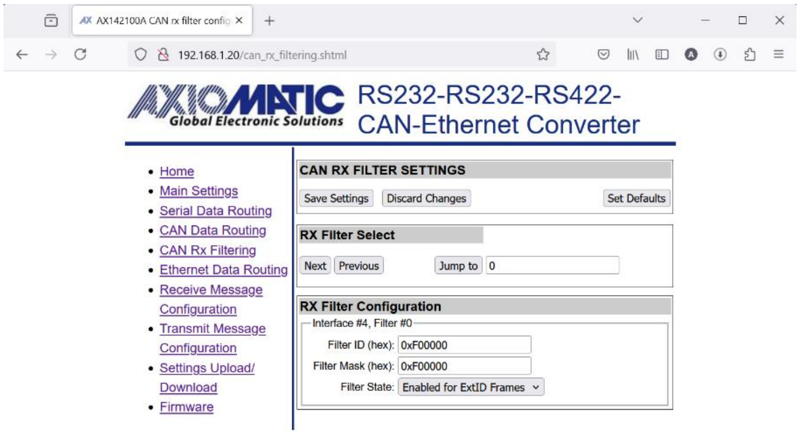
The output interfaces should be entered as comma separated list with no spaces. Filter ID and Mask are identical to the hardware filter configuration, these two settings are used in a software filter for selecting the frames that will be routed to the configured output interfaces.

The **Replace Filter ID (hex)** and **Replace Filter Mask (hex)** can be used for example to modify the Source Address, PGN and/or Priority bits of the J1939 frame. The data replacing function is applied for all frames that pass the software filter and will be done before routing the frame to the configured output interfaces.

To forward all data from frames with a PGN 0xFF01 to interfaces 2 & 3 and modify the forwarded PGN to 0xFF82, the following setup would need to be used:

Output Interfaces: **2,3**
Filter ID (hex): **0xFF0100**
Filter Mask (hex): **0xFFFF00**
Rule State: '**Enabled, use filter&mask**' selected
Frame ID Type: '**Extended (29bit)**' selected
RTR: '**don't care**' selected
Replace Filter ID (hex): **0xFF8200**
Replace Filter Mask (hex): **0xFFFF00**
'Data replacing': '**Use replace filter&mask**' selected
'Data forwarding': '**Forward full message**' selected

<configured ip>/can_rx_filtering.shtml



CAN RX FILTER SETTINGS

RX Filter Select

Jump to: 0

RX Filter Configuration

Interface #4, Filter #0

Filter ID (hex): 0xF00000

Filter Mask (hex): 0xF00000

Filter State: Enabled for ExtID Frames

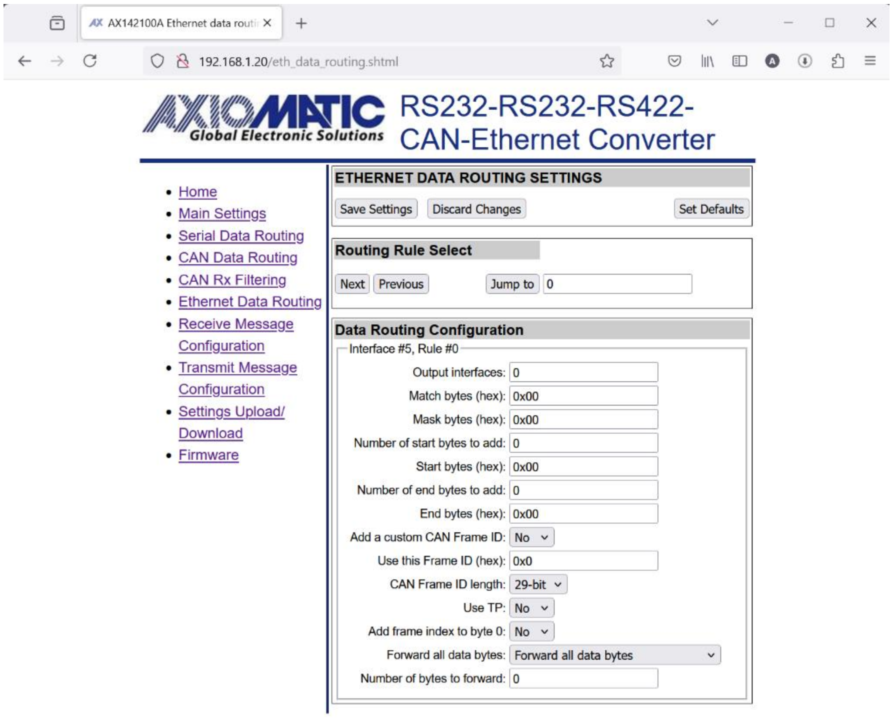
The receive filter is used for selecting which CAN frames will be received. All received CAN frames that pass the reception filter will be forwarded to the data routing module.

The configured CAN ID filter will be assigned to the CAN interface’s acceptance filter registers. No additional software filtering will be done in the message reception. However, the data routing module supports software filtering for selecting the frames that will be routed.

In case all CAN receive filters are disabled, only selected J1939 CAN frames will be accepted. The accepted messages are the ones sent to Global Address (0xFF) and messages sent to 3RS-ENET-CAN’s address, (default 0x80). Successful reception of all other CAN frames requires a custom CAN receive filter to be defined.

The **Filter ID (hex)** defines the 29-bit extended or 11-bit standard frame ID. The Filter Mask bit ‘1’ forces the compare, ‘0’ marks the bit as ‘don’t care’. To configure a filter for receiving all possible frames, the ID and Mask should be set to ‘0’ and **Filter State** should be set to Enabled for both ID types.

<configured ip>/eth_data_routing.shtml


AX142100A Ethernet data routi
192.168.1.20/eth_data_routing.shtml
AXIOMATIC Global Electronic Solutions
Home
Main Settings
Serial Data Routing
CAN Data Routing
CAN Rx Filtering
Ethernet Data Routing
Receive Message Configuration
Transmit Message Configuration
Settings Upload/ Download
Firmware
Save Settings
Discard Changes
Set Defaults
Next
Previous
Jump to

RS232-RS232-RS422-CAN-Ethernet Converter

ETHERNET DATA ROUTING SETTINGS

Routing Rule Select

Jump to: 0

Data Routing Configuration

Interface #5, Rule #0

Output interfaces:	0
Match bytes (hex):	0x00
Mask bytes (hex):	0x00
Number of start bytes to add:	0
Start bytes (hex):	0x00
Number of end bytes to add:	0
End bytes (hex):	0x00
Add a custom CAN Frame ID:	No
Use this Frame ID (hex):	0x0
CAN Frame ID length:	29-bit
Use TP:	No
Add frame index to byte 0:	No
Forward all data bytes:	Forward all data bytes
Number of bytes to forward:	0

The routing rules are applied to all Ethernet frames that are sent to the configured (local) TCP/UDP port. Ethernet interface supports 3 routing rules.

Each of the rules have a list of output interfaces, match bytes and mask bytes (software filter), add start and end bytes and CAN options, such as add a custom CAN frame ID, use Ext/Std ID and whether to use TP or not. For data forwarded to CAN interface, it is also possible to use CAN message byte 0 as an index.

The **Output interfaces** should be entered as comma separated list with no spaces. Match and Mask Bytes define a software filter for selecting the frames that will be routed to the configured output interfaces.

Start bytes (hex) and **Number of start bytes to add** define the bytes that should be added to the beginning of the forwarded frame.

End bytes (hex) and **Number of end bytes to add** define the bytes that should be added to the end of the forwarded frame.

In case the CAN Interface (interface #4) is among the Output interfaces, the forwarded frames that end up to CAN bus can be configured to have a specific CAN frame ID. In case a CAN frame ID is not defined, the first 29/11 bits (depending on the CAN ID type) will be used as the CAN frame ID.

In case **Use TP** is selected, the forwarded frame will be wrapped to TP frames in case the length exceeds 8 bytes. In case TP is not used, the frame will be sent as multiple single CAN frames. The option to add frame index to byte 0 has an effect only if TP is not used.

The Match and Mask Bytes are applied like this on the received Ethernet frame data. In case comparison is true, the data is forwarded:

```
"RX data & mask" == match
```

The **Match bytes (hex)**, **Mask bytes (hex)**, **Number of start bytes to add**, **Start bytes (hex)**, **Number of end bytes to add** and **End bytes (hex)** are applied to the received serial data.

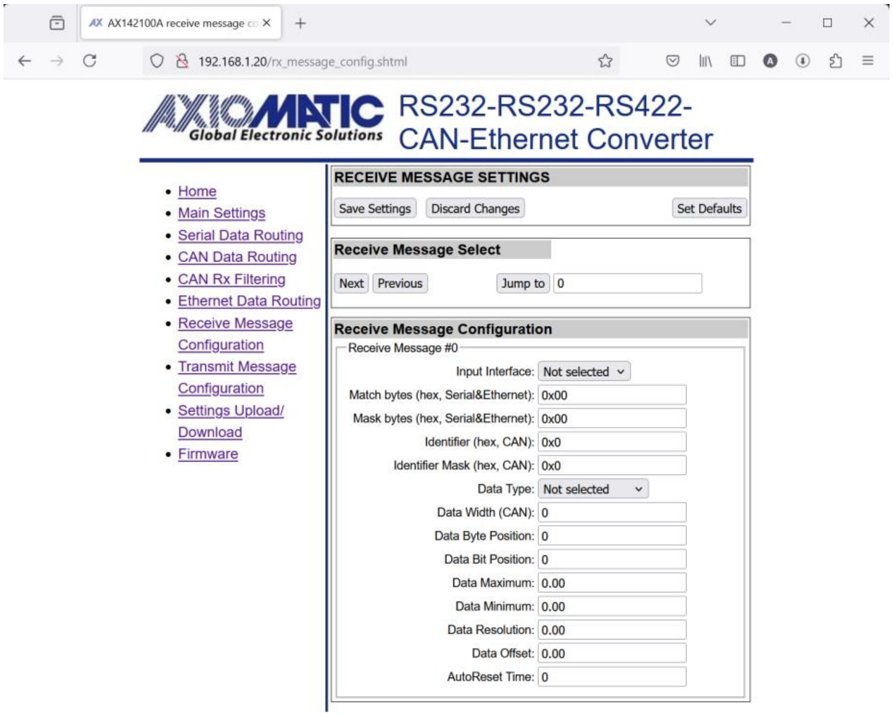
Add a custom CAN Frame ID, **Use this Frame ID (hex)**, **CAN Frame ID length**, **Use TP** and **Add frame index to byte 0** are applied to data that is forwarded to the CAN interface (#4).

Forward all data bytes / **Number of bytes to forward** are applied to all forwarded data.

Please note, that the TP messaging is used only when 29bit CAN frame ID is specified. In case TP is in use, the PGN wrapped inside the TP frame is specified using the **Use this Frame ID (hex)** option.

The **Add frame index to byte 0** option can be used with 11bit frame IDs. This implements “TP like” CAN output.

<configured ip>/rx_message_config.shtml



Jump to: 0

Receive Message #0

Input Interface: Not selected

Match bytes (hex, Serial&Ethernet): 0x00

Mask bytes (hex, Serial&Ethernet): 0x00

Identifier (hex, CAN): 0x0

Identifier Mask (hex, CAN): 0x0

Data Type: Not selected

Data Width (CAN): 0

Data Byte Position: 0

Data Bit Position: 0

Data Maximum: 0.00

Data Minimum: 0.00

Data Resolution: 0.00

Data Offset: 0.00

AutoReset Time: 0

A value from a received frame, such as GPS data, can be parsed using the configuration options available in the Receive Message Configuration page. 3RS-ENET-CAN converter supports 4 serial message configurations.

Match bytes (hex, Serial&Ethernet) define the start of the serial message that should be parsed from the serial data stream. **Mask bytes (hex, Serial&Ethernet)** set the mask that will be used in the Match bytes detection.

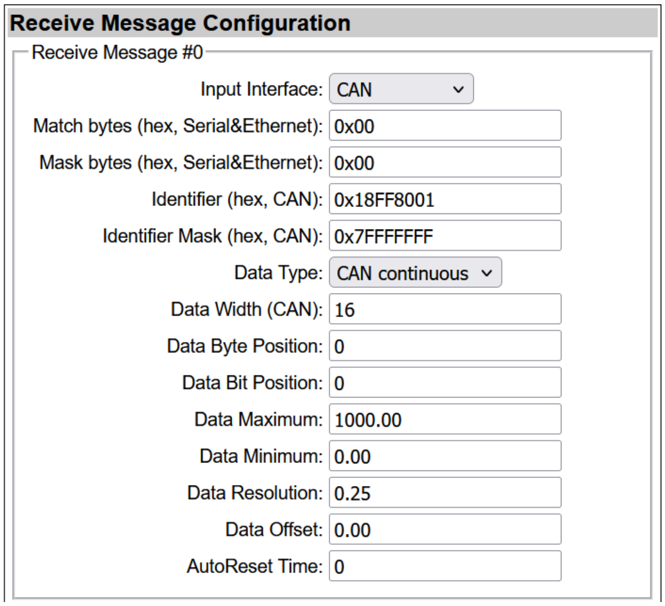
When reading CAN data, the **Identifier (hex, CAN)** and **Identifier Mask (hex, CAN)** define the rules for checking the received CAN frames.

The data type to be parsed is selected from the **Data Type** drop down menu.

Configuration and data range for the data to be parsed is defined in the **Data Width (CAN)**, **Data Byte Position**, **Data Bit Position**, **Data Maximum**, **Data Minimum**, **Data Resolution** and **Data Offset**.

If the received data needs to expire after a certain time, this can be defined using the **AutoReset Time**.

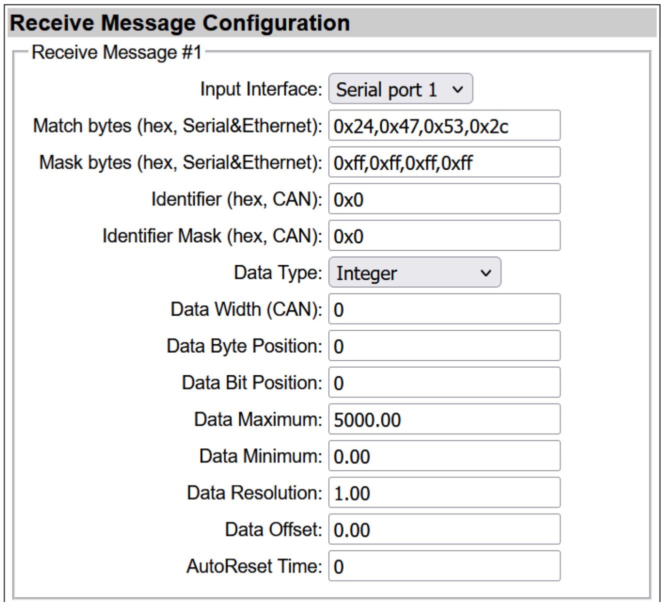
Receive message configuration examples



Receive Message Configuration

Receive Message #0

Input Interface:	CAN
Match bytes (hex, Serial&Ethernet):	0x00
Mask bytes (hex, Serial&Ethernet):	0x00
Identifier (hex, CAN):	0x18FF8001
Identifier Mask (hex, CAN):	0x7FFFFFFF
Data Type:	CAN continuous
Data Width (CAN):	16
Data Byte Position:	0
Data Bit Position:	0
Data Maximum:	1000.00
Data Minimum:	0.00
Data Resolution:	0.25
Data Offset:	0.00
AutoReset Time:	0



Receive Message Configuration

Receive Message #1

Input Interface:	Serial port 1
Match bytes (hex, Serial&Ethernet):	0x24,0x47,0x53,0x2c
Mask bytes (hex, Serial&Ethernet):	0xff,0xff,0xff,0xff
Identifier (hex, CAN):	0x0
Identifier Mask (hex, CAN):	0x0
Data Type:	Integer
Data Width (CAN):	0
Data Byte Position:	0
Data Bit Position:	0
Data Maximum:	5000.00
Data Minimum:	0.00
Data Resolution:	1.00
Data Offset:	0.00
AutoReset Time:	0

The above configuration (on the left) reads in a CAN frame with J1939 PGN 0xFF80. The CAN frame is defined using the full ID and a mask that requires that all bits in the CAN frame need to match the configured ID.

The CAN data is 16 bits wide, and parsing starts from CAN payload byte 0, bit 0. The maximum value for the data is 1000.00 (parsed value) and the resolution to use when parsing CAN data is 0.25 units per bit. **Data Type** *CAN continuous* defines that the maximum, minimum, offset and resolution settings are applied. With *CAN discrete* data, the offset and resolution are not applied, and the maximum value is set by the **Data Width (CAN)**.

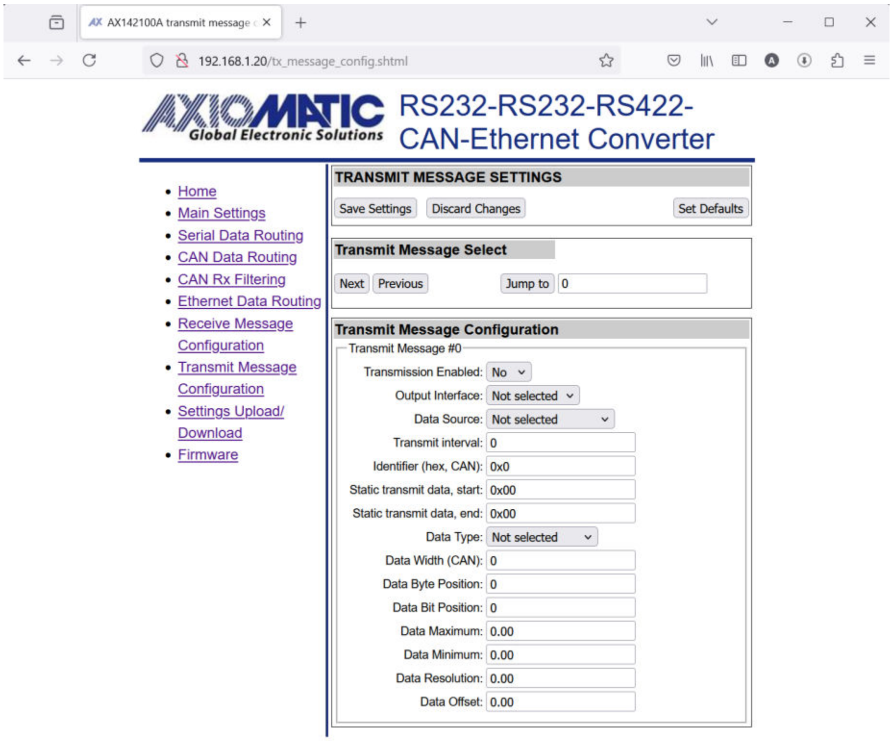
Since the **AutoReset Time** is *0*, the received data will be valid until the next CAN frame is received or the 3RS-ENET-CAN converter's power is cycled.

On the right side, an example Receive Message Configuration for serial data parsing is shown. This configuration defines that a serial message with the first four bytes *0x24, 0x47, 0x53, 0x2C* (`$GS,x\r\n` in ascii) shall be read in (in which x is the value to parse). The Mask bytes define that all four bytes need to match fully.

The data type is *Integer*, and the maximum value is 5000. **Data Resolution** is set to *1*, so the parsed value is converted to Integer type with no additional scaling.

The data is parsed starting from the first byte following the configured **Match bytes (hex, Serial&Ethernet)**.

<configured ip>/tx_message_config.shtml



TRANSMIT MESSAGE SETTINGS

Transmit Message Select

Jump to: 0

Transmit Message Configuration

Transmit Message #0

Transmission Enabled:	No
Output Interface:	Not selected
Data Source:	Not selected
Transmit interval:	0
Identifier (hex, CAN):	0x0
Static transmit data, start:	0x00
Static transmit data, end:	0x00
Data Type:	Not selected
Data Width (CAN):	0
Data Byte Position:	0
Data Bit Position:	0
Data Maximum:	0.00
Data Minimum:	0.00
Data Resolution:	0.00
Data Offset:	0.00

A periodically transmitted data message can be configured using the configuration options on the Transmit Message Configuration page. The message can be sent to all communication interfaces on the 3RS-ENET-CAN converter. 3RS-ENET-CAN converter supports 4 transmit messages.

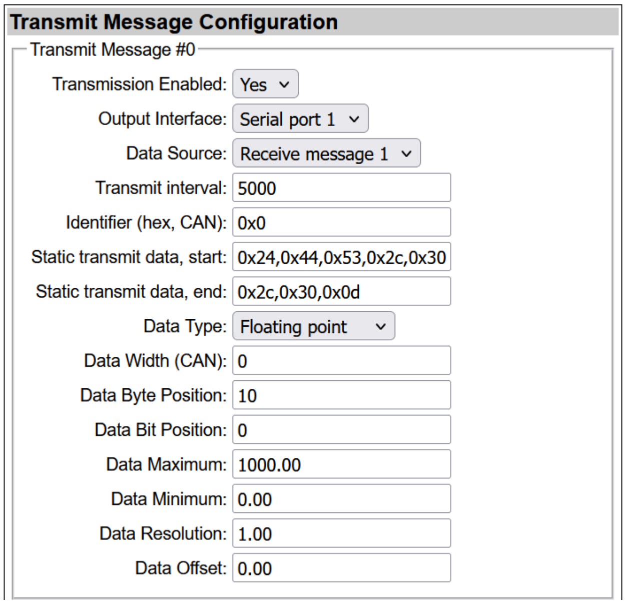
Output Interface lists all supported interfaces for sending the message. **Data Source** selects the source for the data to be included into the transmit message. **Transmit Interval** defines the periodic transmission interval in milliseconds.

Identifier (hex, CAN) is the CAN frame ID to use. This will be applied only to messages that are transmitted to CAN.

Static transmit data, start defines the static bytes to add to a serial/ethernet message. These bytes are added before the variable data field. **Static transmit data, end** defines the bytes to add after the variable data field.

Data Type, **Data Width (CAN)**, **Data Byte Position**, **Data Bit Position**, **Data Maximum**, **Data Minimum**, **Data Resolution** and **Data Offset** configure the variable data, inserted between the static start and end bytes or to a CAN frame.

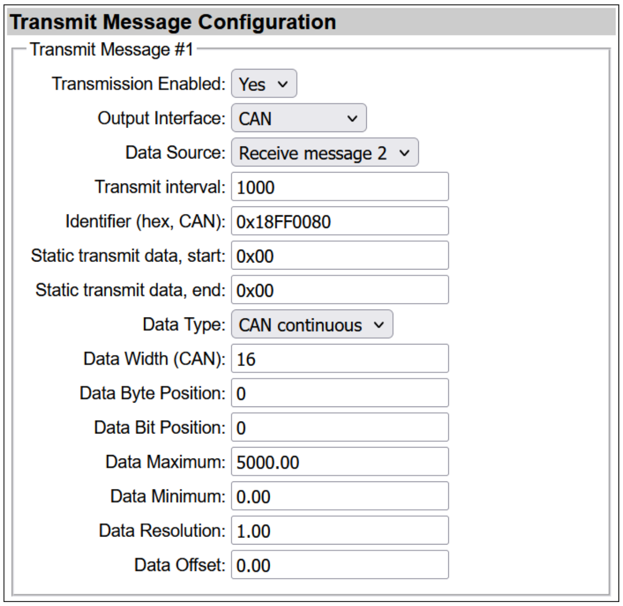
Transmit message configuration examples



Transmit Message Configuration

Transmit Message #0

Transmission Enabled:	Yes
Output Interface:	Serial port 1
Data Source:	Receive message 1
Transmit interval:	5000
Identifier (hex, CAN):	0x0
Static transmit data, start:	0x24,0x44,0x53,0x2c,0x30
Static transmit data, end:	0x2c,0x30,0x0d
Data Type:	Floating point
Data Width (CAN):	0
Data Byte Position:	10
Data Bit Position:	0
Data Maximum:	1000.00
Data Minimum:	0.00
Data Resolution:	1.00
Data Offset:	0.00



Transmit Message Configuration

Transmit Message #1

Transmission Enabled:	Yes
Output Interface:	CAN
Data Source:	Receive message 2
Transmit interval:	1000
Identifier (hex, CAN):	0x18FF0080
Static transmit data, start:	0x00
Static transmit data, end:	0x00
Data Type:	CAN continuous
Data Width (CAN):	16
Data Byte Position:	0
Data Bit Position:	0
Data Maximum:	5000.00
Data Minimum:	0.00
Data Resolution:	1.00
Data Offset:	0.00

The above configuration (on the left) transmits a message `$DS,0,0,0,x,0\n` (in which x is the variable data field) into the serial port #1.

The **Data Source** for the variable data field is *CAN Receive Message #1* and the message is transmitted every 5000ms.

The complete list of **Static bytes, start** is *0x24,0x44,0x53,0x2c,0x30,0x2c,0x30,0x2c,0x30,0x2c*, which is `$DS,0,0,0,` in ascii. The **Static bytes, end** is *0x2c,0x30,0x0d*. This is `,0\n` in ascii.

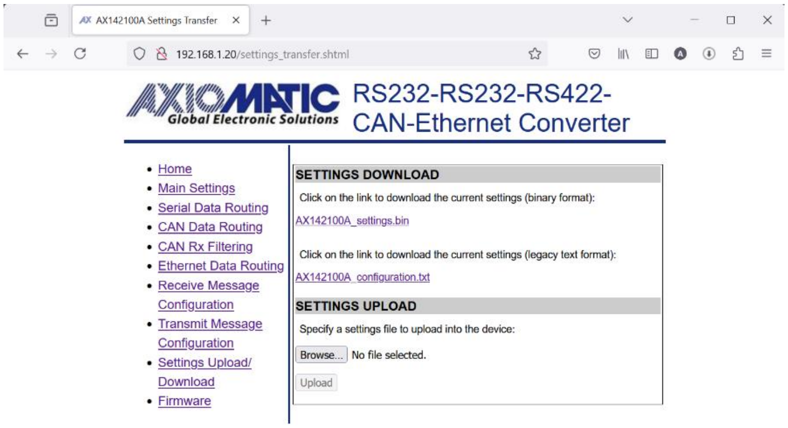
The **Data Type** to insert is *Floating point* and the data should be inserted starting from byte 10 in the transmit message. Please note that the **Static bytes, start** need to contain enough bytes to be added to the message before the variable data field, otherwise the transmit message gets inadvertently truncated by null data. Maximum value for the data field is 1000.00 and the data from Receive Message #1 is not scaled (**Data Resolution** is *1.00*).

On the right side, a CAN transmit message is configured. This message uses data from *Receive Message #2* and is sent at 1000ms intervals.

Identifier (hex, CAN) defines the CAN ID to use. **Static transmit data, start** and **Static transmit data, end** are not configured since those settings are not used in CAN transmit messages.

The **Data Type** *CAN continuous* defines that the data in CAN payload bytes should be scaled using **Data Maximum**, **Data Minimum**, **Data Resolution** and **Data Offset** settings. **Data Byte Position** and **Data Bit Position** settings define the data location in the CAN payload data bytes.

<configured ip>/settings_transfer.shtml


AX142100A Settings Transfer
192.168.1.20/settings_transfer.shtml
AXIOMATIC Global Electronic Solutions
RS232-RS232-RS422-
CAN-Ethernet Converter
Home
Main Settings
Serial Data Routing
CAN Data Routing
CAN Rx Filtering
Ethernet Data Routing
Receive Message Configuration
Transmit Message Configuration
Settings Upload/ Download
Firmware
Browse...
Upload

SETTINGS DOWNLOAD

Click on the link to download the current settings (binary format):

AX142100A_settings.bin

Click on the link to download the current settings (legacy text format):

AX142100A_configuration.txt

SETTINGS UPLOAD

Specify a settings file to upload into the device:

No file selected.

The AX142100A supports settings upload and download using the legacy text and binary format. The settings can be downloaded from the AX142100A by using the corresponding link on the Settings Upload/Download page.

The settings upload function opens a dialog for selecting a previously saved settings file. Both types of settings, text and binary can be selected.

<configured ip>/fullconfig

The 3RS-ENET-CAN supports the use of cURL (or equivalent) for full settings file download and upload in the legacy text format. This is an alternative method for the method found on the ‘Settings Upload/Download’ page.

Please note that to access the configuration, the correct password needs to be entered first.

The current configuration can be downloaded to PC using command:

```
curl -o "./config.file" "http://192.168.1.20/fullconfig"
```

The saved configuration can be uploaded to the 3RS-ENET-CAN device:

```
curl --upload-file "./config.file" "http://192.168.1.20/fullconfig"
```

Note, that cURL configuration upload and download are supported for backwards compatibility purposes only. cURL won’t return meaningful status after successful data transfer, in most cases only the status and message

```
curl: (52) Empty reply from server
```

or similar is shown.

4.2. TCP/UDP Connections

The forwarded frames can be sent as proprietary TCP or UDP frames. A client can listen to these frames by initiating a TCP (or UDP) connection to port 4000 (or to custom port, configured using EA or a web browser) on the 3RS-ENET-CAN. These forwarded data messages are sent when the data become available from serial ports or from CAN interface

The TCP/UDP port can be written to, the received frames will be forwarded to output interfaces specified on the routing configuration page #5.

The *Message Header* contains:

4-byte *Axiomatic Tag,* AXIO in capital letters

2-byte *Protocol ID*, 20008 = 0x4E28

2-byte *Message ID*

1-byte *Message Version*, 0 (for future use)

2-byte *Message Data Length*

The proprietary messaging protocol *Message Header* format is presented below.

<table>
<tr><th><i>Octet</i></th><th rowspan="2"><i>0</i></th><th rowspan="2"><i>1</i></th><th rowspan="2"><i>2</i></th><th rowspan="2"><i>3</i></th></tr>
<tr><th><i>Offset Octet</i></th></tr>
<tr><td rowspan="3"><i>0</i></td><td><b>A</b></td><td><b>X</b></td><td><b>I</b></td><td><b>O</b></td></tr>
<tr><td>0x41</td><td>0x58</td><td>0x49</td><td>0x4F</td></tr>
<tr><td colspan="4">Axiomatic Tag</td></tr>
<tr><td rowspan="2"><i>4</i></td><td><b>0x28</b></td><td><b>0x4E</b></td><td colspan="2" rowspan="2">Message ID</td></tr>
<tr><td colspan="2">Protocol ID (20008)</td></tr>
<tr><td rowspan="2"><i>8</i></td><td><b>0x00</b></td><td colspan="2" rowspan="2">Message Data Length</td><td rowspan="2">Message Data</td></tr>
<tr><td>Message Version=0</td></tr>
</table>

Table 2 – TCP message header format

The *Axiomatic Tag* is used for the message header identification.

The *Protocol ID* defines a proprietary protocol carried by this message. This field allows different protocols to use the same protocol independent message structure. The AX142100A uses Protocol ID = 0x4E28

The *Message ID* defines the type of the Message Data:

Message ID	Message name
0	Undefined message
1	Forwarded data

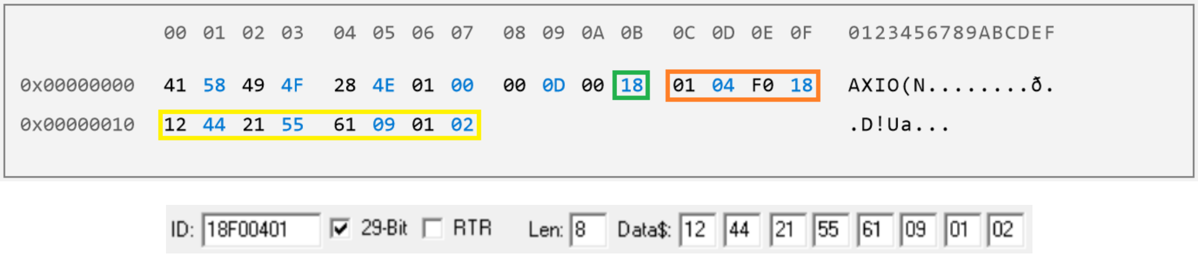
The first byte of the payload data in the Ethernet frame contains status bits that control how the AX142100A handles the received Ethernet data. In case the Raw data flag is set, all following bytes are considered as data with no special formatting.

<first byte> & 0x40 == Raw data

In case the Raw data flag is not set, the data bytes are considered as CAN data

<first byte> & 0x10 == 0x10 -> 29bit CAN frame ID
<first byte> & 0x10 == 0x00 -> 11bit CAN frame ID
<first byte> & 0x0F == CAN data length

An example printout of the TCP client (see also Table 3 – Example TCP client implementation) reveals the TCP frame contents when a CAN frame is forwarded to Ethernet.



```
            00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F  0123456789ABCDEF

0x00000000  41 58 49 4F  28 4E 01 00  00 0D 00 18  01 04 F0 18  AXIO(N........ð.
0x00000010  12 44 21 55  61 09 01 02                            .D!Ua...
```

ID: 18F00401 ☑ 29-Bit □ RTR Len: 8 Data$: 12 44 21 55 61 09 01 02

Figure 3 – TCP/IP frame contents vs. CAN data

The frame starts with the header bytes described in Table 2. After the header, the first byte of the payload data is marked with green. It contains a flag that the CAN frame has 29bit ID and 8 data bytes. The CAN frame ID is marked with orange and CAN data bytes with yellow.

Please note that the AX142100A considers the Ethernet frames like the one above as CAN data (no Raw data flag set). When this data is forwarded to the CAN interface, the frame ID type (29bit/11bit) and the ID bytes are automatically picked up from the Ethernet frame.

On the other hand, when the Ethernet frame contains Raw data, the data is forwarded ‘as is’ and no special processing is applied (other than the routing rules defined for Ethernet data).

```c
#include <winsock2.h>
#include <Ws2tcpip.h>
#include <stdio.h>

#define DEFAULT_BUFLEN        256

#define IP_ADDRESS            "192.168.1.20"
#define FWD_DATA_PORT         4000

#define dRAW_DATA_FLAG        0x40


int main(void) {

    int iResult, index;
    WSADATA wsaData;

    SOCKET ConnectSocket = INVALID_SOCKET;
    struct sockaddr_in clientService;

    int recvbuflen = DEFAULT_BUFLEN;
    char recvbuf[DEFAULT_BUFLEN];

    // Initialize Winsock
    iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
    if (iResult != NO_ERROR) {
        printf("WSAStartup failed with error: %d\n", iResult);
        return 1;
    }

    // Create a socket for connecting to the AX142100
    ConnectSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (ConnectSocket == INVALID_SOCKET) {
        printf("socket failed with error: %d\n", WSAGetLastError());
        WSACleanup();
        return 1;
    }

    clientService.sin_family = AF_INET;
    clientService.sin_addr.s_addr = inet_addr( IP_ADDRESS );
    clientService.sin_port = htons( FWD_DATA_PORT );

    // Connect to the AX142100
    iResult = connect( ConnectSocket, (SOCKADDR*) &clientService, sizeof(clientService) );
    if (iResult == SOCKET_ERROR) {
        printf("connect failed with error: %d\n", WSAGetLastError() );
        closesocket(ConnectSocket);
        WSACleanup();
        return 1;
    }

    // Receive data until the AX142100 closes the connection
    do {
        memset((void *)recvbuf, 0x00, sizeof(recvbuf));
        iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);
        if ( iResult > 0 )
        {
            printf("Bytes received: %d (msg data in hex below)\n", iResult);
            for ( index = 1; index < iResult+1; index++ )
            {
                printf("%02X ", (unsigned char)recvbuf[index-1]);
                if ( (index % 8) == 0 ) printf("\n");
            }
            printf("\n");

            // Send back with ID + 1 (Frame ID is in indexes 12 ... 15)
            if( recvbuf[12] < 255 ) recvbuf[12]++;

            #if 0
            // Send back as raw data instead of CAN data
            recvbuf[11] |= dRAW_DATA_FLAG;
            printf("message flagged as raw data.\n");
            #endif

            iResult = send( ConnectSocket, recvbuf, iResult, 0 );
            if (iResult == SOCKET_ERROR) {
                printf("send failed with error: %d\n", WSAGetLastError());
                closesocket(ConnectSocket);
                WSACleanup();
                return 1;
            }

            printf("%d bytes sent back\n", iResult);
        }
        else if ( iResult == 0 )
            printf("Connection closed\n");
        else
            printf("recv failed with error: %d\n", WSAGetLastError());

    } while( iResult > 0 );

    return 0;
}
```

Table 3 – Example TCP client implementation

The example can be compiled using MinGW: <MinGW location>\bin\gcc.exe -Wall -o data_client data_client.c -lws2_32

5. ECU SETPOINTS ACCESSED WITH AXIOMATIC ELECTRONIC ASSISTANT

This section describes in detail each setpoint, and their default and ranges. The setpoints are divided into setpoint groups as they are shown in EA. For more information on how each setpoint is used by 3RS-ENET-CAN, refer to the relevant section in this user manual.

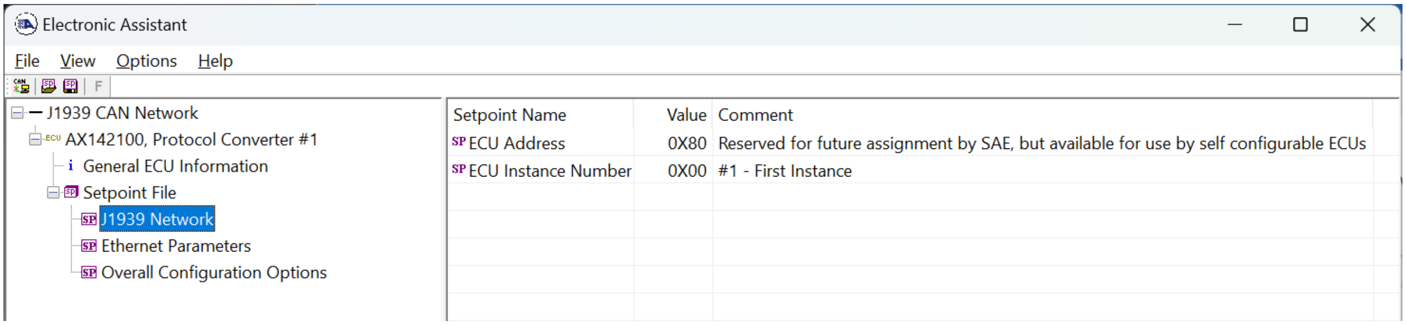
5.1. J1939 Setpoints

"**ECU Instance Number**" and "**ECU Address**" setpoints and their effect are defined in section 3.2.

Name	Range	Default	Notes
ECU Instance Number	0-7	0x00	Per J1939-81
ECU Address	0-253	0x80	Preferred address for a self-configurable ECU

Table 4 – J1939 Setpoints

If non-default values for the **"ECU Instance Number"** or **"ECU Address"** are used, they will be mirrored during a setpoint file flashing, and will only take effect once the entire file has been downloaded to the unit. After the setpoint flashing is complete, the unit will claim the new address and/or re-claim the address with the new NAME. If these setpoints are changing, it is recommended to close and re-open the CAN connection on EA after the file is loaded so that only the new NAME and address are showing in the J1939 CAN Network ECU list.

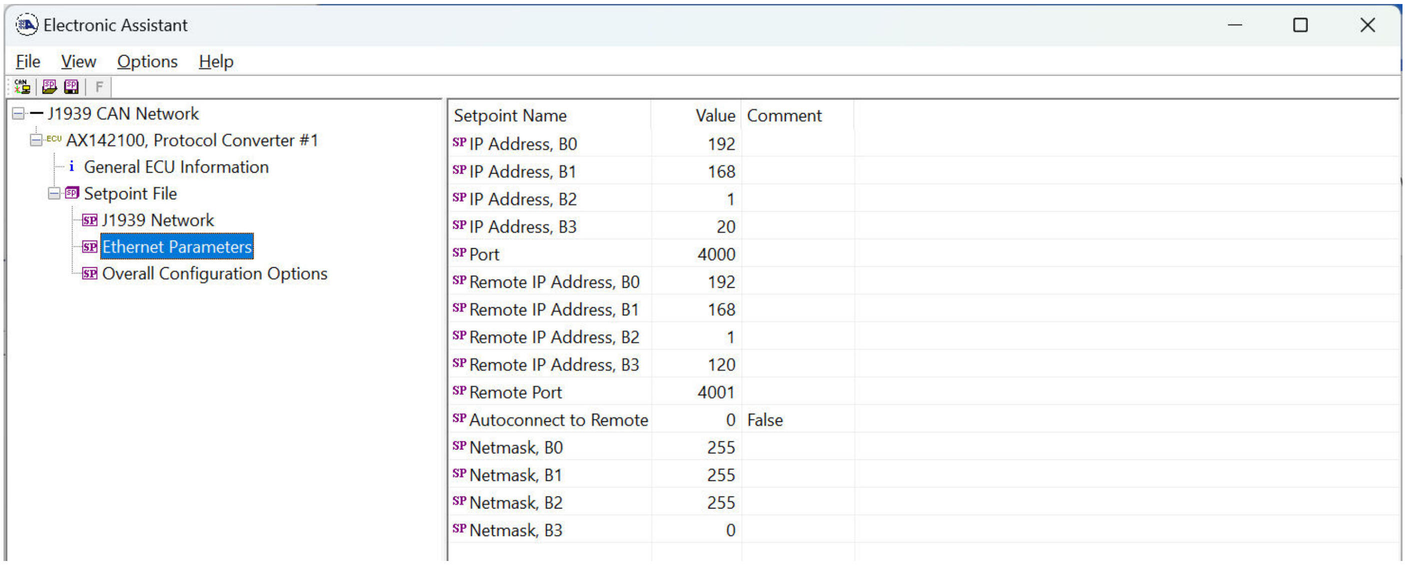


Setpoint Name	Value	Comment
ECU Address	0X80	Reserved for future assignment by SAE, but available for use by self configurable ECUs
ECU Instance Number	0X00	#1 - First Instance

Figure 4 – Screen Capture of J1939 Network Setpoints

5.2. Ethernet Parameter Setpoints

The Ethernet parameters can be configured using EA. A power cycle is needed for taking the new network settings in use.



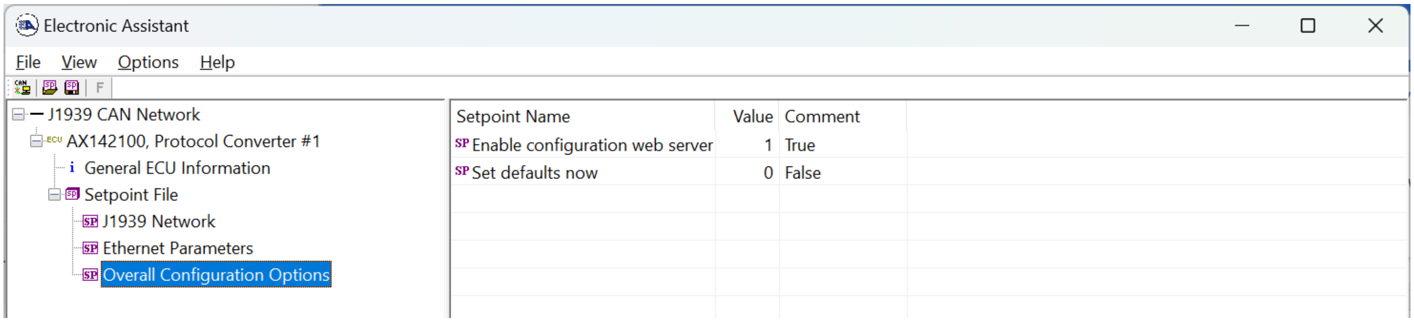
Setpoint Name	Value	Comment
SP IP Address, B0	192	
SP IP Address, B1	168	
SP IP Address, B2	1	
SP IP Address, B3	20	
SP Port	4000	
SP Remote IP Address, B0	192	
SP Remote IP Address, B1	168	
SP Remote IP Address, B2	1	
SP Remote IP Address, B3	120	
SP Remote Port	4001	
SP Autoconnect to Remote	0	False
SP Netmask, B0	255	
SP Netmask, B1	255	
SP Netmask, B2	255	
SP Netmask, B3	0	

Figure 5 – Screen Capture of Ethernet Parameter Setpoints

Name	Range	Default	Notes
IP Address, B0	0…255	192	These settings define an IP address: 192.168.1.20
IP Address, B1	0…255	168	
IP Address, B2	0…255	1	
IP Address, B3	0…255	20	
Port	0…65535	4000	Default port for incoming TCP connections
Remote IP Address, B0	0…255	192	These settings define an IP address for remote connection: 192.168.1.120
Remote IP Address, B1	0…255	168	
Remote IP Address, B2	0…255	1	
Remote IP Address, B3	0…255	120	
Remote Port	0…65535	4001	Default port for remote TCP connection
Autoconnect to Remote	0, 1	0 – False	Whether to automatically initiate remote TCP/UDP connection
Netmask, B0	0…255	255	These settings define a netmask 255.255.255.0
Netmask, B1	0…255	255	
Netmask, B2	0…255	255	
Netmask, B3	0…255	0	

Table 5 – Ethernet Parameter Setpoints

5.3. Overall Configuration Options



Setpoint Name	Value	Comment
Enable configuration web server	1	True
Set defaults now	0	False

Figure 6 – Screen Capture of Overall Configuration Options Setpoints

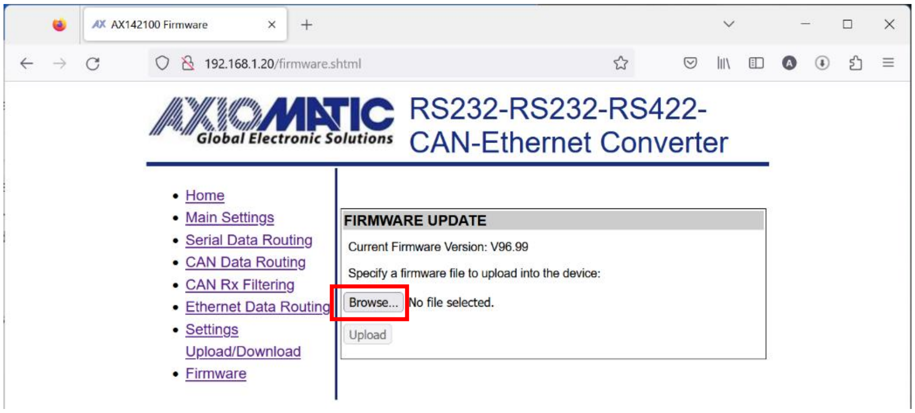
Name	Range	Default	Notes
Enable configuration web server	0, 1	1 – True	Configuration web server running on port 80 (TCP)
Set defaults now	0, 1	0 – False	This setpoint is password protected. The password is '**SetDefaults**'.

Table 6 – Overall Configuration Options Setpoints

6. REFLASHING OVER ETHERNET USING A WEB BROWSER

The AX142100A can be upgraded with new application firmware using a web browser. Once the correct configuration password is entered, the firmware reflash can be done using the 'Firmware' page.

<configured ip>/firmware.shtml



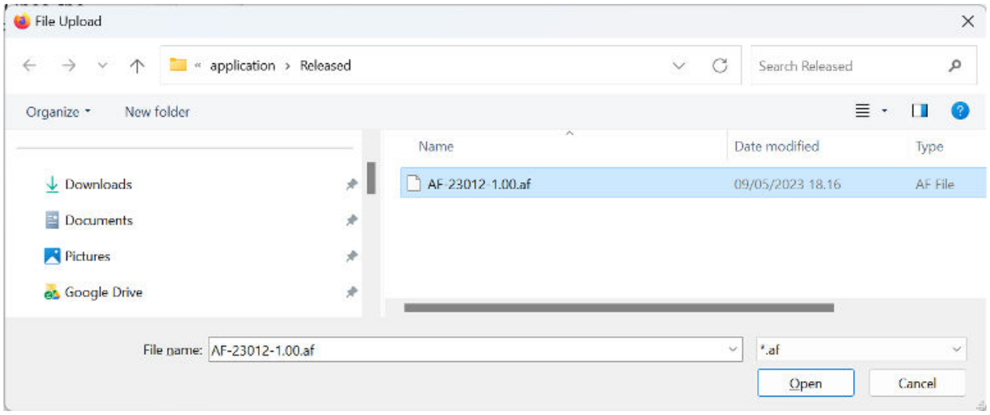
FIRMWARE UPDATE

Current Firmware Version: V96.99

Specify a firmware file to upload into the device:

No file selected.

On the 'Firmware' page, a file selection dialog can be opened by pressing the 'Browse…' button.

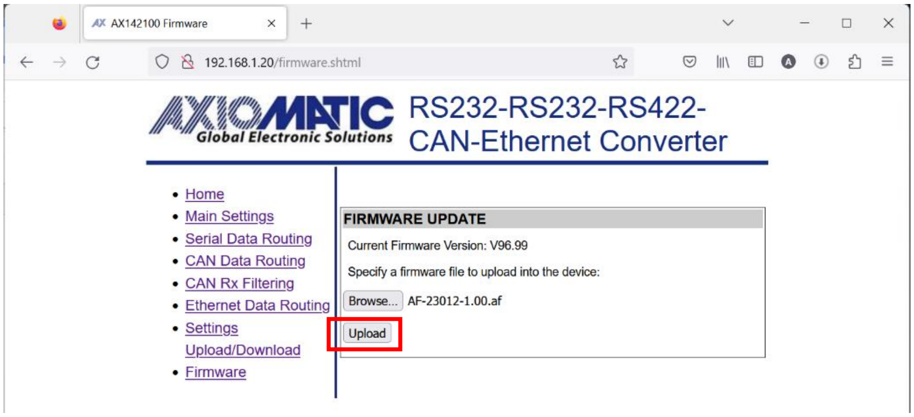


Name	Date modified	Type
AF-23012-1.00.af	09/05/2023 18:16	AF File

File name: AF-23012-1.00.af

*.af

Navigate to where you had saved the **AF-23012-x.xx.af** file sent from Axiomatic. (Note: only binary (.af) files can be flashed using the web browser firmware update interface.)



RS232-RS232-RS422-CAN-Ethernet Converter

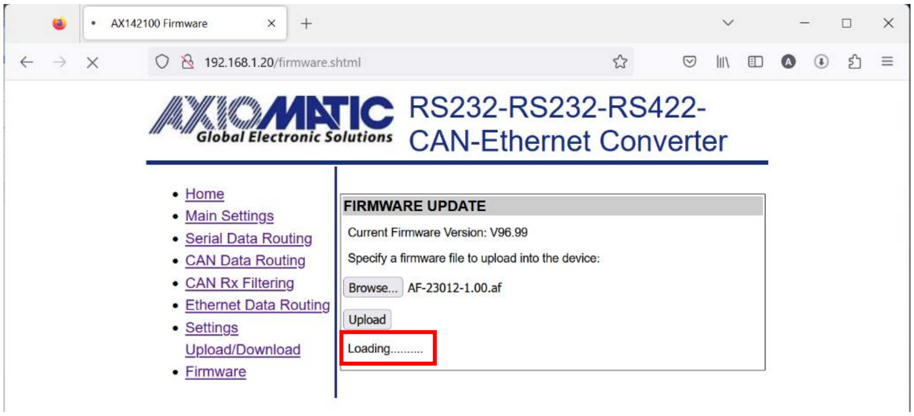
FIRMWARE UPDATE

Current Firmware Version: V96.99

Specify a firmware file to upload into the device:

AF-23012-1.00.af

Once the file is selected, the actual upload/upgrade process is started by pressing the ‘Upload’ button.



RS232-RS232-RS422-CAN-Ethernet Converter

FIRMWARE UPDATE

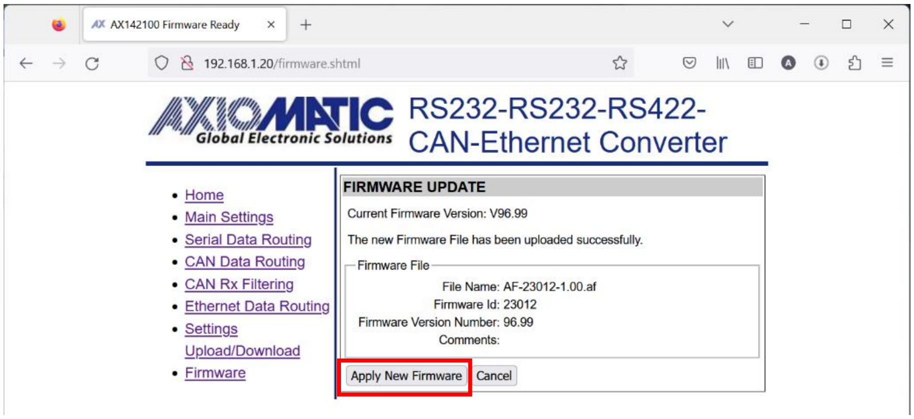
Current Firmware Version: V96.99

Specify a firmware file to upload into the device:

AF-23012-1.00.af

Loading..........

The firmware upload process is shown below the ‘Upload’ button.



RS232-RS232-RS422-CAN-Ethernet Converter

FIRMWARE UPDATE

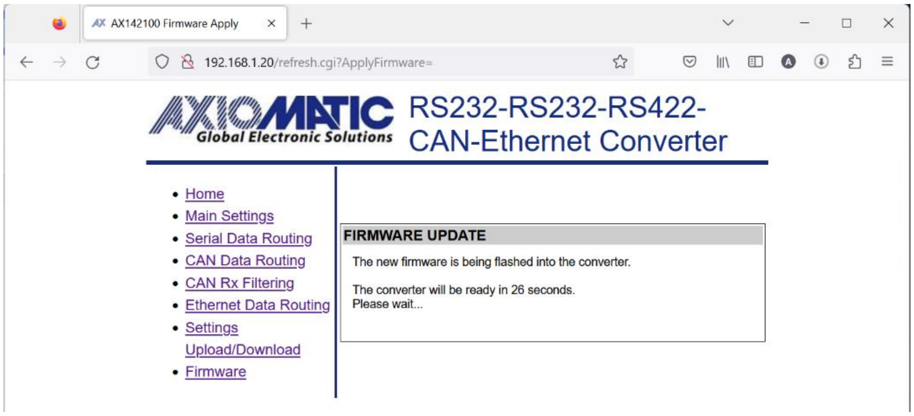
Current Firmware Version: V96.99

The new Firmware File has been uploaded successfully.

Firmware File

File Name: AF-23012-1.00.af
Firmware Id: 23012
Firmware Version Number: 96.99
Comments:

Once the upload is finished and the file checked and stored to a temporary location on the AX142100A, the user is prompted to either to ‘Apply New Firmware’ or cancel the operation.



FIRMWARE UPDATE

The new firmware is being flashed into the converter.

The converter will be ready in 26 seconds.
Please wait...

The firmware reflash procedure takes 30 seconds to finish. After this the AX142100A reboots automatically to the new firmware and returns to the password dialog.

APPENDIX A - TECHNICAL SPECIFICATION

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/.

All specifications are typical at nominal input voltage and 25°C unless otherwise specified.

Power

Power Supply Input	12 or 24 VDC nominal (9 to 36 VDC)
Quiescent Current	150 mA @ 12 V; 90 mA @ 24 V typical
Surge Protection	95 VDC
Under-Voltage Protection	Hardware shutdown at 6 VDC
Over-Voltage Protection	Hardware shutdown at 45 VDC
Reverse Polarity Protection	Provided up to -36 VDC

Functionality

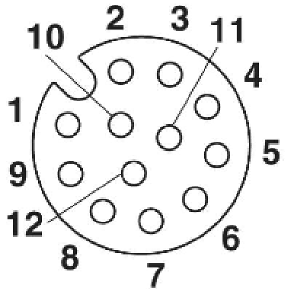
<table>
<tr><td>Conversion Platform</td><td>The Protocol Converter comes pre-programmed with standard protocol conversion logic for bidirectional data exchange between an Ethernet (proprietary TCP communications), an RS-422 bus, two RS-232 buses and a CAN network (SAE J1939).

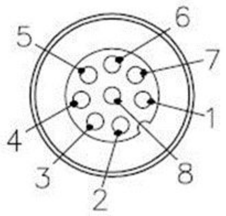
Data is forwarded “as-is” between the different serial ports. Also, CAN/Ethernet data is forwarded directly to serial interfaces with the configuration allowing the user to specify the CAN message ID (or TCP port) to listen for data to be forwarded.</td></tr>
<tr><td>Ethernet</td><td>1 10/100 Mbit Ethernet compliant port
10BASE-T, 100BASE-Tx (auto-negotiation and full-duplex supported)
Auto-MDIX</td></tr>
<tr><td>RS-422</td><td>1 RS-422 port
Baud rate: up to 10.5 Mbit/s
Note: RS-422 connections can be used as RS-485. See pinout.</td></tr>
<tr><td>RS-232</td><td>2 RS-232 ports for serial communications
Three-wire
Baud rate: up to 400 kbit/s</td></tr>
<tr><td>ASCII Features</td><td><table>
<tr><td>Maximum Number of ASCII devices</td><td>2</td></tr>
<tr><td>Serial Communications Port 0</td><td>RS422</td></tr>
<tr><td>Serial Communications Port 1</td><td>RS232</td></tr>
<tr><td>Message Queue Size</td><td>Configurable</td></tr>
</table></td></tr>
<tr><td>CAN</td><td>1 SAE J1939 port
Isolated
Baud rate: 250 kbit/s (default)</td></tr>
</table>

General Specifications

Microcontroller	STM32H723VGT 32-bit, 1 MB flash memory
Isolation	CAN isolation: 330 Vrms
Web Interface	Refer to the User Manual. The functionality of the web interface includes but is not limited to the following. Specify CAN message filters and CAN message IDs to be received Link RS-232 or RS-422 to CAN bus and Ethernet Define CAN node ID and baud rate Define Ethernet parameters (IP address, netmask) Configure message queues
User Interface	Axiomatic Electronic Assistant (P/N: **AX070502** or **AX070506K**) 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 IP address configuration and firmware reflashing.
LED Indicators	Power LED GREEN = Power ON RED = Fault condition GREEN/RED = Power OFF 2 GREEN LEDs for Ethernet LINK/ACT: ON means connection (LINK) Flashing means activity (ACT) OFF means Ethernet connection is down 10/100: Transmission Speed 100 Mbit/s = ON Transmission Speed 10 Mbit/s = OFF

<table>
<tr><td>Compliance</td><td>RoHS</td></tr>
<tr><td>Operating Temperature</td><td>-40°C to 70°C (-40°F to 158°F)</td></tr>
<tr><td>Storage Temperature</td><td>-40°C to 85°C (-40°F to 185°F)</td></tr>
<tr><td>Weight</td><td>0.172 lb. (0.078 kg)</td></tr>
<tr><td>Protection</td><td>IP67</td></tr>
<tr><td>Enclosure and Dimensions</td><td>Nylon 6/6, 30% glass fill
Ultrasonically welded
4.19 in x 1.81 in x 1.31 in (106 mm x 46mm x 33 mm)
L X W X H includes integral connectors
Flammability rating: UL 94V-0
Refer to dimensional drawing.</td></tr>
<tr><td>Electrical Connections</td><td>
<b>CAN</b> / <b>RS-232</b> / <b>RS-422 Connector</b>
1 Phoenix Contact M12 12-pin connector (A-coded), Female P/N: 1441833

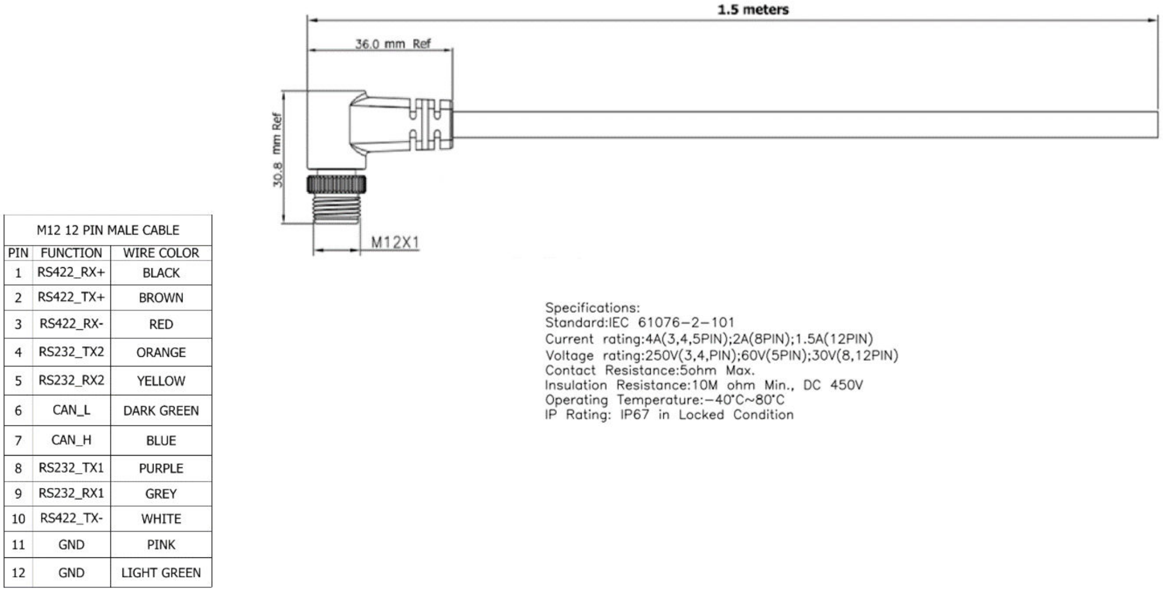
Note: To use RS-422 as RS-485, connect the Tx+ and Rx+ pin to D+ on the RS-485 connector. Also connect the Tx- and Rx- to pin D-.
<table>
<tr><th>PIN #</th><th>Description</th><td rowspan="13">


</td></tr>
<tr><td>1</td><td>RS-422 RX+</td></tr>
<tr><td>2</td><td>RS-422 TX+</td></tr>
<tr><td>3</td><td>RS-422 RX-</td></tr>
<tr><td>4</td><td>RS-232 TX 2</td></tr>
<tr><td>5</td><td>RS-232 RX 2</td></tr>
<tr><td>6</td><td>CAN_L</td></tr>
<tr><td>7</td><td>CAN_H</td></tr>
<tr><td>8</td><td>RS-232 TX 1</td></tr>
<tr><td>9</td><td>RS-232 RX 1</td></tr>
<tr><td>10</td><td>RS-422 TX-</td></tr>
<tr><td>11</td><td>GND</td></tr>
<tr><td>12</td><td>GND</td></tr>
</table>
<b>Ethernet</b> / <b>Power Connector</b>
1 Phoenix Contact M12 8-pin connector (A-coded), Female P/N: 1441817
<table>
<tr><th>PIN #</th><th>Description</th><td rowspan="9">


</td></tr>
<tr><td>1</td><td>Power +</td></tr>
<tr><td>2</td><td>Power -</td></tr>
<tr><td>3</td><td>Power -</td></tr>
<tr><td>4</td><td>Ethernet TX-</td></tr>
<tr><td>5</td><td>Ethernet RX+</td></tr>
<tr><td>6</td><td>Ethernet TX+</td></tr>
<tr><td>7</td><td>Power +</td></tr>
<tr><td>8</td><td>Ethernet RX-</td></tr>
</table>
</td></tr>
<tr><td>Mating Connectors</td><td>Not supplied
Mating connectors should meet the following standard for M12 connectors,
IEC 61076-2-101:2012. They should be A-coded.</td></tr>
<tr><td>Mating Cables</td><td><b>AX070531</b> (1.7 m (5.5 ft.), 8-pin M12 A-coded, Unterminated Leads, Ethernet Jack, Ethernet and Power Cable) mates with the Ethernet / Power Connector

<b>AX070533</b> (RS-232, RS-232 (RS-485), RS-422 Cable 1.5 m (5 ft.), 12-pin M12, Unterminated Leads, CAN Cable) mates with the CAN / RS-232 / RS-422 / RS-485 Connector</td></tr>
<tr><td>Mounting</td><td>Mounting holes sized for #10 or M5 bolts</td></tr>
</table>

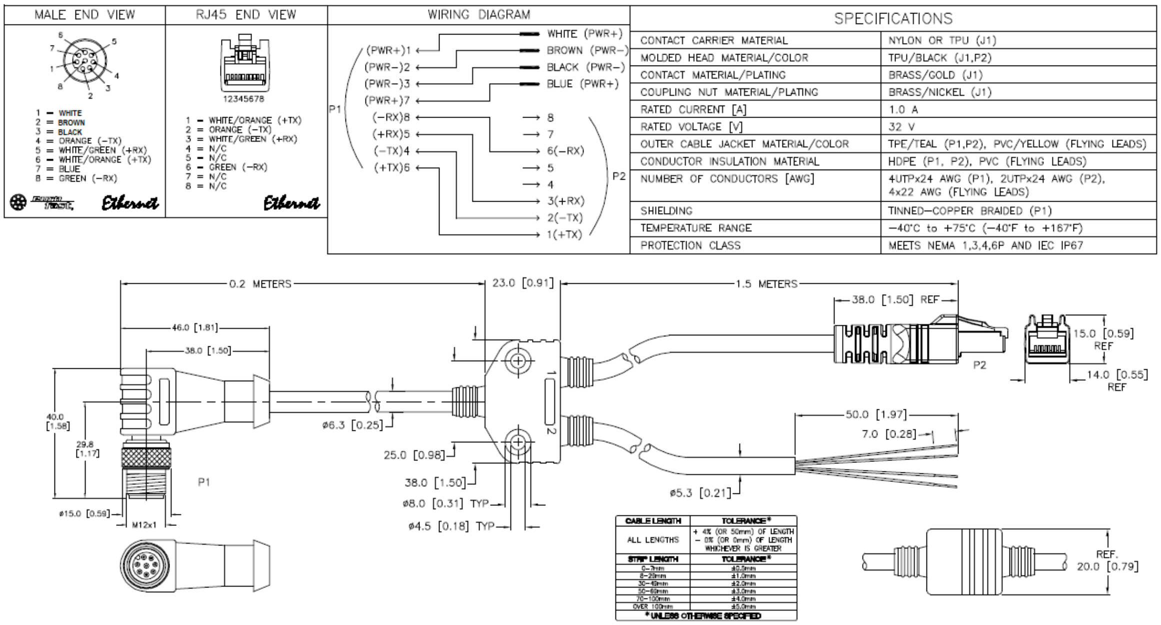
Dimensional Drawings



M12 12 PIN MALE CABLE		
PIN	FUNCTION	WIRE COLOR
1	RS422_RX+	BLACK
2	RS422_TX+	BROWN
3	RS422_RX-	RED
4	RS232_TX2	ORANGE
5	RS232_RX2	YELLOW
6	CAN_L	DARK GREEN
7	CAN_H	BLUE
8	RS232_TX1	PURPLE
9	RS232_RX1	GREY
10	RS422_TX-	WHITE
11	GND	PINK
12	GND	LIGHT GREEN

Specifications:
Standard:IEC 61076-2-101
Current rating:4A(3,4,5PIN);2A(8PIN);1.5A(12PIN)
Voltage rating:250V(3,4,PIN);60V(5PIN);30V(8,12PIN)
Contact Resistance:5ohm Max.
Insulation Resistance:10M ohm Min., DC 450V
Operating Temperature:−40°C~80°C
IP Rating: IP67 in Locked Condition

AX070533 Mating Cable



MALE END VIEW

1 = WHITE
2 = BROWN
3 = BLACK
4 = ORANGE (−TX)
5 = WHITE/GREEN (+RX)
6 = WHITE/ORANGE (+TX)
7 = BLUE
8 = GREEN (−RX)

RJ45 END VIEW

1 = WHITE/ORANGE (+TX)
2 = ORANGE (−TX)
3 = WHITE/GREEN (+RX)
4 = N/C
5 = N/C
6 = GREEN (−RX)
7 = N/C
8 = N/C

SPECIFICATIONS	
CONTACT CARRIER MATERIAL	NYLON OR TPU (J1)
MOLDED HEAD MATERIAL/COLOR	TPU/BLACK (J1,P2)
CONTACT MATERIAL/PLATING	BRASS/GOLD (J1)
COUPLING NUT MATERIAL/PLATING	BRASS/NICKEL (J1)
RATED CURRENT [A]	1.0 A
RATED VOLTAGE [V]	32 V
OUTER CABLE JACKET MATERIAL/COLOR	TPE/TEAL (P1,P2), PVC/YELLOW (FLYING LEADS)
CONDUCTOR INSULATION MATERIAL	HDPE (P1, P2), PVC (FLYING LEADS)
NUMBER OF CONDUCTORS [AWG]	4UTPx24 AWG (P1), 2UTPx24 AWG (P2), 4x22 AWG (FLYING LEADS)
SHIELDING	TINNED−COPPER BRAIDED (P1)
TEMPERATURE RANGE	−40°C to +75°C (−40°F to +167°F)
PROTECTION CLASS	MEETS NEMA 1,3,4,6P AND IEC IP67

CABLE LENGTH	TOLERANCE*
ALL LENGTHS	+ 4% (OR 50mm) OF LENGTH − 0% (OR 0mm) OF LENGTH WHICHEVER IS GREATER
STRIP LENGTH	TOLERANCE*
0−7mm	±0.5mm
8−29mm	±1.0mm
30−49mm	±2.0mm
50−69mm	±3.0mm
70−100mm	±4.0mm
OVER 100mm	±5.0mm
* UNLESS OTHERWISE SPECIFIED	

AX070531 Mating Cable

AXIOMATIC
Global Electronic Solutions

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