User Manual UMAX140900A
Version 2
Firmware 2.xx

USER MANUAL

Ethernet to CAN Converter

P/N: AX140900A

Axiomatic Technologies Oy
Höytämöntie 6 C
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

A	Ampere
API	Application Programming Interface
ARP	Address Resolution Protocol
AX	Axiomatic
°C	Celsius (degree)
CAN	Controller Area Network
CE	Conformité Européenne (European Conformity)
EA	Electronic Assistant®. PC application software from Axiomatic
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMC	Electromagnetic Compatibility
ESD	Electrostatic Discharge
FCC	Federal Communications Commission
G	Acceleration in Gravity Units
GPL	General Public License
hr	hour
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ID	Identifier
IEC	International Electrotechnical Commission
IP	Internet Protocol or Ingress Protection (for housing)
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
L	Length (for size)
LAN	Local Area Network
LED	Light-Emitting Diode
m	meters
MAC	Media Access Control (address)
Mbps	Megabits per second
MDIX	Medium Dependent Interface Crossover (MDI-X)
ms	millisecond
M2M	Machine to Machine
PA	Polyamide
PHY	Physical Layer Transceiver (Ethernet chip)
P/N	Part Number
PoE	Power Over Ethernet
REST	Representational State Transfer
RESTful	Based on REST API (Representational State Transfer Application programming interface)
RoHS	Restriction of Hazardous Substances
RTOS	Real-Time Operating System
SP	Service Pack
SSP	Software Support Package
SW	Software
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

UL	Underwriters Laboratories (safety organization)
URL	Uniform Resource Locator
USB	Universal Serial Bus
V	Volt
VDC	Volt Direct Current
W	Watt or Width (for size)
WAN	Wide Area Network

TABLE OF CONTENTS

1 INTRODUCTION

The following user manual describes architecture and functionality of the Ethernet to CAN Converter. It also contains technical specifications of the device.

This converter is an upgraded version of AX140900, Ethernet to CAN Converter, with a new and improved functionality. It can be used as a drop-in replacement of AX140900 in most applications.

The user manual is valid for application firmware with the same major version number as the user manual. For example, this user manual is valid for any application firmware version 2.xx. Updates specific to the user manual are done by adding letters: A, B, …, Z to the user manual version number.

The user can check the application firmware version number using the device embedded web server interface.

2 CONVERTER DESCRIPTION

The Ethernet to CAN Converter converts CAN frames into UDP or TCP IP datagrams and sends them over the Ethernet network. The device can also convert UDP or TCP datagrams into CAN frames and transmit them on the CAN network.

The converter has one CAN and one Ethernet port. It supports a high-speed CAN with baud rate up to 1Mbit/s and a fast 100Mbit/s Ethernet. All standard and extended CAN frames, including data and remote frames, are supported.

The power can be passed through to the CAN port. Protection is provided.

The converter contains a web server hosting a website that users can access using any web browser on a PC or other device. The website displays the converter configuration and diagnostics and can be used to update configuration parameters and flash new firmware. The website is password protected. For M2M communication, the web server supports a RESTful interface.

A simple Windows command-line application `AxioDisc.exe` is provided to locate the converter on the LAN.

To ensure low latency in processing CAN and Ethernet messages, the converter software runs under control of a real-time operating system.

The converter is designed to work on off-road machinery or in a harsh industrial environment with power transients, high humidity, and vibrations.

2.1 Hardware Block Diagram

The converter hardware block diagram is presented in Figure 1.

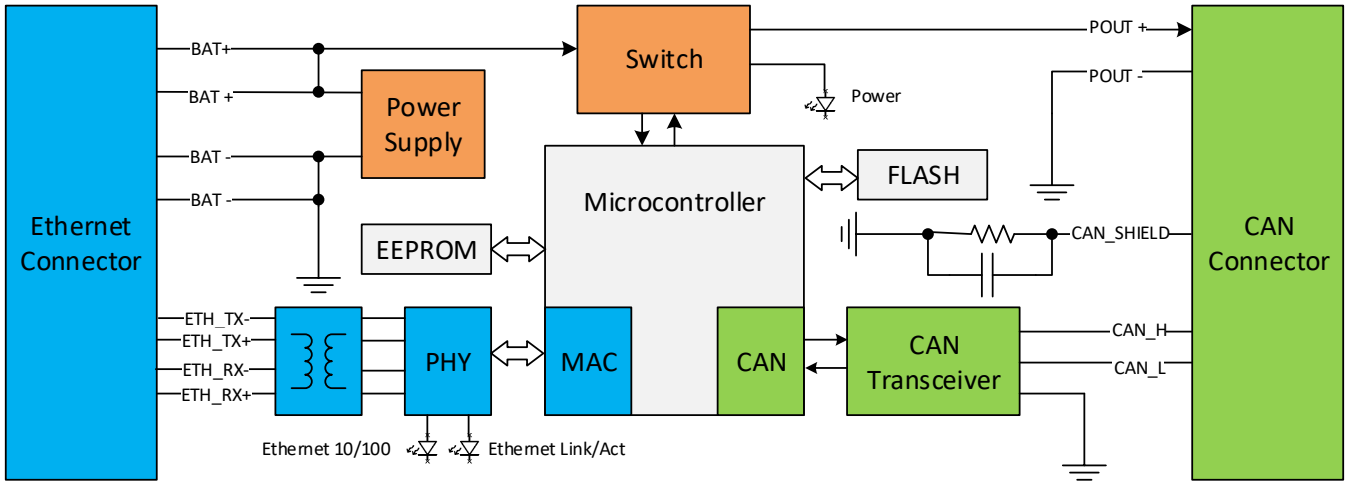


Figure 1. Converter Hardware Block Diagram

The converter is powered from the Ethernet connector using the dedicated power lines (BAT+ and BAT-), see Figure 1. The power from the Ethernet connector can be delivered to the CAN connector power lines (POUT+, POUT-) through the switch controlled by the microcontroller.

The switch has a bicolor Power LED indicator. It lights green when the converter is powered (from the Ethernet connector) and turns red if there is an error on the CAN connector power line POUT+.

The Ethernet PHY has hardwired Speed (10/100) and Link/Activity LED Indicators.

An ARM Cortex-M4 microcontroller runs TCP/IP stack and all Ethernet to CAN conversion logic.

2.2 Device Organization

The Ethernet to CAN Converter software organization is shown in Figure 2.

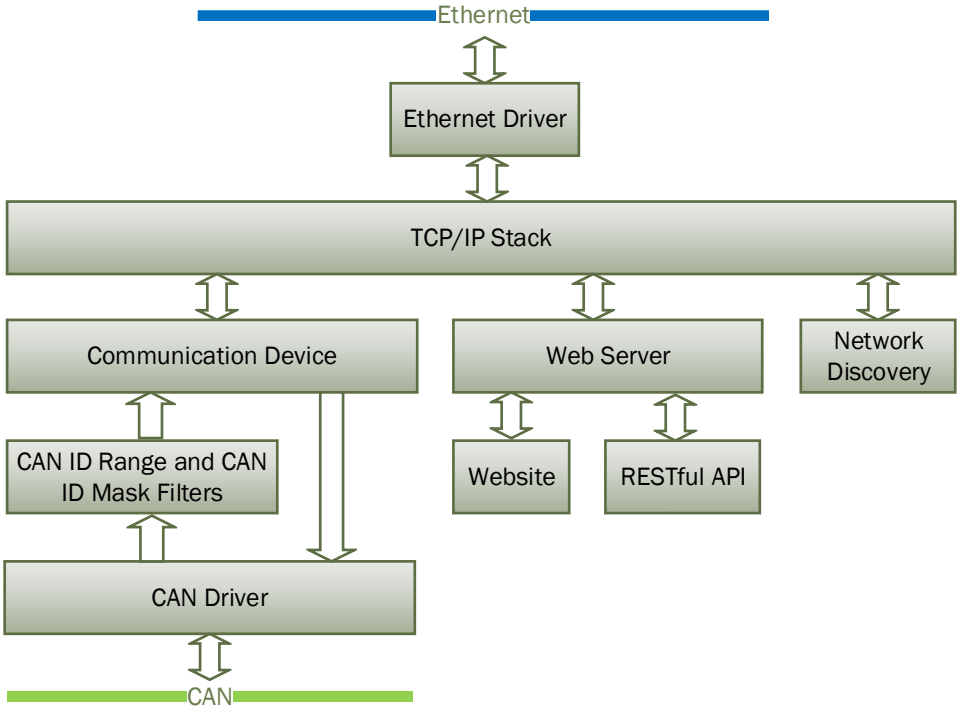


Figure 2. Converter Software Organization

The input Ethernet messages from the *Ethernet Driver* are processed by the *TCP/IP Stack* and then routed to the *Communication Device* for conversion into CAN frames. The Ethernet messages are also routed to the *Web Server* and the *Network Discovery* modules.

The *TCP/IP Stack* maintains necessary software infrastructure to support UDP and TCP IP communication.

The *Communication Device* establishes and maintains connection with other IP devices on the Ethernet network through data sockets. It also utilizes an Axiomatic proprietary communication protocol to communicate CAN frames over IP network [1].

CAN frames extracted from IP messages are sent directly to the CAN bus through the *CAN Driver*. CAN frames received from the CAN bus go through the *CAN ID Range* and *CAN ID Mask* filters before they are converted into IP messages and sent over to connected IP nodes by the *Communication Device*.

The *Web Server* supports HTTP requests to the device internal *Website* and the *RESTful API*.

The *Network Discovery* module supports an Axiomatic proprietary network discovery protocol to locate the converter IP address on the LAN [2].

2.2.1 Communication Device

The *Communication Device* supports a client/server communication model. In this model, the *Communication Device* operates primarily as a server, allowing external clients to establish independent connections with the device.

In addition to the server role, the device can also act as a client, if the *Auto Connect to Remote* configuration parameter is set to *Yes*. In this case, the device will try to establish a connection with a customer-specified remote server.

The total number of remote connections is limited to 10. If the CAN network traffic is high, this number should be further reduced, or the connections will become unstable due to limited internal resources of the microcontroller, which are dynamically allocated between open connections.

The *Communication Device* can use either UDP or TCP internet protocol, depending on the value of the *Device Port Type* configuration parameter.

2.2.1.1 UDP Protocol

The UDP protocol is set by default. Since it is a connectionless protocol, one data socket serves all device communication needs, see Figure 3.

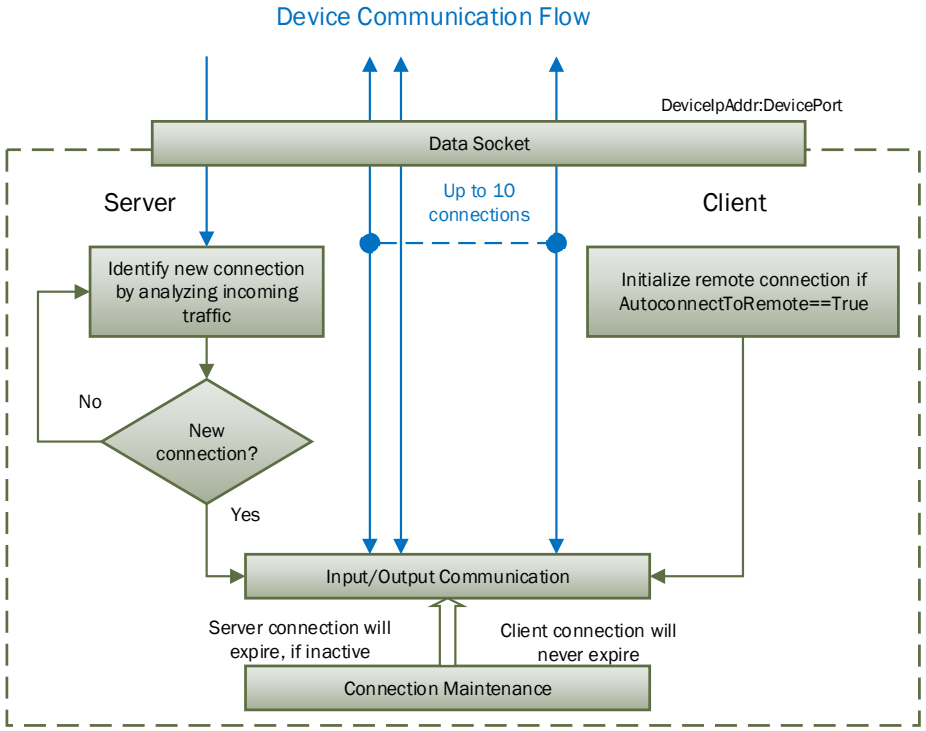


Figure 3. Communication Device. UDP Protocol

All connections with the device are virtual. On the server side, the device analyzes the incoming traffic to check for the new connections. Once a new IPAddress:Port combination is

detected, the connection is established, and the device starts sending CAN data with Heartbeat messages to the new node.

There are no restrictions on the IP address and port for the incoming connections.

If a client-side is activated by the *Autoconnect to Remote* configuration parameter, the device will automatically start sending CAN data with Heartbeat messages to the remote node on start-up.

To ensure that the device does not send data to not functioning (dead) or disconnected nodes, the server-side connections will expire in 10 sec of inactivity, if no valid data is received from the remote node. The client-side connection will never expire.

2.2.1.2 TCP Protocol

When TCP protocol is used, the *Communication Device* opens an individual data socket for each device connection, see: Figure 4.

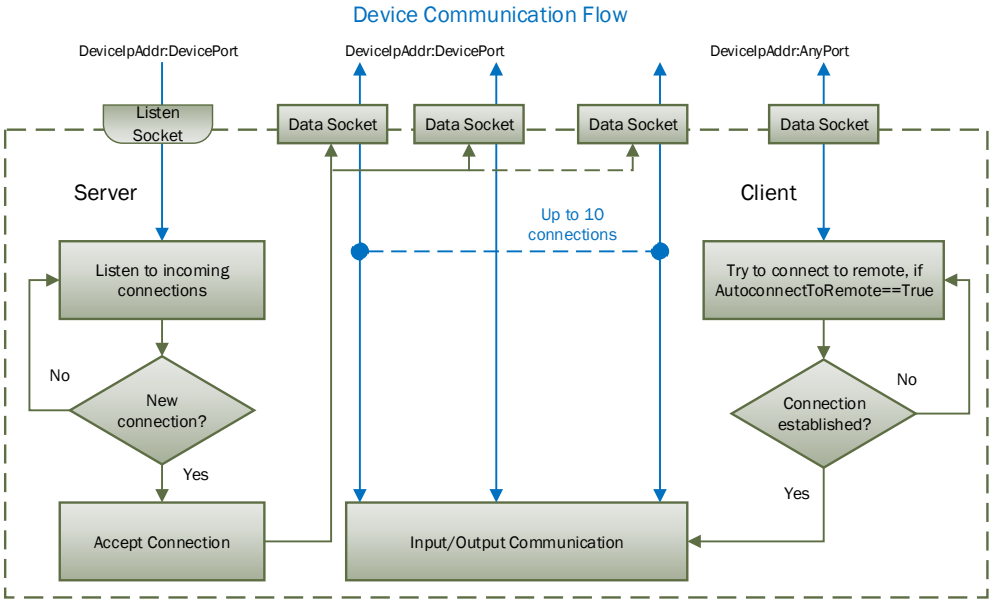


Figure 4. Communication Device. TCP Protocol

The server side opens a listening socket for incoming connections. Once a connection is accepted, a new data socket is created to handle input/output communication with the remote node. There are no restrictions on the IP address and port for the incoming connections, similar to the UDP mode.

On the client side, if *Auto Connect to Remote* is set to *Yes*, a data socket is created for connection with the remote node. A random free port number is assigned to the socket. If the connection drops, the device will try to automatically reconnect with the node to maintain the client connection.

2.2.1.3 CAN Frame Routing

Once an IP connection with a remote node is established, the *Communication Device* starts sending CAN frames encoded into IP messages to that node. The way how CAN frames are then routed inside of individual IP nodes to CAN ports is controlled by a routing mechanism implemented in Axiomatic proprietary communication protocol [1].

The CAN frame routing is achieved by adding a special routing address to each CAN frame transmitted on IP network. To allow simultaneously sending CAN frames to different CAN ports on an IP device, this address is designed the way that it can contain several individual CAN port IDs.

The CAN routing address is presented by two parameters: *Channel Group (CG)* and *Channel ID Set (CIDS)*: (CG, CIDS). The *Channel Group* uniquely identifies a set of 32 channels. The *Channel ID Set* defines channels, or CAN ports, in the *Channel Group* that are addressed by this routing address.

Each Channel ID is presented as a bit (flag) at the channel number position in the *Channel ID Set* field. The *Channel ID Set* can contain up to 32 Channel IDs that are ORed together in the 4-byte *Channel ID Set* field. This allows sending a CAN frame simultaneously to up to 32 independent CAN ports on the same device, see below:

$$\text{CAN Address} = (CG, CIDS)=(CG,[CN_1,CN_2,\ldots,CN_{32}])=(CG, CID_1|CID_2|,\ldots,|CID_{32}) \quad (1)$$

Where:

$CN_i = i$, $i=1,\ldots,32$. Channel Number CN_i is shown in the CAN Address notation only if the channel is selected. If no channels are selected, 0 is shown.

$CID_i=\{ 1<<(CN_i-1)$, $i=1,\ldots,32$, when Channel Number CN_i is selected; 0 – otherwise$\}$.

| - Bitwise OR operator.

Each CAN port has its own receiving and transmitting CAN routing address. Both addresses can be the same. The transmitting routing address is sent with CAN frames received by the port and transmitted in IP messages, and the receiving routing address is used to filter out the incoming CAN frames in IP messages that will be transmitted by that port.

The CAN port will transmit the incoming CAN frame on the CAN bus only if its routing address matches the CAN port receiving address.

Two addresses are considered to match if their *Channel Groups* are equal and there is at least one selected (set to 1) common Channel ID in their *Channel ID Sets*, see below:

$$\text{AddressMatch}((CG_1, CIDS_1), (CG_2, CIDS_2)) = (CG_1 == CG_2) \;\&\&\; ((CIDS_1 \;\&\; CIDS_2)>0) \quad (2)$$

Where:

== - Logical EQUAL operator.

&& - Logical AND operator.

& - Bitwise AND operator.

When receiving routing addresses of CAN ports on one device match transmitting routing addresses of CAN ports on another device, the CAN ports on the devices will be virtually connected together.

CAN routing addresses with all Channel IDs equal to 0 (CG=0…255, CIDS=0) are undefined. They do not match any routing address. When a receiving routing address is undefined, the CAN port will not receive any CAN frames from IP network. When a transmitting routing address is undefined, the CAN port will not transmit any CAN frames on IP network.

For backward compatibility, it is assumed that legacy devices and software that use the old version of Axiomatic communication protocol (that does not support CAN frame routing), use the default (CG=0, CIDS=1) routing address.

2.2.1.3.1 CAN Routing Address Assignment

A CAN routing address assignment in the user's system is application specific. Depending on the protocol implementation, both receiving and transmitting routing addresses of a CAN port can contain one or multiple channel numbers and use the same or different channel group.

In a simple point-to-point connection, the same receiving and transmitting address with one channel number can be used for both virtually connected CAN ports, see Figure 5. The CAN ports are named by their transmitting routing address pairs (CG,[CN]): CAN<CG>_<CN>.

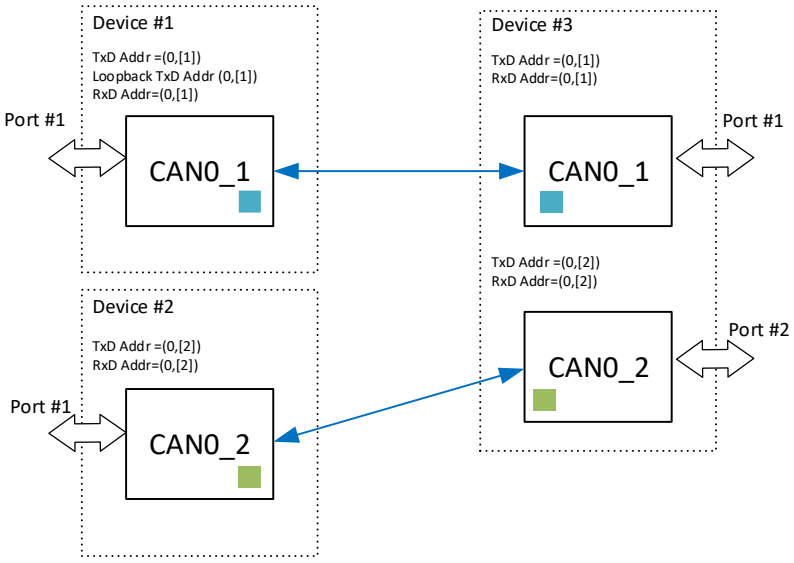


Figure 5. CAN Routing Address Assignment. Example of Point-to-Point Connection

In this example, CAN0_1 port on the Device #1 is virtually connected to CAN0_1 port on the Device #3, and CAN0_2 port on the Device #2 is virtually connected to CAN0_2 port on the Device #3. Loopback messages from CAN0_1 port on the Device #1 are also transmitted to CAN0_1 port on the Device #3.

In more elaborated applications, a practical solution is to use a transmitting routing address with one channel number and a receiving routing address with multiple channel numbers for a CAN port. This approach gives a better control over input connections to the CAN port in comparison with one receiving channel with multiple transmitting channels scheme. It is also easier to manage than a multiple receiving channels with multiple transmitting channels scheme.

When a transmitting routing address with one channel number is used, the channel number can be set to the global port number of the CAN port in the application system. The channel numbers in the receiving routing address will be then set to the global port numbers of CAN ports connected to the given port in the system. It is assumed that the global CAN port numbers are unique for a given channel group.

An example of an application system with one-to-many and many-to-one CAN port connections is presented in Figure 6.

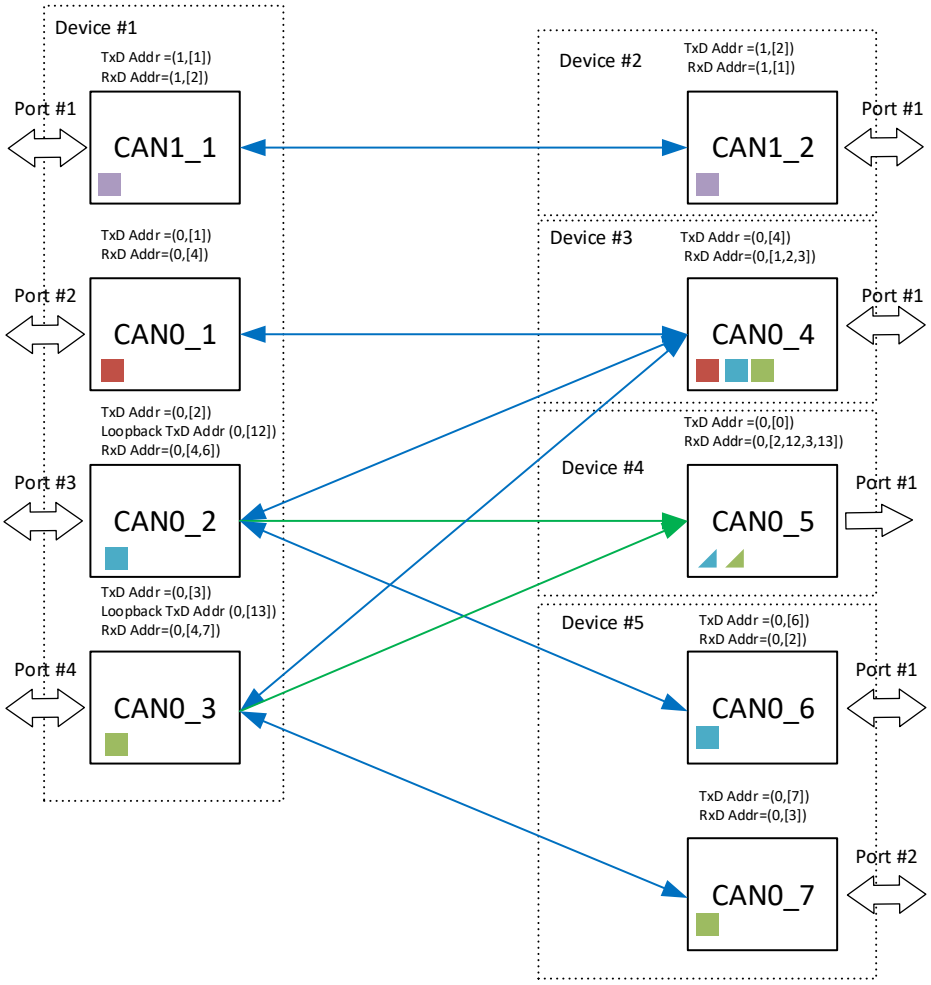


Figure 6. CAN Routing Address Assignment. Example of One-to-Many and Many-to-One Connections

In this example, the channel group CG=1 is used to connect CAN1_1 port on the Device #1 with CAN1_2 port on the Device #2. The channel group CG=0 is used to connect CAN ports the following way:

- CAN0_1, CAN0_2, CAN0_3 ports on the Device #1 are connected to CAN0_4 port on the Device #3.
- CAN0_2, CAN0_3 (and their loopback channels) on the Device #1 are connected one-way (only for transmission) to CAN0_5 port on the Device #4 to monitor CAN traffic on CAN0_2 and CAN0_3 ports through this port.
- CAN0_2 port on the Device #1 is connected to CAN0_6 port on the Device #5.
- CAN0_3 port on the Device #1 is connected to CAN0_7 port on the Device #5.

2.2.2 Web Server

The converter web server delivers a front-end user interface with the converter through the embedded website and supports a RESTful API for communication with other devices on the IP network.

2.2.2.1 Website

The converter web server runs a dynamic website that shows the device general information, configuration parameters, and real-time diagnostics. The website is password protected. It can

be used to update, save or restore configuration parameters, change the device password, and upload new firmware.

The user's web browser should support JavaScript.

For more information, see *Converter Configuration*, *Converter Diagnostics,* and *Firmware Update* sections of this document.

2.2.2.2 RESTful Interface

The RESTful API allows the converter to communicate with other devices on the IP network to read and modify the converter configuration parameters, receive diagnostic information, upload or download the converter configuration files, and update the converter firmware in the field.

For more information, see *RESTful Interface* section of this document.

2.2.3 Network Discovery

The device supports a proprietary Axiomatic discovery protocol [2]. It allows to find the device IP address on the LAN using Axiomatic discovery console application `AxioDisc.exe`. For more information, see *Converter Discovery* section.

3 CONVERTER CONFIGURATION

The converter supports configuration over the internal website running on the device embedded web server. For security reasons, the device website is password protected.

The device power should not be interrupted during updating configuration parameters to avoid possible corruption of nonvolatile memory.

3.1 Connecting to the Device

The default *Device IP Address* is “192.168.0.34” and the default *Web Server Port* is “80”. Please, make sure that there are no other devices on this IP address when connecting the converter to your LAN for the first time.

To connect to the device, the user should run any web browser and point it to the *Device IP Address*. It is not necessary to specify the *Web Server Port*, if the web server uses the default standard port 80.

After a successful connection, the user will see the device login page, see Figure 7.

Figure 7. Device Login Page

If necessary, the user will need to allow JavaScript (this setting is default in the majority of web browsers). If JavaScript is disabled, the website will show a message asking to activate JavaScript at the top of the web page, see Figure 8.

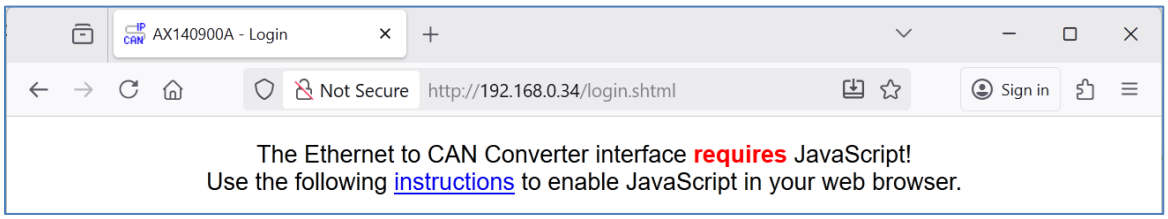


Figure 8. Enable JavaScript Prompt

The device default password is **Axiomatic1** (case sensitive).

The device home page will be shown upon entering the correct password. The user can see the password text when *Show password* checkbox is clicked.

It is strongly advisable to change the default password to a unique one after performing the initial device setup to prevent unauthorized access to the device, see *Password Update Web Page* subsection of this document.

To protect the device from password guessing, the number of unsuccessful attempts to connect to the device is limited. Access to the device will be denied for several minutes upon reaching this limit. Also, for security reasons, the device web session will be automatically closed and the user logged out on the user's inactivity.

In case the password is lost, the user should contact Axiomatic for the password recovery procedure [4].

3.2 Device Homepage

The device home page shows the device information, including the converter part number, serial number, and firmware version, see Figure 9. It also shows *Ethernet* and *CAN* configuration parameters including the status of CAN ID range and CAN ID mask input filters.

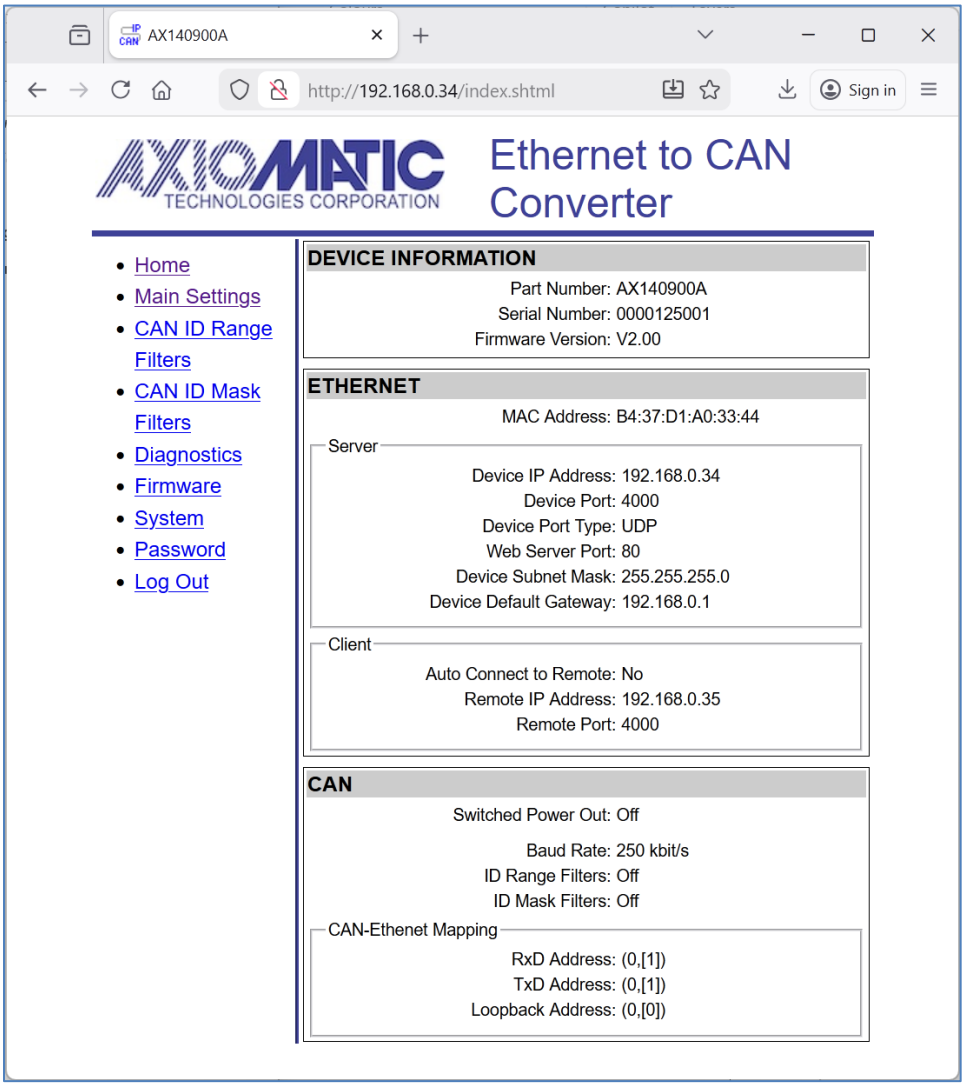


Figure 9. Converter Home Page[1]

[1]The Firmware Version number may be different from the firmware version described in the manual.

The *Ethernet* configuration parameters are combined into *Server* and *Client* groups for convenience.

The *CAN* configuration parameters include *CAN-Ethernet Mapping* group presenting CAN routing addresses for Axiomatic proprietary CAN to Ethernet converting protocol [1].

The *Ethernet* and *CAN* configuration parameters have tooltips clarifying their meaning, see Figure 10.

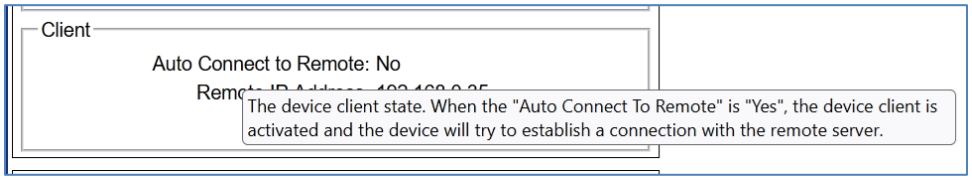


Figure 10. A Tooltip for the "Auto Connect to Remote" Configuration Parameter

3.3 Changing Configuration Parameters

The converter configuration parameters can be changed either on an individual basis through the configuration web pages: *Main Settings*, *CAN ID Range Filters* and *CAN ID Mask Filters* or by using a configuration file on the *System Settings* web page. Configuration parameters can be also read and changed using RESTful interface.

3.3.1 Configuration Web Pages

The user can change all configuration parameters except CAN ID range and mask filters in interactive mode using the *Main Settings* web page, see Figure 11. CAN ID range and mask filters have their own web pages.

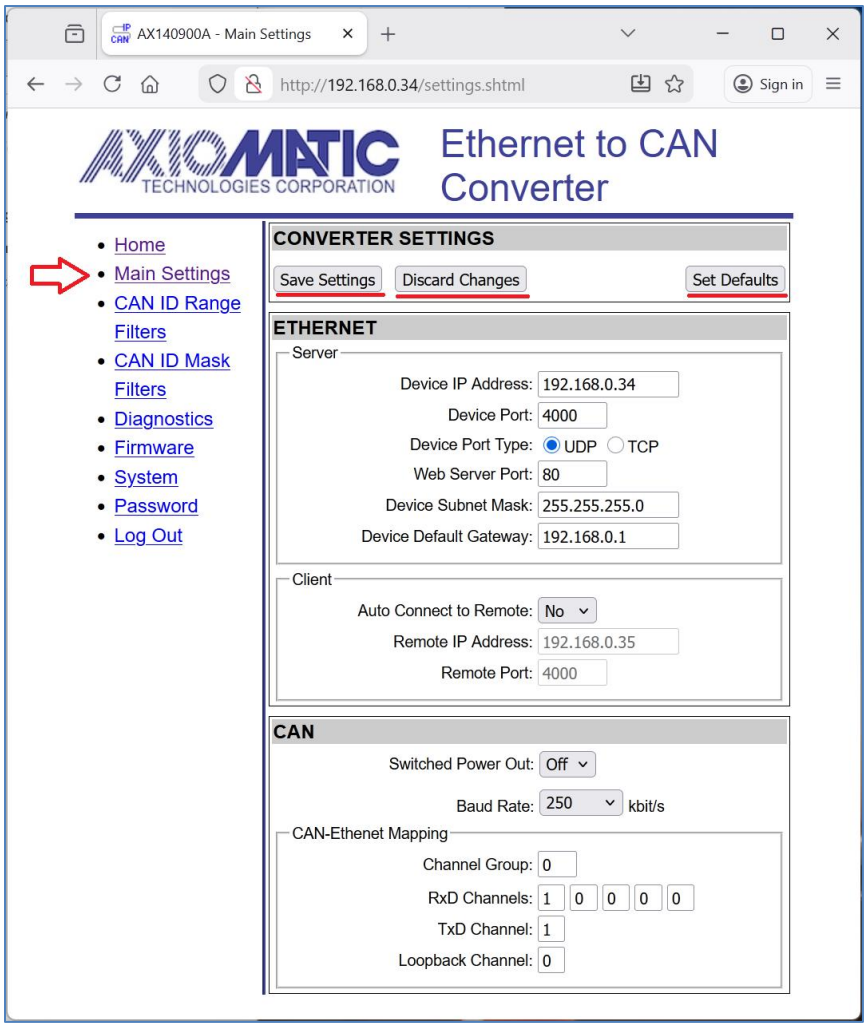


Figure 11. Converter Main Settings Web Page

The configuration web pages can be reached by clicking on the link on the left side of the website.

Each configuration web page has fields to enter values of configuration parameters and three buttons: *Save Settings*, *Discard Settings* and *Set Defaults*.

The *Save Settings* button will save configuration parameters in non-volatile memory and apply the new settings immediately after saving. The converter performs all necessary internal reconfigurations and resets on the fly without rebooting the device.

The *Discharge Settings* button will bring back the original converter settings on the web page before editing. In case the user leaves the page without saving, all changes will be discarded as well.

the *Set Defaults* button will load the default values of the configuration parameters into the data fields on the web page. The configuration parameters will not be automatically saved.

The configuration parameters have tooltips for the user's convenience. The *Remote IP Address* and *Remote Port* are disabled when *Auto Connect to Remote* is set to *No*.

When the user presses *Save Settings* button, the web page runs a script to check the validity of the new configuration parameters before uploading them to the web server. For example, the following alert message will be displayed if the user enters the same value for the *Device Port* and *Web Server Port* addresses, see Figure 12.

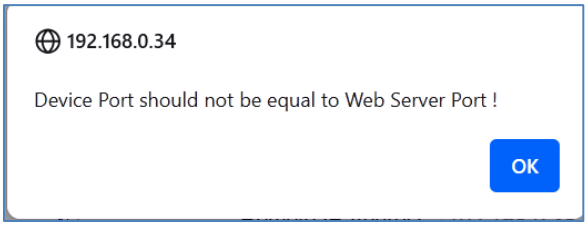


Figure 12. Settings Alert. Error in Configuration Parameters

The website alert messages should be enabled (not suppressed) in the browser to see this and other feedback messages.

After pressing the *Save Settings* button and saving the configuration parameters in non-volatile memory, the converter replies with a confirmation message showing the result of the saving operation. For example, for a successful operation, the following message will appear, see Figure 13.

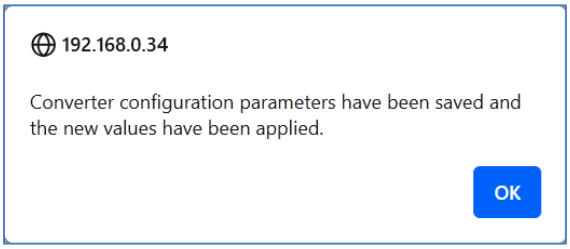


Figure 13. Settings Alert. Configuration Parameters have been Changed Successfully

If the user has changed the *Device IP Address* or *Web Server Port*, the website will be automatically re-loaded at the new location in the web browser, see Figure 14.

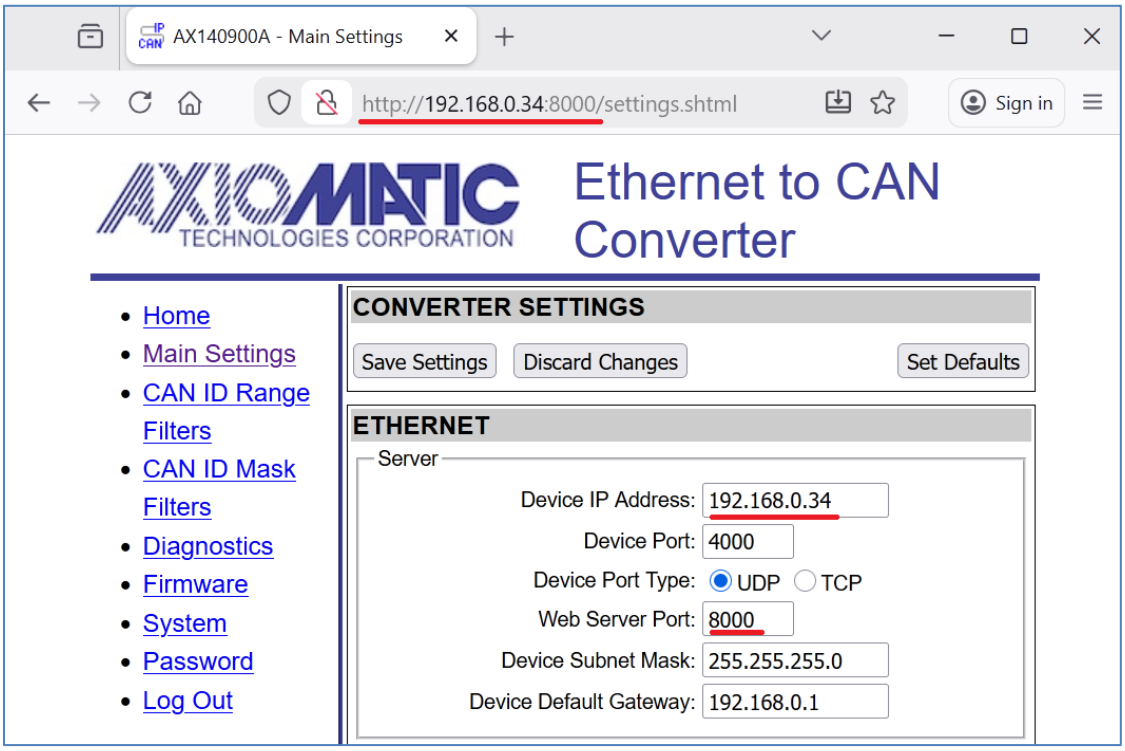


Figure 14. Website Automatic Relocation

3.3.2 Ethernet Configuration

All *Ethernet* configuration parameters can be changed through the *Main Settings* web page, except the *MAC Address,* which is programmed at the factory. The user-configurable configuration parameters are presented in Table 1.

The user should ensure the correctness of IP addresses. This includes avoiding special IP addresses (broadcast, multicast, loopback, etc.) when configuring the *Device IP Address* and the *Device Default Gateway*. The only exception is “0.0.0.0” address that can be used to disable the device default gateway.

The converter will check the validity of the IP addresses before saving the new configuration parameters to avoid permanent loss of communication with the embedded web server. Use 1…223 (except 127) for the first (rightmost) address octet in the *Device IP Address* and the *Device Default Gateway*. The *Device Default Gateway* should not be equal to the *Device IP Address*.

The *Device Subnet Mask* should have all ones on the left side and all zeros on the right side when converted to the binary presentation as per RFC 1878. The *Device Subnet Mask* should not be equal to "255.255.255.255” when the *Device Default Gateway* is equal to “0.0.0.0”.

Table 1. Ethernet Configuration Parameters

Configuration Parameter	Default Value	Range	Description
Device IP Address	192.168.0.34	Any IP address[1]	The device IP address. The embedded web server uses the same IP address.
Device Port	4000	Any IP port, except the *Web Server Port* and	The device server port. The device is listening to this port for incoming connections. The *Discovery Protocol Port*

Configuration Parameter	Default Value	Range	Description
		the *Discovery Protocol Port*	(35100) and the *Web Server Port* should not be used.
Device Port Type	UDP	{UDP, TCP}	Type of the IP protocol used by the device. The device server and client use the same IP protocol.
Web Server Port	80	Any IP port, except the *Device Port* and the *Discovery Protocol Port*[1]	The communication port of the embedded web server
Device Subnet Mask	255.255.255.0	Any IP subnet mask[1]	The device subnet mask. Used also by the embedded web server
Device Default Gateway	192.168.0.1	Any IP address[1]	The device default gateway. Used also by the embedded web server
Auto Connect to Remote	No	{No, Yes}	The device client state. When the *Auto Connect to Remote* is *Yes*, the device client is activated, and the device will try to establish a connection with the remote server.
Remote IP Address	192.168.0.35	Any IP address	The remote server IP address. Used by the device client, when the *Auto Connect to Remote* is *Yes*
Remote Port	4000	Any port value	The remote server port. Used by the device client, when the *Auto Connect to Remote* is *Yes*

[1]Must be assigned by the network administrator. Restrictions apply

3.3.3 CAN Configuration

The CAN configuration parameters can be changed through the *Main Settings*, *CAN ID Range Filters,* and *CAN ID Mask Filters* web pages. The main CAN configuration parameters are available through the *Main Settings* web page, see Table 2.

Table 2. Main CAN Configuration Parameters

Configuration Parameter	Default Value	Range	Description
Switched Power Out	Off	{Off, On}	State of the switch delivering power to the CAN connector
Baud Rate	250 kbit/s	{1000, 666.6(6), 500, 250, 125, 100, 83.3(3), 50, 20, 10}[1]	The CAN baud rate
Channel Group	0	[0…255]	*CAN Port Channel Group*. This parameter is the first part of the *CAN Port Receiving, Transmitting, and Loopback Addresses*[2].
RxD Channels	[1,0,0,0,0]	[[0…32]…[0…32]]	Array of 5 *CAN Port Receiving Channel Numbers.* This array is converted to the *CAN Port Receiving Channel ID Set*, which is the second part of the *CAN Port Receiving Address*[2]. No CAN receiving from Ethernet if all channel numbers are 0

Configuration Parameter	Default Value	Range	Description
TxD Channel	1	[0…32]	*CAN Port Transmitting Channel Number*. This parameter is converted to the *CAN Port Transmitting Channel ID Set*, which is the second part of the *CAN Port Transmitting Address*[2]. No CAN transmitting on Ethernet if 0.
Loopback Channel	0	[0…32]	*CAN Port Loopback Channel Number*. This parameter is converted to the *CAN Port Loopback Channel ID Set*, which is the second part of the *CAN Port Loopback Address*[2]. No CAN transmitting on Ethernet if 0

[1] 666.6(6) and 83.3(3) kbit/s are defined as 667 and 83 kbit/s respectively, in the drop-down menu.
[2] Defined in Axiomatic proprietary communication protocol [1], see *CAN Frame Routing* subsection.

There are three CAN routing addresses associated with the CAN port on the converter: *CAN Port Receiving Address*, *CAN Port Transmitting Address*, and *CAN Port Loopback Address*. They all share the same *Channel Group* address.

The *CAN Port Receiving Address* is used to filter out the incoming CAN frames the converter receives from the Ethernet port. When the CAN frame address matches the *CAN Port Receiving Address,* the frame is received by the CAN port from Ethernet and transmitted on the CAN bus.

When all *RxD Channels* are set to 0, the *CAN Port Receiving Address* is undefined, and the CAN port will not receive any CAN frames from Ethernet.

The *CAN Port Transmitting Address* is the routing address the converter transmits with all CAN frames received from the CAN bus on the Ethernet port. When *TxD Channel* is set to 0, the *CAN Port Transmitting Address* is undefined, and the convertor will not transmit any CAN frames on Ethernet.

The *CAN Port Loopback Address* is the routing address the converter transmits with all CAN bus loopback frames on the Ethernet port. If the *Loopback Channel* is set to 0, the *CAN Port Loopback Address* is undefined, and the convertor will not transmit any CAN bus loopback frames on Ethernet. This is the default setting of the *CAN Port Loopback Address*[1].

[1]In case the user wants to activate loopback messages the same way as it is done in AX140900, the user should set the *Loopback Channel* to the same value as the *TxD Channel*. This way, the *CAN Port Loopback Address* will be equal to the *CAN Port Transmitting Address* and loopback CAN frames will be sent to the same address as the received CAN frames from the CAN bus.

The default values of the *CAN Port Receive Address* and the *CAN Port Transmit Address* are set to match the default (0,1) routing address of the converter to seamlessly work with legacy devices that use the old version of Axiomatic proprietary communication protocol [1] that does not support CAN frame routing.

3.3.4 CAN ID Range Filters

The CAN ID range filters are set through the *CAN ID Range Filters* configuration web page, see Figure 15.

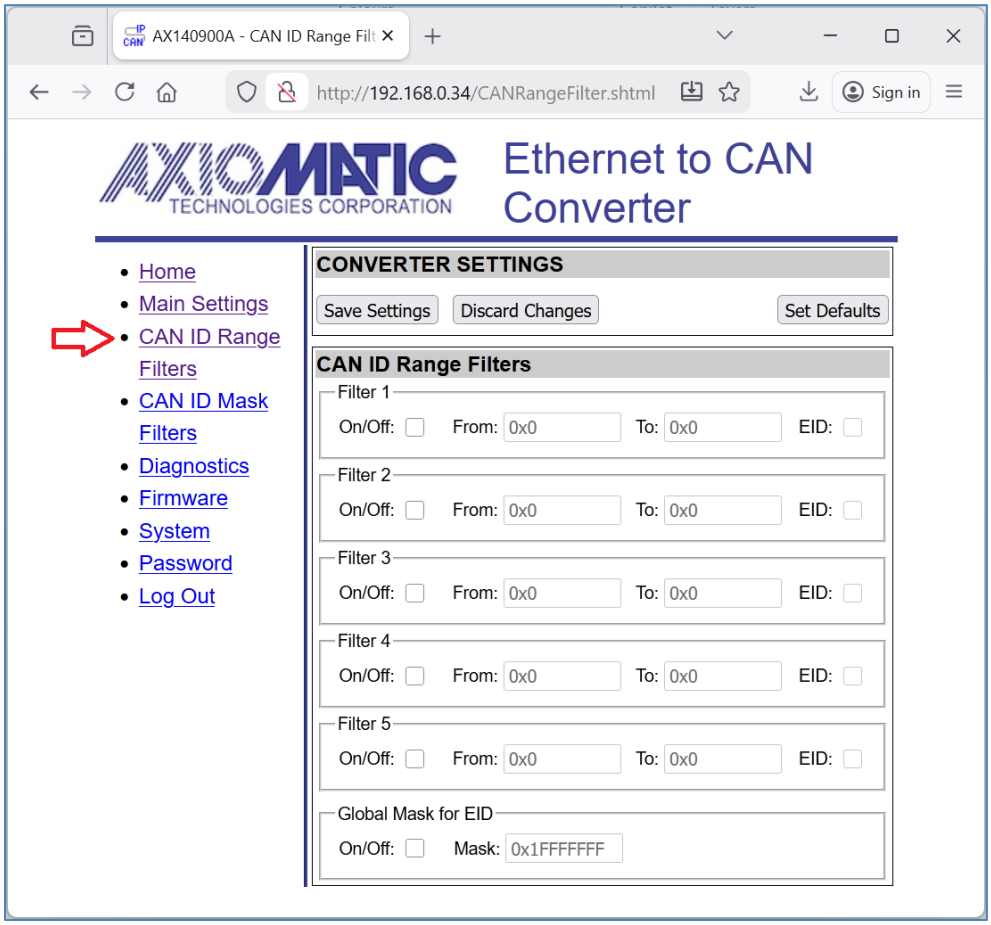


Figure 15. Converter CAN ID Range Filters Page

The user can independently configure five CAN ID range filters: *Filter 1, Filter 2, …, Filter 5.* Configuration parameters of the CAN ID range filters are presented in Table 3.

Table 3. Configuration Parameters of CAN ID Range Filters

Configuration Parameter	Default Value	Range	Description
Individual Filters			
On/Off	Off	{Off, On}	State of the CAN ID range filter
From	0	If EID=1, then [0...0x1FFFFFFF], else [0...0x7FF]	First CAN frame ID that will be accepted, when state of the CAN ID range filter is “On”
To	0	If EID=1, then [0...0x1FFFFFFF], else [0...0x7FF]	Last CAN frame ID that will be accepted, when state of the CAN ID range filter is “On”
EID	Off	{Off, On}	Extended CAN frame ID flag
Global Mask for EID			
On/Off	Off	{Off, On}	State of the global mask for extended CAN frame IDs
Mask	0	[0...0x1FFFFFFF]	Global mask for extended CAN frame IDs

Once the filter is activated by checking the *On/Off* box, the CAN input frames will pass through to the Ethernet port only if their CAN ID is within the range specified by *From* and *To* configuration parameters, see:

$$If\ ID_{CAN}\ \in\ [ID_{From};\ ID_{To}],\ then\ the\ CAN\ frame\ is\ accepted. \quad (3)$$

Where: ID_{CAN} – CAN frame ID,
ID_{From} – *From* configuration parameter,
ID_{To} – *To* configuration parameter.

The *EID* box (*Extended ID* box) defines whether the CAN frame ID is regular or extended.

The user can activate the *Global Mask for EID* configuration parameter through the *On/Off* checkbox at the bottom of the webpage. It will apply a global mask, set by the *Mask* configuration parameter, to the CAN frame ID. This Mask will be applied only for Extended ID CAN frames. Regular CAN frames will not be affected by this setting, see:

$$\forall\ ID_{CAN} \in \boldsymbol{EID},\ if\ (ID_{CAN}\ \&\ Mask)\ \in\ [ID_{From};\ ID_{To}],\ then\ the\ CAN\ frame\ is\ accepted. \quad (4)$$

Where: $\boldsymbol{EID}$ – Extended ID CAN frame set,
$Mask$ – *Mask* configuration parameter.
& – Bitwise AND operator.

The *Global Mask for EID* setting is suitable for J1939 filtering.

All CAN ID range filters run in parallel. It is sufficient to satisfy requirements of any active filter to pass the CAN frame to the Ethernet network.

If no active filters are defined, it is considered that the CAN ID range filters are disabled, and do not participate in the message filtering process. In this case, *ID Range Filters* are *Off* on the home page, and, if CAN ID mask filters are also disabled, all CAN input frames will be sent to the Ethernet port.

3.3.5 CAN ID Mask Filters

The CAN ID mask filters are set through the *CAN ID Mask Filter* configuration web page, see Figure 16.

There are five independent CAN ID mask filters: *Filter 1, Filter 2, …, Filter 5* available to the user. Configuration parameters of the CAN ID mask filters are presented in Table 4.

Table 4. Configuration Parameters of CAN ID Mask Filters

Configuration Parameter	Default Value	Range	Description
Individual Filters			
On/Off	Off	{Off, On}	State of the CAN ID mask filter
Mask	0	If EID=1, then [0...0x1FFFFFFF], else [0...0x7FF]	Mask applied to the CAN frame ID when state of the CAN ID mask filter is “On”
ID	0	If EID=1, then [0...0x1FFFFFFF], else [0...0x7FF]	*ID* that CAN frame ID is compared with after application of the *Mask*, when state of the CAN ID mask filter is “On”

Configuration Parameter	Default Value	Range	Description
EID	Off	{Off, On}	Extended CAN ID flag

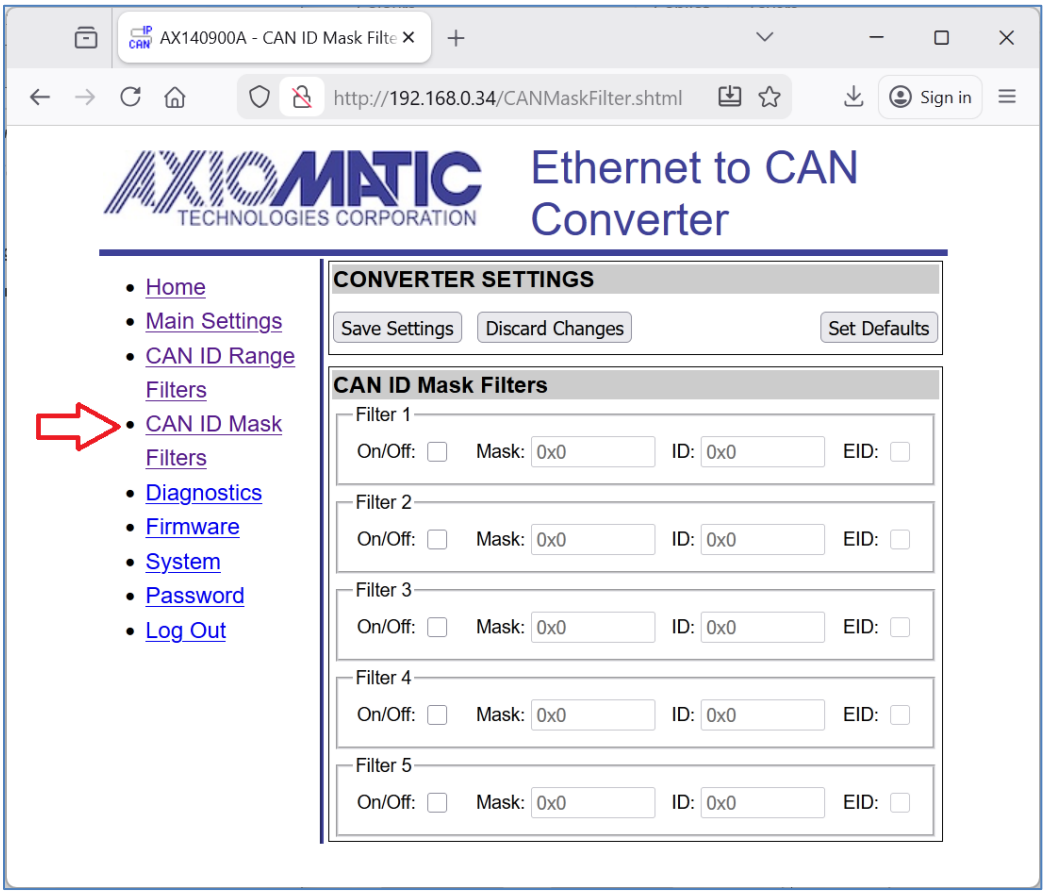


Figure 16. Converter CAN ID Mask Filters Page

Once the filter is activated by checking the *On/Off* box, the CAN input frames will pass through the filter to the Ethernet port only if their CAN frame ID satisfies the following condition:

$$If\ ID = ID_{CAN}\ \&\ Mask,\ then\ the\ message\ is\ accepted. \quad (5)$$

Where: ID_{CAN} – CAN frame ID,
$Mask$ – *Mask* configuration parameter,
ID – *ID* configuration parameter,
& – Bitwise AND operator.

The *EID* box (*Extended ID* box) defines whether the CAN message ID is regular or extended.

All CAN ID mask filters run in parallel the same way as CAN ID range filters. It is sufficient to satisfy requirements of any active filter to pass the CAN frame to the Ethernet port.

If no active filters are defined, it is considered that the CAN ID mask filters are disabled, and do not participate in the message filtering process. In this case, *ID Mask Filters* are *Off* on the home page, and, if CAN ID range filters are also disabled, all CAN input frames will be sent to the Ethernet port.

3.3.6 System Settings Web Page

The device configuration can be saved and then restored back from a configuration file. The configuration file operations are provided on the *System Settings* web page, see Figure 17, accessible by clicking on the *System* link on the left side of the website.

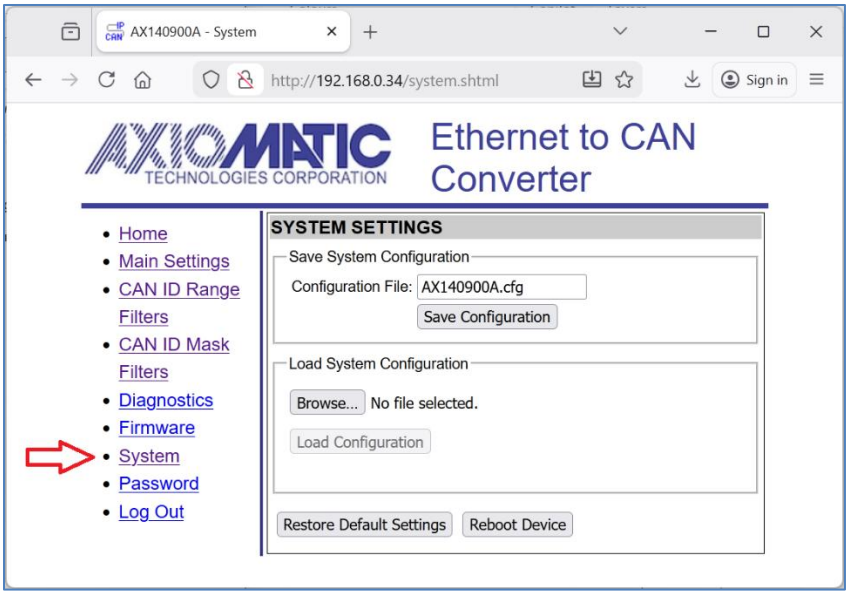


Figure 17. System Settings Web Page

3.3.6.1 Saving System Configuration

To save system configuration, the user should enter the system configuration file name in the *Configuration File* field and then press the *Save Configuration* button.

The default system configuration file name is “AX140900A.cfg”. The configuration file will be generated and saved in the *Downloads* location of the web browser, see Figure 18.

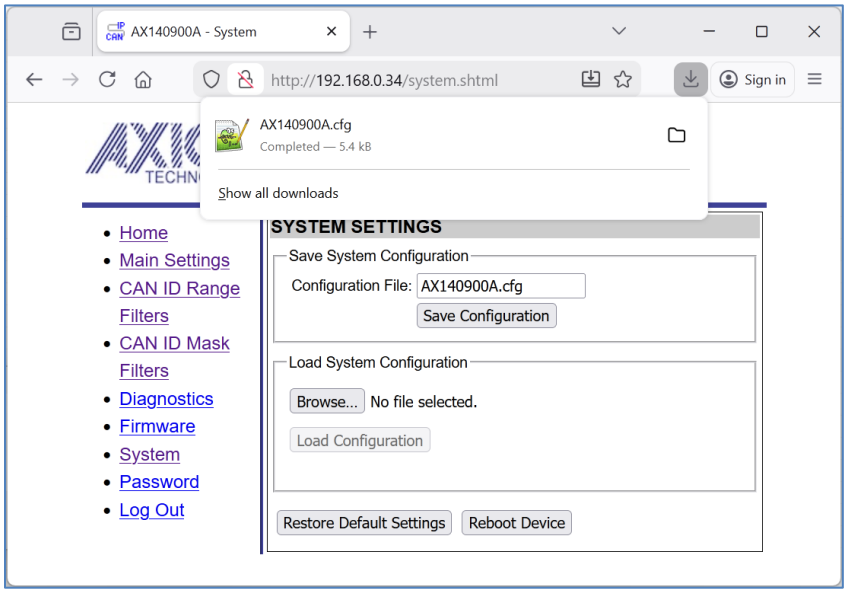


Figure 18. Saving System Configuration

In case the web session is expired on inactivity, an empty configuration file will be generated.

3.3.6.2 Loading System Configuration

The user can restore system configuration from a configuration file. The configuration file will be processed, and configuration parameters will be updated during the file upload operation.

To upload the device configuration file, the user should first select the configuration file by pressing the *Browse…* button in the *Load System Configuration* group on the *System Settings* web page. Then the user should press the *Load Configuration* button to upload the selected configuration file to the device.

The result of the upload operation will be shown to the user in an alert message from the web site. For example, a successful upload of previously saved configuration parameters will result in the following message, see Figure 19.

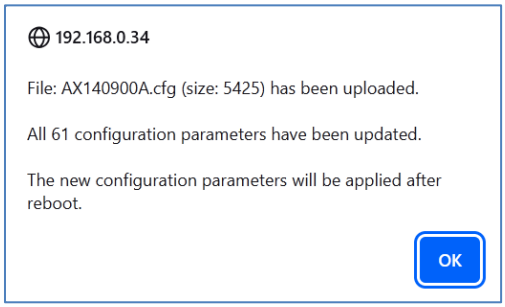


Figure 19. Loading System Configuration File Alert Message.
All Configuration Parameters Have Been Updated

The device upload operation provides extended diagnostic information to the user. In case of an error in the configuration file, a detailed description and location of the error will be reported. Similarly, the exact number of the updated configuration parameters will be reported on a successful operation, together with the total number of the device configuration parameters that could have been updated.

For example, a syntax error in the *RemotePort* configuration parameter, when instead of *RemotePort* an incorrect *RemotePort** name is written, will result in the following error message, see Figure 20.

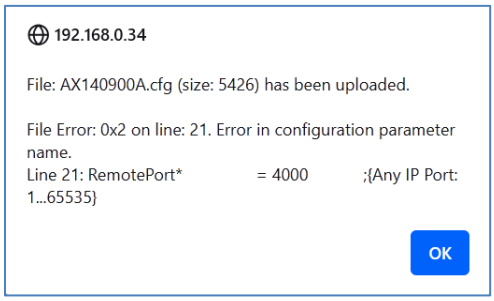


Figure 20. Loading System Configuration File Alert Message.
Error in Configuration File

This message contains a file error number, an error description, and a line number where this error was found in the configuration file. The file line content is also shown to facilitate debugging of the configuration file.

An error in the value of a configuration parameter is presented the same way. For example, if the *RemotePort* configuration parameter has an incorrect value of “400033”, the following error message will be generated, see Figure 21.

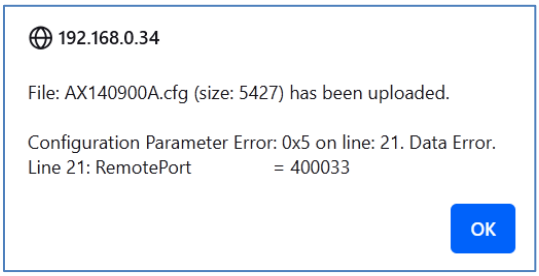


Figure 21. Loading System Configuration File Alert Message. Error in Configuration Parameter Value

After the configuration file has been parsed, the device checks integrity of the uploaded new configuration. Any errors will result in a data integrity error message. For example, if the *DeviceDefaultGateway* is set to “0.0.0.0” when the *DeviceSubnetMask* is equal to “255.255.255.255”, the following message will be generated, see Figure 22.

Figure 22. Loading System Configuration File Alert Message. Data Integrity Error

In case, for example, only 60 configuration parameters out of the total 61 updatable device configuration parameters have been updated, the result message will show this information, see Figure 23.

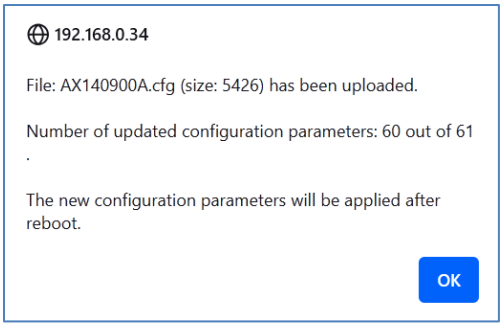


Figure 23. Loading System Configuration File Alert Message. Configuration Parameters Have Been Partially Updated

The user will need to reboot the device to apply the new configuration parameters already saved in non-volatile memory after a successful upload operation. This can be done by using the *Reboot* button on the *System Settings* web page.

All changes in configuration parameters will be rolled back if an upload operation has failed.

3.3.6.3 Restoring Default Settings

The user can restore the device to the factory default configuration by pressing the *Restore Default Settings* button on the *System Settings* page, see Figure 24.

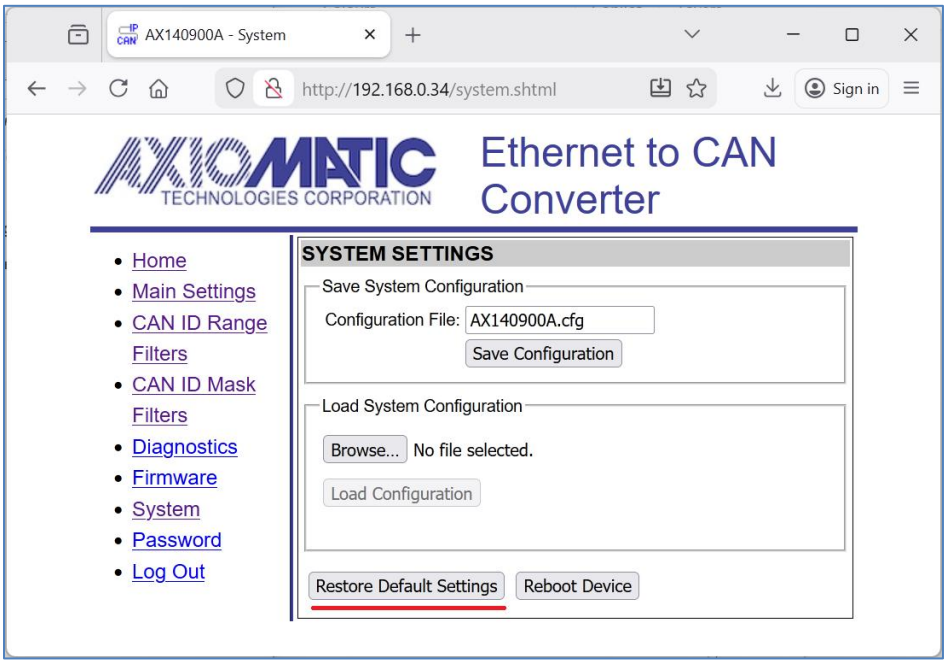


Figure 24 System Settings Page. Restore Default Settings

This operation complements the *Set Defaults* operation on configuration web pages. The main difference is that the *Restore Default Settings* operation restores all device configuration parameters, not only the ones presented on the configuration web page. The exception is the device password, which is not affected by the *Restore Default Settings* operation.

The confirmation alert message will appear to protect the device configuration from accidental modification, see Figure 25.

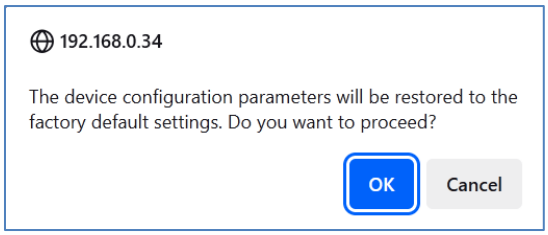


Figure 25. Restoring Default Settings. Confirmation Alert Message

If the user chooses to proceed, the second alert message will inform the user of the result of this operation, see Figure 26.

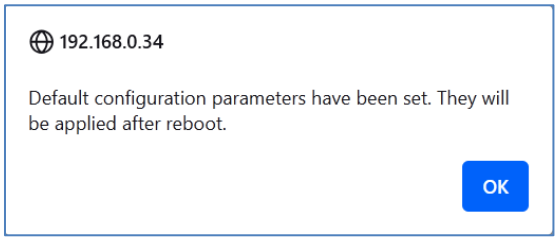


Figure 26. Restoring Default Setting. Successful Result Alert Message

3.4 Configuration File Format

The device configuration is stored in a human-readable text file based on a proprietary implementation of a well-known INI file format, https://en.wikipedia.org/wiki/INI_file.

The device configuration parameters are grouped in configuration parameter groups that form INI file sections. The user can edit a configuration file generated by the device in any text editor and change values of configuration parameters according to the user's requirements.

A device configuration file generated by the device with default configuration settings is presented in Figure 27.

```
; Device Configuration File
; -------------------------
; File format v1.00. Copyright (c) 2025 Axiomatic Technologies Corporation.

[Info]
; This information group was automatically generated by the device.
; All configuration parameters in the device "Info" group are read-only.
PartNumber                = AX140900A
SerialNumber              = 0000125001
MACAddr                   = B4:37:D1:A0:22:33
FirmwareID                = 25106
FirmwareVersionNumber     = 2.00

[Ethernet]
DeviceIpAddr              = 192.168.0.34    ;{Any IP Address: x.x.x.x. Restrictions apply}
DevicePort                = 4000            ;{Any IP Port: 1...65535. Restrictions apply}
PortType                  = UDP             ;{UDP, TCP}
RemoteIpAddr              = 192.168.0.35    ;{Any IP Address: x.x.x.x. Restrictions apply}
RemotePort                = 4000            ;{Any IP Port: 1...65535}
WebServerPort             = 80              ;{Any IP Port: 1...65535. Restrictions apply}
DeviceSubnetMask          = 255.255.255.0   ;{Any IP Subnet Mask: x.x.x.x, Restrictions apply}
DeviceDefaultGateway      = 192.168.0.1     ;{Any IP Address: x.x.x.x. Restrictions apply}
AutoConnectToRemote       = 0               ;{1-Yes, 0-No}

[CAN]
SwitchedPowerOut          = 0               ;{1-Yes, 0-No}
BaudRate                  = 250             ;{1000,667,500,250,125,100,83,50,20,10}
IDRangeFilter1OnOff       = 0               ;{1-On, 0-Off}
IDRangeFilter1EID         = 0               ;{1-Yes, 0-No}
IDRangeFilter1From        = 0x0             ;{If IDRangeFilter1EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter1To          = 0x0             ;{If IDRangeFilter1EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter2OnOff       = 0               ;{1-On, 0-Off}
IDRangeFilter2EID         = 0               ;{1-Yes, 0-No}
IDRangeFilter2From        = 0x0             ;{If IDRangeFilter2EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter2To          = 0x0             ;{If IDRangeFilter2EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter3OnOff       = 0               ;{1-On, 0-Off}
IDRangeFilter3EID         = 0               ;{1-Yes, 0-No}
IDRangeFilter3From        = 0x0             ;{If IDRangeFilter3EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter3To          = 0x0             ;{If IDRangeFilter3EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter4OnOff       = 0               ;{1-On, 0-Off}
IDRangeFilter4EID         = 0               ;{1-Yes, 0-No}
IDRangeFilter4From        = 0x0             ;{If IDRangeFilter4EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter4To          = 0x0             ;{If IDRangeFilter4EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter5OnOff       = 0               ;{1-On, 0-Off}
IDRangeFilter5EID         = 0               ;{1-Yes, 0-No}
IDRangeFilter5From        = 0x0             ;{If IDRangeFilter5EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilter5To          = 0x0             ;{If IDRangeFilter5EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDRangeFilterIEDMaskOnOff = 0               ;{1-On, 0-Off}
IDRangeFilterIEDMask      = 0x1fffffff      ;{0...0x1fffffff}
IDMaskFilter1OnOff        = 0               ;{1-On, 0-Off}
IDMaskFilter1EID          = 0               ;{1-Yes, 0-No}
IDMaskFilter1Mask         = 0x0             ;{If IDMaskFilter1EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDMaskFilter1ID           = 0x0             ;{If IDMaskFilter1EID=1, then 0...0x1fffffff, else
0...0x7ff}
```

```
IDMaskFilter2OnOff        = 0             ;{1-On, 0-Off}
IDMaskFilter2EID          = 0             ;{1-Yes, 0-No}
IDMaskFilter2Mask         = 0x0           ;{If IDMaskFilter2EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDMaskFilter2ID           = 0x0           ;{If IDMaskFilter2EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDMaskFilter3OnOff        = 0             ;{1-On, 0-Off}
IDMaskFilter3EID          = 0             ;{1-Yes, 0-No}
IDMaskFilter3Mask         = 0x0           ;{If IDMaskFilter3EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDMaskFilter3ID           = 0x0           ;{If IDMaskFilter3EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDMaskFilter4OnOff        = 0             ;{1-On, 0-Off}
IDMaskFilter4EID          = 0             ;{1-Yes, 0-No}
IDMaskFilter4Mask         = 0x0           ;{If IDMaskFilter4EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDMaskFilter4ID           = 0x0           ;{If IDMaskFilter4EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDMaskFilter5OnOff        = 0             ;{1-On, 0-Off}
IDMaskFilter5EID          = 0             ;{1-Yes, 0-No}
IDMaskFilter5Mask         = 0x0           ;{If IDMaskFilter5EID=1, then 0...0x1fffffff, else
0...0x7ff}
IDMaskFilter5ID           = 0x0           ;{If IDMaskFilter5EID=1, then 0...0x1fffffff, else
0...0x7ff}
CANChannelGroup           = 0             ;{0...255}
CANRxDChannel1            = 1             ;{0...32, Channel is undefined if 0}
CANRxDChannel2            = 0             ;{0...32, Channel is undefined if 0}
CANRxDChannel3            = 0             ;{0...32, Channel is undefined if 0}
CANRxDChannel4            = 0             ;{0...32, Channel is undefined if 0}
CANRxDChannel5            = 0             ;{0...32, Channel is undefined if 0}
CANTxDChannel             = 1             ;{0...32, Channel is undefined if 0}
CANLoopbackChannel        = 0             ;{0...32, Channel is undefined if 0}
```

Figure 27. Converter Configuration File. Default Configuration Parameters

The *[Info]* configuration parameter group is automatically generated by the device for information purposes only. It contains read-only configuration parameters. It is completely optional and can be omitted if the file is prepared by the user.

The *[Ethernet]* and *[CAN]* configuration parameter groups contain 9 and 52 updatable configuration parameters respectively. The user can change any of them.

For the user's convenience, the device automatically writes allowed values of an updatable configuration parameter in comments beside that configuration parameter when a configuration file is generated.

For detailed information on the updatable configuration parameters, please refer to:

- Table 1. Ethernet Configuration Parameters.
- Table 2. Main CAN Configuration Parameters.
- Table 3. Configuration Parameters of CAN ID Range Filters.
- Table 4. Configuration Parameters of CAN ID Mask Filters.

There are no specific restrictions on the number of configuration parameters in a configuration file. The file can have all or just one configuration parameter provided that the configuration parameter group of the configuration parameter is also specified. This allows creation of a configuration file that changes only a specific set of configuration parameters without affecting all other settings.

For example, a configuration file that only changes the *Remote Port* value to 4001 is presented in Figure 28.

```
; Device Configuration File
; -------------------------
; This file will set the Remote Port
```

```
[Ethernet]
RemotePort              = 4001          ;{Any IP Port: 1...65535}
```

Figure 28. Device Configuration File to Update Remote Port Value

3.5 Password Update Web Page

The device password can be changed on the *Password Update* web page by clicking on the *Password* link on the left side of the device website, see Figure 29.

To update the device password, the user should enter the current passwords and then enter and confirm the new password.

The password should contain at least one number, one uppercase and one lowercase English letter. Special characters are allowed[1]. Spaces are allowed in the middle of the password. The password length should be from 8 to 30 characters. The new password should be different from the old one. The user will be prompted to follow the password rules in case any of the password requirements are not met.

[1]Column (:) was not allowed in firmware V1.xx. It still can be used in firmware V1.xx if the firmware was downgraded from a later version. The password is preserved when upgrading or downgrading the firmware.

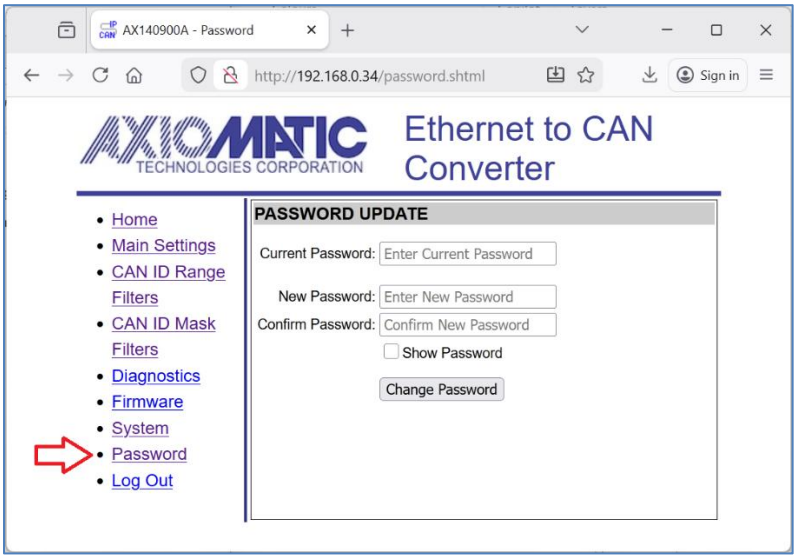


Figure 29. Password Update Web Page

The user can see all entered passwords when the *Show Password* checkbox is selected.

The result of the password update operation will be shown to the user in an alert message from the device web site after pressing the *Change Password* button, see Figure 30.

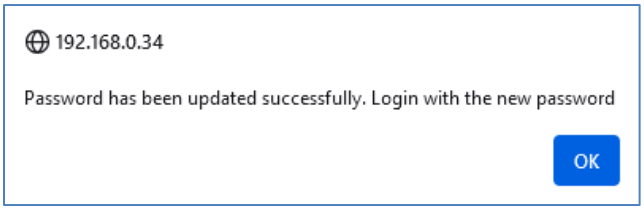


Figure 30. Password Update Alert Message

After the password has been changed successfully, the user will be automatically logged out and prompted to login again using the new password.

4 RESTFUL INTERFACE

For exchanging information with other devices on the IP network, the Ethernet to CAN converter supports a RESTful interface (REST API) that utilizes web services for intercomputer communication.

The RESTful interface allows external devices to read and modify the converter configuration parameters, receive diagnostic information, upload or download the converter configuration file, and update the converter firmware in the field. The user can do the same operations available on the device website using the RESTful interface except of changing the device password, which can be done only on the device website for security reasons.

The RESTful interface is based on a client-server model. A client is an external computer system, and a server is the internal web server on the Ethernet to CAN converter running the REST API. The client sends standard HTTP requests to the server, and the server responds with requested data or performs necessary actions based on the client's request.

For request and response encoding, the RESTful interface mainly uses JSON data format described in ECMA-404.

All API resource URL paths start with `/api` prefix. They are followed by `/v1` prefix for the current version of the RESTful interface, followed by the application resource name with optional resource sub-names. For example, a URL path: `/api/v1/config/ethernet` is used for operations on the converter *Ethernet* configuration parameters.

All supported URLs respond to GET, HEAD, PUT, POST, OPTIONS, and DELETE request methods, even if the method is not implemented. For unsupported methods and non-API requests, the RESTful interface will return 501 Not Implemented error.

For unsupported resources, the RESTful interface will return 404 Not Found error.

HEAD and OPTIONS methods have a standard implementation. For request methods that are not implemented on a supported resource, the RESTful interface will return 405 Method Not Allowed error.

The RESTful interface requires Basic HTTP authentication, as per RFC 7617. On request to `Basic realm="api"`, the client should provide a user-id equal to 'admin' and a password equal to the current password of the device website, see *Connecting to the Device* subsection of this document. The OPTIONS method does not require a user authentication.

4.1 Device Configuration

The user can retrieve or change the device configuration using RESTful operations on the device configuration resources, see Table 5, Table 6, and Table 7. The configuration resources present the same configuration parameter groups as for the configuration file, see *Configuration File Format* subsection of this document.

All updated configuration parameters are stored in non-volatile memory and are applied after the device reset.

Table 5. RESTful Interface. Device Configuration Resource

Resource URL Path	/api/v1/config
Resource Description	All configuration parameters of the device
Request Name	**Get Device Configuration Parameters**
Description	Requests configuration parameters from all device configuration parameter groups
Method	GET
Request Body	No Data
Response	200 OK
Response Body	Configuration parameters from all device configuration parameter groups. Content-Type: application/json. For example:


```
{
    "Info": {
        "PartNumber": "AX140900A",
        "SerialNumber": "0000125001",
        "MACAddr": "B4:37:D1:A0:22:33",
        "FirmwareID": 25106,
        "FirmwareVersionNumber": "1.00"
    },
    "CAN": {
        "SwitchedPowerOut": 0,
        "BaudRate": 250,
        "IDRangeFilters": {
            "Filters": [
                {
                    "OnOff": 0,
                    "EID": 0,
                    "From": "0x0",
                    "To": "0x0"
                },
                {
                    "OnOff": 0,
                    "EID": 0,
                    "From": "0x0",
                    "To": "0x0"
                },
                {
                    "OnOff": 0,
                    "EID": 0,
                    "From": "0x0",
                    "To": "0x0"
                },
                {
                    "OnOff": 0,
                    "EID": 0,
                    "From": "0x0",
                    "To": "0x0"
                },
                {
                    "OnOff": 0,
                    "EID": 0,
                    "From": "0x0",
                    "To": "0x0"
                }
            ],
            "EIDMaskOnOff": 0,
            "EIDMask": "0x1fffffff"
        },
        "IDMaskFilters": {
            "Filters": [
                {
                    "OnOff": 0,
                    "EID": 0,
                    "Mask": "0x0",
                    "ID": "0x0"
                },
                {
                    "OnOff": 0,
                    "EID": 0,
                    "Mask": "0x0",
                    "ID": "0x0"
```

	{
	"IDRangeFilters": [
	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	}
	]
	},
	"CANChannelGroup": 0,
	"CANRxDChannels": [
	1,
	0,
	0,
	0,
	0
	],
	"CANTxDChannel": 1,
	"CANLoopbackChannel": 0
	},
	"Ethernet": {
	"DeviceIpAddr": "192.168.0.34",
	"DevicePort": 4000,
	"PortType": "UDP",
	"RemoteIpAddr": "192.168.0.35",
	"RemotePort": 4000,
	"WebServerPort": 80,
	"DeviceSubnetMask": "255.255.255.0",
	"DeviceDefaultGateway": "192.168.0.1",
	"AutoConnectToRemote": 0
	}
	}

Request Name	**Modify Device Configuration Parameters**
Description	Modifies all or a specific set of the device configuration parameters
Method	PUT
Request Body	Configuration parameters that need to be modified in all device configuration parameter groups, see response body to GET method on this resource for an example. Content-Type: application/json. The user can submit only those parameters that need to be modified. IDRangeFilters, IDMaskFilters, and CANRxDChannels will be entirely updated if submitted.
Response	200 OK

Response Body	Summary of the operation. Content-Type: application/json. For example:
	{
	"UpdatedConfigParam": 61,
	"ReadOnlyConfigParam": 5,
	"UnknownConfigParam": 0,
	"TotalUpdatableConfigParam": 61
	}

Response on Error	400 Bad Request

Response Body on Error	Error message. Content-Type: application/json. For Example:
	{
	"Error": "Incorrect value of: CAN.BaudRate"
	}

Table 6. RESTful Interface. Device Configuration. Information Group Resource

Resource URL Path	/api/v1/config/info
Resource Description	Configuration parameters of the device information group

Request Name	**Get Information Parameters**
Description	Requests configuration parameters of the device information group
Method	GET
Request Body	No Data
Response	200 OK

Response Body	Configuration parameters of the device information group. Content-Type: application/json. For example:
	`{`
	`"PartNumber": "AX140900A",`
	`"SerialNumber": "0000125001",`
	`"MACAddr": "B4:37:D1:A0:22:33",`
	`"FirmwareID": 25106,`
	`"FirmwareVersionNumber": "1.00"`
	`}`

Request Name	**Modify Information Parameters**
Description	This request is kept for consistency with modification requests to other device configuration resources. It cannot modify any information parameters, since all of them are read-only parameters.
Method	PUT
Request Body	Configuration parameters of the device information group. See response body to GET method on this resource for an example. Content-Type: application/json.
Response	200 OK

Response Body	Summary of the operation. Content-Type: application/json. For example:
	`{`
	`"UpdatedConfigParam": 0,`
	`"ReadOnlyConfigParam": 5,`
	`"UnknownConfigParam": 0,`
	`"TotalUpdatableConfigParam": 0`
	`}`

Table 7. RESTful Interface. Device Configuration. Ethernet Configuration Group Resource

Resource URL Path	/api/v1/config/ethernet
Resource Description	Configuration parameters of the device Ethernet group
Request Name	**Get Ethernet Configuration Parameters**
Description	Requests configuration parameters of the device Ethernet group
Method	GET
Request Body	No Data
Response	200 OK

Response Body	Configuration parameters of the device Ethernet group. Content-Type: application/json. For example:
	`{`
	`"DeviceIpAddr": "192.168.0.34",`
	`"DevicePort": 4000,`
	`"PortType": "UDP",`
	`"RemoteIpAddr": "192.168.0.35",`
	`"RemotePort": 4000,`
	`"WebServerPort": 80,`
	`"DeviceSubnetMask": "255.255.255.0",`
	`"DeviceDefaultGateway": "192.168.0.1",`
	`"AutoConnectToRemote": 0`
	`}`

Request Name	**Modify Ethernet Configuration Parameters**
Description	Modifies all or a specific set of configuration parameters in the device Ethernet group
Method	PUT
Request Body	Configuration parameters that need to be modified in the device Ethernet group. See response body to GET method on this resource for an example. Content-Type: application/json.
Response	200 OK

Response Body	Summary of the operation. Content-Type: application/json. For example:
	{
	"UpdatedConfigParam": 9,
	"ReadOnlyConfigParam": 0,
	"UnknownConfigParam": 0,
	"TotalUpdatableConfigParam": 9
	}

Response on Error	400 Bad Request

Response Body on Error	Error message. Content-Type: application/json. For Example:
	{
	"Error": "Incorrect value of: DevicePort"
	}

Table 8. RESTful Interface. Device Configuration. CAN Configuration Group Resource

Resource URL Path	/api/v1/config/can
Resource Description	Configuration parameters of the device CAN group
Request Name	**Get CAN Configuration Parameters**
Description	Requests configuration parameters of the device CAN group
Method	GET
Request Body	No Data
Response	200 OK

Response Body	Configuration parameters of the device CAN group. Content-Type: application/json. For example:
	{
	"SwitchedPowerOut": 0,
	"BaudRate": 250,
	"IDRangeFilters": {
	"Filters": [
	{
	"OnOff": 0,
	"EID": 0,
	"From": "0x0",
	"To": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"From": "0x0",
	"To": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"From": "0x0",
	"To": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"From": "0x0",
	"To": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"From": "0x0",
	"To": "0x0"
	}
	],
	"EIDMaskOnOff": 0,
	"EIDMask": "0x1fffffff"
	},
	"IDMaskFilters": {
	"Filters": [
	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	},

	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	},
	{
	"OnOff": 0,
	"EID": 0,
	"Mask": "0x0",
	"ID": "0x0"
	}
	]
	},
	"CANChannelGroup": 0,
	"CANRxDChannels": [
	1,
	0,
	0,
	0,
	0
	],
	"CANTxDChannel": 1,
	"CANLoopbackChannel": 0
	}

Request Name	**Modify CAN Configuration Parameters**
Description	Modifies all or a specific set of configuration parameters in the device CAN group
Method	PUT
Request Body	Configuration parameters that need to be modified in the device CAN group. See response body to GET method on this resource for an example. Content-Type: application/json.
Response	200 OK

Response Body	Summary of the operation. Content-Type: application/json. For example:
	{
	"UpdatedConfigParam": 9,
	"ReadOnlyConfigParam": 0,
	"UnknownConfigParam": 0,
	"TotalUpdatableConfigParam": 9
	}

The user can modify only the entire set of CAN ID Range or CAN ID Mask filters. Issuing, for example, *Modify CAN Configuration Parameters* request with the request body presented in Figure 31, will result in: first erasing all CAN ID Range filters, disabling CAN EID Mask, and then setting the first CAN ID Range filter to the specified in the request `{OnOff=1,EID=0,From=0x1,To="0x6"}` parameters.

{
"IDRangeFilters": {
"Filters": [
{
"OnOff": 1,
"EID": 0,
"From": "0x1",
"To": "0x6"
}
]
}

```
}
```

Figure 31. CAN ID Range Filter Setting Example

The same applies to modification of CAN RxD Channels. The user can modify only the entire channel set on the device with a new channel set presented in *Modify Device Configuration Parameters* or *Modify CAN Configuration Parameters* request.

4.2 Configuration file

The user can upload a configuration file with a new device configuration or download a configuration file with existing device configuration using RESTful operations on the device *Configuration File* resource, see Table 9. The new device configuration will be applied after the device reset.

Table 9. RESTful Interface. Configuration File Resource

Resource URL Path	/api/v1/config-file
Resource Description	Configuration file resource
Request Name	**Download Configuration File**
Description	Downloads the device configuration file with the existing device configuration
Method	GET
Request Body	No Data
Response	200 OK
Response Body	Device configuration file content. Content-Type: text/plain. For example:

```
; Device Configuration File
; -------------------------
; File format v1.00. Copyright (c) 2025 Axiomatic Technologies Corporation.

[Info]
; This information group was automatically generated by the device.
; All configuration parameters in the device "Info" group are read-only.
PartNumber                = AX140900A
SerialNumber              = 0000125001
MACAddr                   = B4:37:D1:A0:22:33
FirmwareID                = 25106
FirmwareVersionNumber     = 1.00

[Controller]

[Ethernet]
DeviceIpAddr              = 192.168.0.34    ;{Any IP Address: x.x.x.x.
Restrictions apply}
DevicePort                = 4000            ;{Any IP Port: 1...65535.
Restrictions apply}
PortType                  = UDP             ;{UDP, TCP}
RemoteIpAddr              = 192.168.0.35    ;{Any IP Address: x.x.x.x.
Restrictions apply}
RemotePort                = 4000            ;{Any IP Port: 1...65535}
WebServerPort             = 80              ;{Any IP Port: 1...65535.
Restrictions apply}
DeviceSubnetMask          = 255.255.255.0   ;{Any IP Address: x.x.x.x, as per
RFC 1878}
DeviceDefaultGateway      = 192.168.0.1     ;{Any IP Address: x.x.x.x.
Restrictions apply}
AutoConnectToRemote       = 0               ;{1-Yes, 0-No}

[CAN]
SwitchedPowerOut          = 0               ;{1-Yes, 0-No}
BaudRate                  = 250
;{1000,667,500,250,125,100,83,50,20,10}
IDRangeFilter1OnOff       = 0               ;{1-On, 0-Off}
IDRangeFilter1EID         = 0               ;{1-Yes, 0-No}
IDRangeFilter1From        = 0x0             ;{If IDRangeFilter1EID=1, then
0...0x1fffffff, else 0...0x7ff}
IDRangeFilter1To          = 0x0             ;{If IDRangeFilter1EID=1, then
0...0x1fffffff, else 0...0x7ff}
```

	IDRangeFilter2OnOff = 0 ;{1-On, 0-Off} IDRangeFilter2EID = 0 ;{1-Yes, 0-No} IDRangeFilter2From = 0x0 ;{If IDRangeFilter2EID=1, then 0...0x1fffffff, else 0...0x7ff} IDRangeFilter2To = 0x0 ;{If IDRangeFilter2EID=1, then 0...0x1fffffff, else 0...0x7ff} IDRangeFilter3OnOff = 0 ;{1-On, 0-Off} IDRangeFilter3EID = 0 ;{1-Yes, 0-No} IDRangeFilter3From = 0x0 ;{If IDRangeFilter3EID=1, then 0...0x1fffffff, else 0...0x7ff} IDRangeFilter3To = 0x0 ;{If IDRangeFilter3EID=1, then 0...0x1fffffff, else 0...0x7ff} IDRangeFilter4OnOff = 0 ;{1-On, 0-Off} IDRangeFilter4EID = 0 ;{1-Yes, 0-No} IDRangeFilter4From = 0x0 ;{If IDRangeFilter4EID=1, then 0...0x1fffffff, else 0...0x7ff} IDRangeFilter4To = 0x0 ;{If IDRangeFilter4EID=1, then 0...0x1fffffff, else 0...0x7ff} IDRangeFilter5OnOff = 0 ;{1-On, 0-Off} IDRangeFilter5EID = 0 ;{1-Yes, 0-No} IDRangeFilter5From = 0x0 ;{If IDRangeFilter5EID=1, then 0...0x1fffffff, else 0...0x7ff} IDRangeFilter5To = 0x0 ;{If IDRangeFilter5EID=1, then 0...0x1fffffff, else 0...0x7ff} IDRangeFilterIEDMaskOnOff = 0 ;{1-On, 0-Off} IDRangeFilterIEDMask = 0x1fffffff ;{0...0x1fffffff} IDMaskFilter1OnOff = 0 ;{1-On, 0-Off} IDMaskFilter1EID = 0 ;{1-Yes, 0-No} IDMaskFilter1Mask = 0x0 ;{If IDMaskFilter1EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter1ID = 0x0 ;{If IDMaskFilter1EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter2OnOff = 0 ;{1-On, 0-Off} IDMaskFilter2EID = 0 ;{1-Yes, 0-No} IDMaskFilter2Mask = 0x0 ;{If IDMaskFilter2EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter2ID = 0x0 ;{If IDMaskFilter2EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter3OnOff = 0 ;{1-On, 0-Off} IDMaskFilter3EID = 0 ;{1-Yes, 0-No} IDMaskFilter3Mask = 0x0 ;{If IDMaskFilter3EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter3ID = 0x0 ;{If IDMaskFilter3EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter4OnOff = 0 ;{1-On, 0-Off} IDMaskFilter4EID = 0 ;{1-Yes, 0-No} IDMaskFilter4Mask = 0x0 ;{If IDMaskFilter4EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter4ID = 0x0 ;{If IDMaskFilter4EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter5OnOff = 0 ;{1-On, 0-Off} IDMaskFilter5EID = 0 ;{1-Yes, 0-No} IDMaskFilter5Mask = 0x0 ;{If IDMaskFilter5EID=1, then 0...0x1fffffff, else 0...0x7ff} IDMaskFilter5ID = 0x0 ;{If IDMaskFilter5EID=1, then 0...0x1fffffff, else 0...0x7ff} CANChannelGroup = 0 ;{0...255} CANRxDChannel1 = 1 ;{0...32, Channel is undefined if 0} CANRxDChannel2 = 0 ;{0...32, Channel is undefined if 0} CANRxDChannel3 = 0 ;{0...32, Channel is undefined if 0} CANRxDChannel4 = 0 ;{0...32, Channel is undefined if 0} CANRxDChannel5 = 0 ;{0...32, Channel is undefined if 0} CANTxDChannel = 1 ;{0...32, Channel is undefined if 0} CANLoopbackChannel = 0 ;{0...32, Channel is undefined if 0}
Request Name	**Upload Configuration File**
Description	Uploads the device configuration file in the device non-volatile memory. The new device configuration will be applied after the device reset.

Method	PUT
Request Body	Configuration file content. Content-Type: form-data
Response	200 OK

Response Body	Summary of the operation. Content-Type: application/json. For example:
	{
	"Error": "Ok",
	"FileName": "AX140900A(.cfg",
	"FileSize": 5425,
	"UpdatedConfigParam": 61,
	"TotalUpdatableConfigParam": 61
	}

Response on Error	400 Bad Request

Response Body on Error	Error message. Content-Type: application/json. For Example:
	{
	"Error": "File has not been uploaded",
	}

4.3 Diagnostics

The user can monitor the internal state of the device using RESTful operations on the device diagnostic resource, see Table 10.

Table 10. RESTful Interface. Diagnostics Resource

Resource URL Path	/api/v1/diagnostics
Resource Description	Device diagnostic resource
Request Name	**Get Device Diagnostics**
Description	Retrieves immediate state of the device diagnostic parameters
Method	GET
Request Body	No Data
Response	200 OK

Response Body	The device diagnostic data. Content-Type: application/json. For example:
	{
	"General": {
	"HealthStatus": "Normal",
	"HealthStatusDescription": ""
	},
	"System": {
	"PowerSupplyVoltage": "11.33",
	"uCTemp": "34.63",
	"SwitchedPowerOutVoltage": "11.30"
	},
	"Ethernet": {
	"NumberOfActiveConnections": 0,
	"NumberOfAvailableConnections": 10,
	"IsRemoteDeviceConnected": 0,
	"EthMessagesRxD": "1882",
	"EthMessagesTxD": "254"
	},
	"CAN": {
	"CANRxDErrors": "0",
	"CANTxDErrors": "0",
	"CANBusOffErrors": "0",
	"CANFramesRxD": "0",
	"CANFramesTxD": "0"
	}
	}

Due to volatile nature of the diagnostic data, the Content-Length value returned in the response to HEAD request can be different from the Content-Length value returned in subsequent GET requests to the device diagnostic resource.

The statistical counters in the diagnostic resource can be reset by posting “ResetDiagnosticCounters” command, see Table 13.

4.4 New Firmware

The new firmware can be uploaded to the device using RESTful operations on the device *New Firmware* resource, see Table 11.

Table 11. RESTful Interface. New Firmware Resource

Resource URL Path	/api/v1/new-firmware
Resource Description	Device new firmware resource
Request Name	**Upload New Firmware**
Description	Uploads new firmware in the device non-volatile memory
Method	POST
Request Body	The new firmware file content. Content-Type: form-data
Response	201 Created

Response Body	Summary of the operation. Content-Type: application/json. For example:
	`{`
	`"FileName": "AF-25106-1.00.af",`
	`"FirmwareVersionNumber": "1.00",`
	`"FirmwareID": 25106,`
	`"FirmwareComments": "Initial Release"`
	`}`

Response on Error	400 Bad Request

Response Body on Error	Error message. Content-Type: application/json. For Example:
	`{`
	`"Error": "Incorrect File Format"`
	`}`

Request Name	**Get New Firmware Information**
Description	Retrieves information of the uploaded new firmware. This information will be available until the new firmware is programmed into the device microcontroller on the next device reset.
Method	GET
Request Body	No Data
Response	200 OK
Response Body	Summary of the uploaded new firmware. It is the same data returned by the device on successful POST operation. Content-Type: application/json.
Response on Error	400 Bad Request
Response Body on Error	Error message if the new firmware has not been uploaded. It is the same data returned by the device on unsuccessful POST operation. Content-Type: application/json.
Request Name	**Delete New Firmware**
Description	Deletes the new uploaded firmware from the device non-volatile memory. The new uploaded firmware will be kept in the device non-volatile memory until the new firmware is programmed into the device microcontroller on the next device reset.
Method	DELETE
Request Body	No Data
Response	204 No Content
Response Body	No Data
Response on Error	400 Bad Request

Response Body on Error	Error message if the new firmware has not been deleted. Content-Type: application/json. For Example:
	`{`
	`"Error": "No new firmware to delete"`
	`}`

The uploaded firmware will be kept in non-volatile memory and programmed into the device microcontroller after the device reset. The user can delete this firmware from non-volatile memory after uploading by posting the *Delete New Firmware* request, see Table 11.

The programming of the new firmware after reset can take up to a minute (usually less than 30s) during which the device will be off-line and will not accept any RESTful commands.

4.5 Commands

The RESTful interface allows users to send commands to the device by creating a new *Command* resource for a specific command, see Table 12.

Table 12. RESTful Interface. Command Resource

Resource URL Path	/api/v1/command
Resource Description	Device command resource
Request Name	**Send Command**
Description	Sends a command to the device
Method	POST

Request Body	The device command name. Content-Type: application/json. For example:
	`{`
	`  "Command": "RestoreDefaultSettings"`
	`}`

Response	204 No Content
Response Body	No Data
Response on Error	400 Bad Request

Response Body on Error	Error message. Content-Type: application/json. For Example:
	`{`
	`  "Error": "Unknown Command"`
	`}`

Request Name	**Get Command Information**
Description	Retrieves information on the last processed command and a list of supported device commands
Method	GET
Request Body	No Data
Response	200 OK or 202 Accepted[1]

Response Body	Summary of the last processed device command. The request also contains a list of supported device commands. Content-Type: application/json. For example:
	`{`
	`  "LastCommand": "RestoreDefaultSettings",`
	`  "LastError": "Ok",`
	`  "AvailableCommands": [`
	`  "RestoreDefaultSettings",`
	`  "Reboot",`
	`  "ResetDiagnosticCounters"`
	`  ]`
	`}`

[1]202 Accepted is for “Reboot” request.

The following commands are supported by the current version of the device firmware, see Table 13. The list of supported device commands can be retrieved by posting the *Get Command Information* request, see Table 12.

Table 13. RESTful Interface Commands

RESTful Command Name	Description
RestoreDefaultSettings	Restores default configuration parameters of the device. The default configuration parameters will be applied after the device reset
Reboot	Reboots the device immediately after responding to the RESTful request
ResetDiagnosticCounters	Resets the device diagnostic counters

5 CONVERTER DIAGNOSTICS

The user can see real-time diagnostic information on the *Diagnostics* page on the converter internal website.

To see the *Diagnostics* page, Figure 32, the user should click on the *Diagnostics* link on the left side of the website.

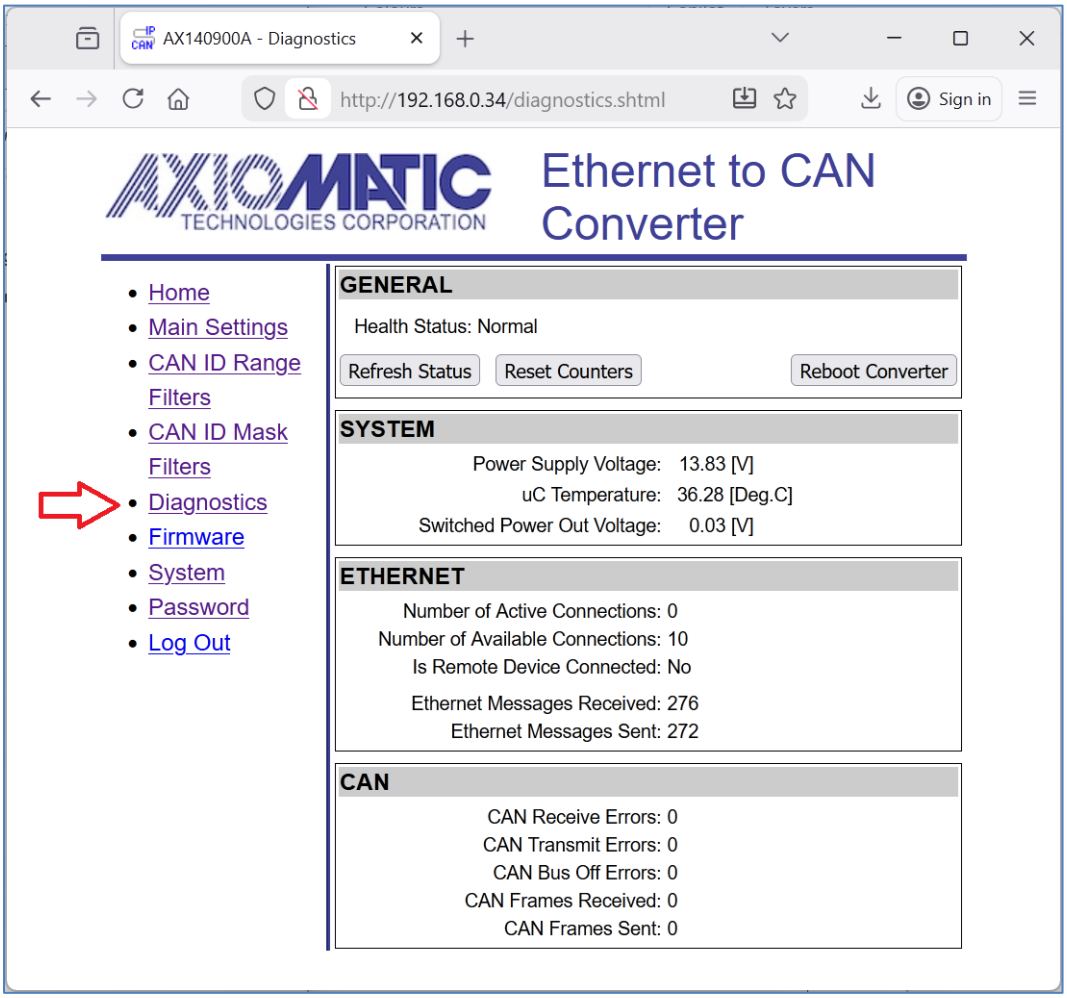


Figure 32. The Converter Diagnostics Page

The *Diagnostics* page shows the *Health Status* of the converter together with the *System*, *Ethernet* and *CAN* status information.

The user can refresh the values on the page by pressing the *Refresh Status* button or reset the statistical counters by pressing *Reset Counters* button. The *Reboot Converter* button activates the converter rebooting.

5.1 Health Status

The converter *Health Status* is an aggregated system run-time parameter calculated on the base of operational statuses of the major device hardware and software components.

The *Health Status* presents the overall operational status of the *CAN to Ethernet Converter*, based on the following rules, see Table 14.

Table 14. Health Status

Health Status	Condition
Error	“Error” is reported when at least one operational status is in “Error” state.
Warning	“Warning” is reported when at least one operational status is in “Warning” state and there are no operational statuses in “Error” state.
Undefined	“Undefined” is reported when at least one operational status is in “Undefined” state and there are no operational statuses in “Error” or “Warning” state.
Normal	“Normal” is reported when all operational statuses are in “Normal” state.

If the *Health Status* is different from “Normal”, the user will see a verbose message on the *Diagnostics* web page below the *Health Status* describing which operational status is causing a problem, see Figure 33.

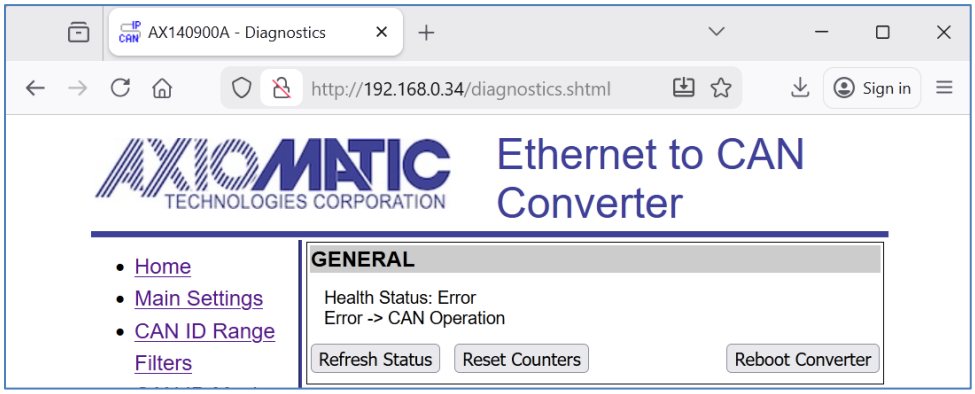


Figure 33. Health Status Message on CAN Error

In case several operational statuses differ from “Normal”, all of them will be shown on the *Diagnostics* page.

5.2 Converter Rebooting

The user can reboot the converter, when necessary, using the *Reboot Converter* button.

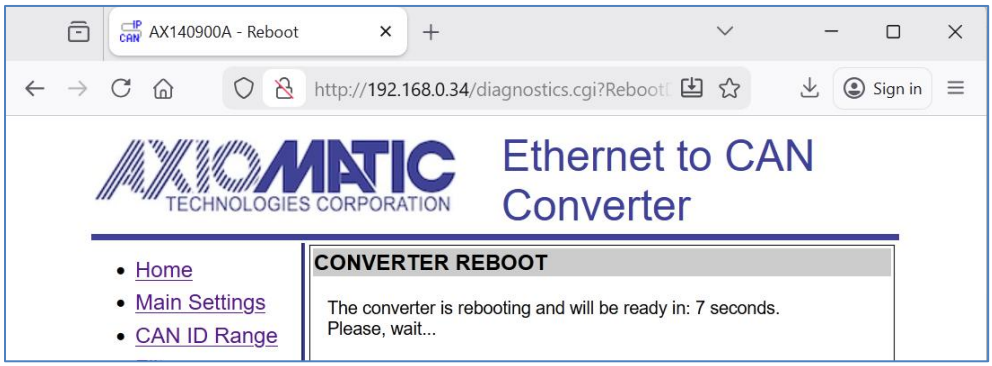


Figure 34. The Converter Reboot Screen

The rebooting operation takes 10 seconds. The user will see the *Reboot* screen with a countdown during this operation, see Figure 34.

When rebooting is over, the device home page will be loaded.

6 FIRMWARE UPDATE

The converter firmware can be updated through the converter internal website in the field.

The update procedure is performed in two stages. First, the application firmware is uploaded into the device internal flash. During this stage, the converter checks the firmware checksum and other conditions to determine whether it can be programmed further into the device microcontroller.

Then, upon the user confirmation, the firmware is programmed into the device microcontroller, and the device is restarted. At the end of this process, the user should see the new firmware version number on the converter home page in the web browser.

The device power should not be interrupted during firmware update to avoid possible corruption of nonvolatile memory.

6.1 Uploading the New Firmware

To upload the new firmware, the user should activate the *Firmware Uploading* page, see Figure 35, by clicking on the *Firmware* link on the left side of the website[1].

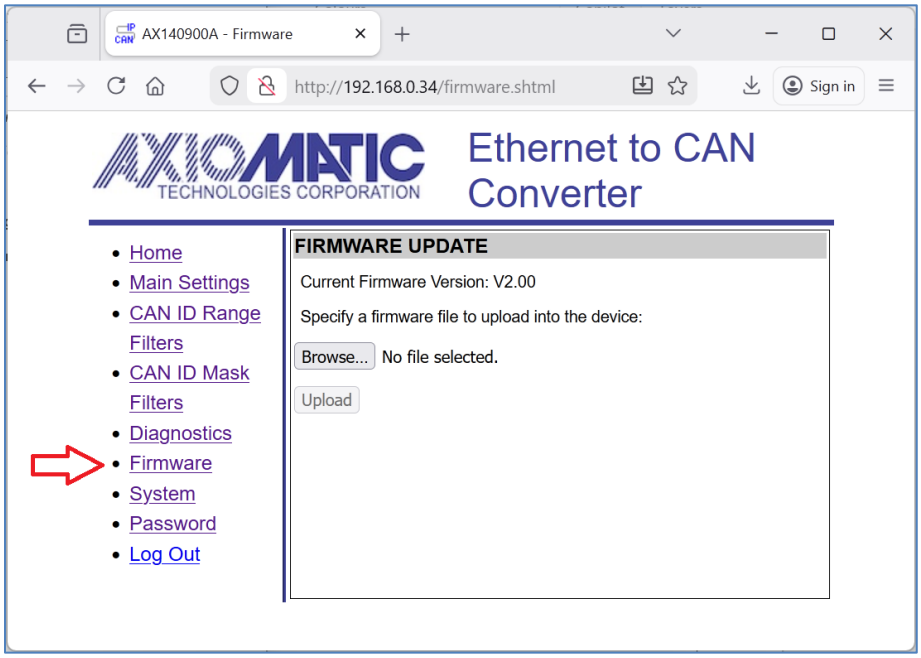


Figure 35. Firmware Uploading Page[1]

[1]The Current Firmware Version number may be different from the firmware version described in the manual.

Then the user selects the new firmware file using the *Browse…* button.

The firmware file is provided by Axiomatic in a proprietary binary format with extension: `.af`. The file name should have the following format: `AF- 25106-X.XX.af`, where the `<X.XX>` field wildcard reflects the firmware version number[2].

[2]`AF- 25106-2.00.af` file will be used for illustration of the firmware update process in this manual.

When the file is selected, the user should press the *Upload* button. The user will see the dynamic message: “Loading…” at the bottom of the screen and then, if everything is in order, the converter will switch automatically to the *Firmware Update* page.

6.2 Applying New Firmware

On the *Firmware Update* page, the user will see the new firmware file information, see Figure 36.

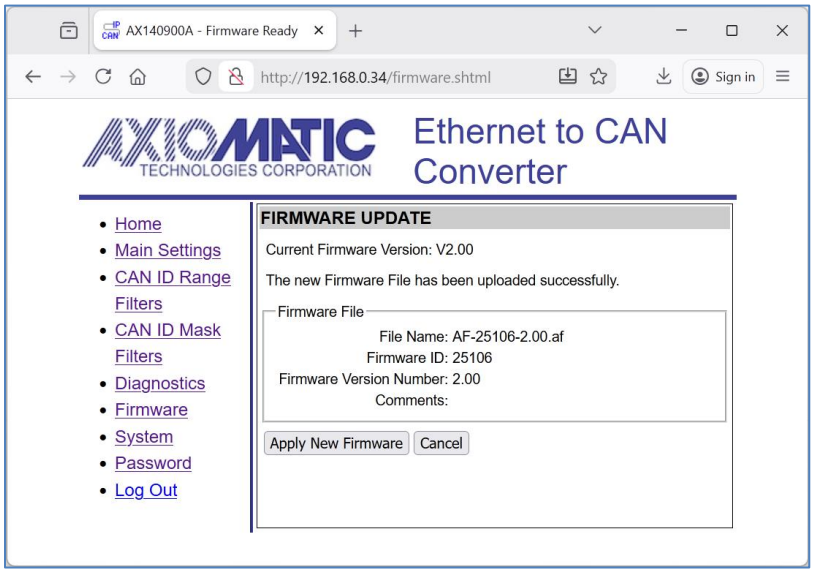


Figure 36. Firmware Update Page

From this point, the user can cancel the firmware update process and keep the old firmware or proceed with flashing the new firmware into the microcontroller by pressing the *Apply New Firmware* button.

When the user presses the *Apply New Firmware* button, the firmware update process is activated, and the *Firmware Upload* page will show the countdown timer, see Figure 37.

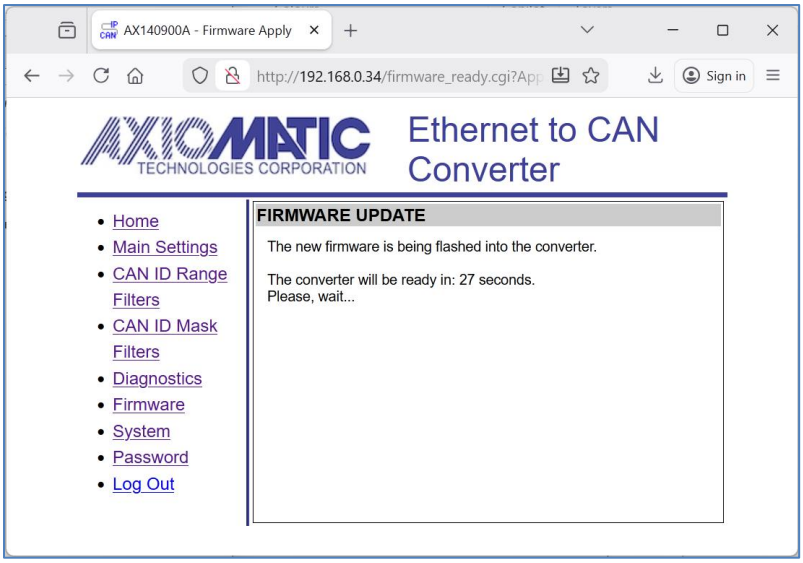


Figure 37. Firmware Update Countdown has been Started

The countdown timer is set for 30 seconds necessary to complete the flashing process and reboot the device.

The device home page will be displayed after rebooting. The user will see the new application firmware version number in the *Device Information* section on the converter home page[1], see Figure 38.

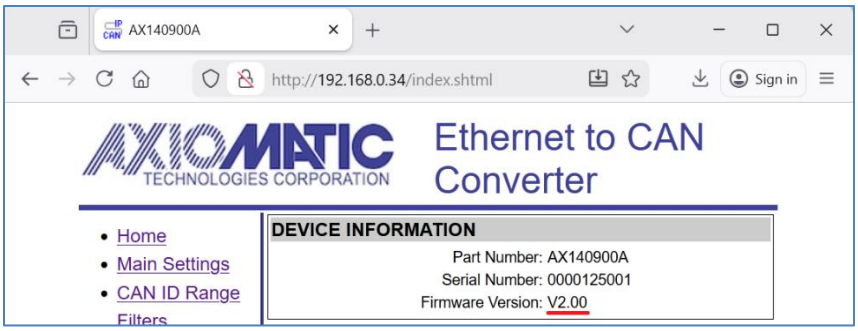


Figure 38. Firmware Version Number After Flashing

[1]In our example, it is the same 2.00 version number since we used firmware version 2.00 to illustrate the firmware update process.

When the firmware is upgraded, all configuration parameters new to that version will take default values. The password is preserved when upgrading or downgrading the firmware.

7 CONVERTER DEPLOYMENT

There are two major approaches in using the Ethernet to CAN converter. One is to use the converter on its own as a CAN extender (or repeater), a bridge or a baud rate converter.

For example, a pair of converters connected through the Ethernet port can bridge together two remote CAN networks. This example can be extended to several CAN networks running at different baud rates in various remote locations, connected together using Ethernet to CAN converters through LAN.

The second approach is to use the converter together with other IP devices that can directly communicate with the converter over IP network. This approach requires writing a custom software for interfacing with the converter.

Since the converter uses an Axiomatic proprietary communication protocol [1], Axiomatic provides CAN-ENET Software Support Package (SSP), p/n AX140910, for interfacing with the converter. The SSP is downloadable from www.axiomatic.com, log-in section. Version 3.0.0 or higher that supports CAN frame routing should be used.

The majority of Axiomatic PC software tools support the Ethernet to CAN converter. They can connect to CAN bus using this converter the same way as they connect to CAN bus using other Axiomatic Ethernet to CAN or Wi-Fi to CAN converters. The Axiomatic Electronic Assistant (EA) can communicate with the converter starting from version 5.11.82.0 and supports CAN frame routing starting from 5.16.133.0. CAN Assistant – Scope and CAN Assistant – Visual support the converter starting from version 3.0.0 and support CAN frame routing starting from version 4.0.0.

The CAN port routing address of the Ethernet to CAN converter should be set to (Group=0, Channel=1) to communicate with Axiomatic PC software in case the software version does not support the CAN frame routing. The *Converter Type* should be set to *Axiomatic Ethernet-CAN Converter* in the PC software.

The use of the Ethernet to CAN converter for bridging CAN networks with or without baud rate conversion is described below. There is no need for custom software for this type of the converter deployment.

7.1 CAN Network Bridging

To bridge two remote CAN networks, the user connects Ethernet to CAN converters one to each of the CAN networks and configure the converters the way that they will talk with each other over Ethernet ports through a local or global IP network, see Figure 39 and Figure 40.

If no CAN filters are activated, this setting simply extends CAN network allowing devices on both CAN networks to communicate with each other as if they were located on one virtual CAN network.

In the simplest scenario, two pre-configured converters can be connected by an Ethernet cable, see Figure 41. Due to the Auto-MDIX feature, either a straight or a crossover cable can be used.

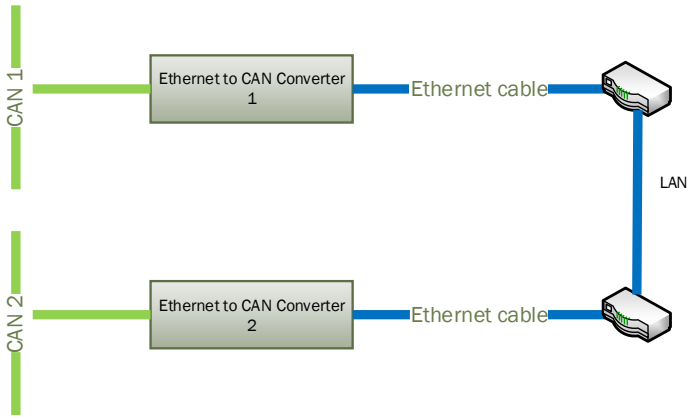


Figure 39. Local Connection through the LAN

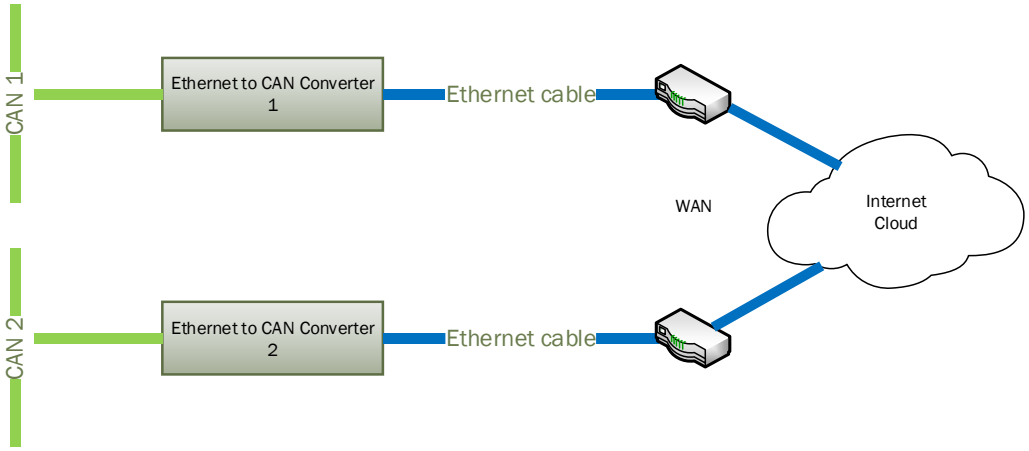


Figure 40. Global Internet Connection through the WAN

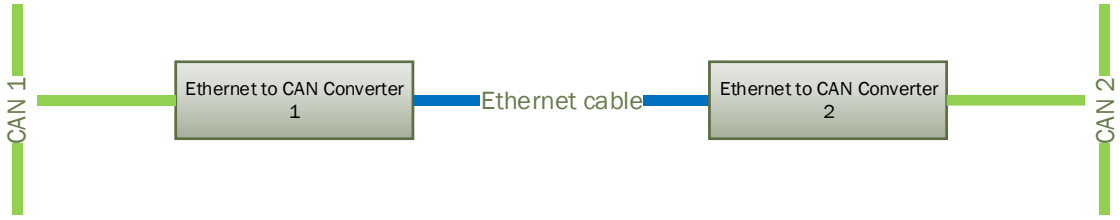


Figure 41. Simple Local Connection

The user can bridge more than two CAN networks on LAN or WAN.

7.1.1 Converter Configuration

After physical connection is established, the converters should be configured to exchange messages between each other. Since the converters support client/server communication model, one of the converters should be configured as a server and the other one – as a client.

7.1.1.1 Server Configuration

To configure the converter as a server, first set: *Device IP Address*, *Device Port*, *Device Subnet Mask* and *Device Default Gateway* to the appropriate values received from your network administrator. Configure the *Device Port Type* to *UDP* or *TCP* depending on the required message reliability and acceptable delays. Use unreliable but fast UDP when speed is a priority or reliable but slow TCP when message reliability is more important than the speed.

Set *Auto Connect to Remote* to *No* to disable the client side of the converter. The *Remote IP Address* and *Remote Port* can be left untouched since they are not used by the converter when the client mode is disabled. They are grayed on the *Settings* page in this mode.

For the CAN network, configure the necessary *Baud Rate* and keep default settings for configuration parameters defining CAN routing addresses (*Channel Group, RxD Channels, TxD Channel, Loopback Channel*). Make sure that the *Loopback Channel* is set to 0 to avoid sending the same messages back and forth between converters, which will lead to network saturation and communication failure.

An example of the converter configuration as a server is presented in Figure 42[1].

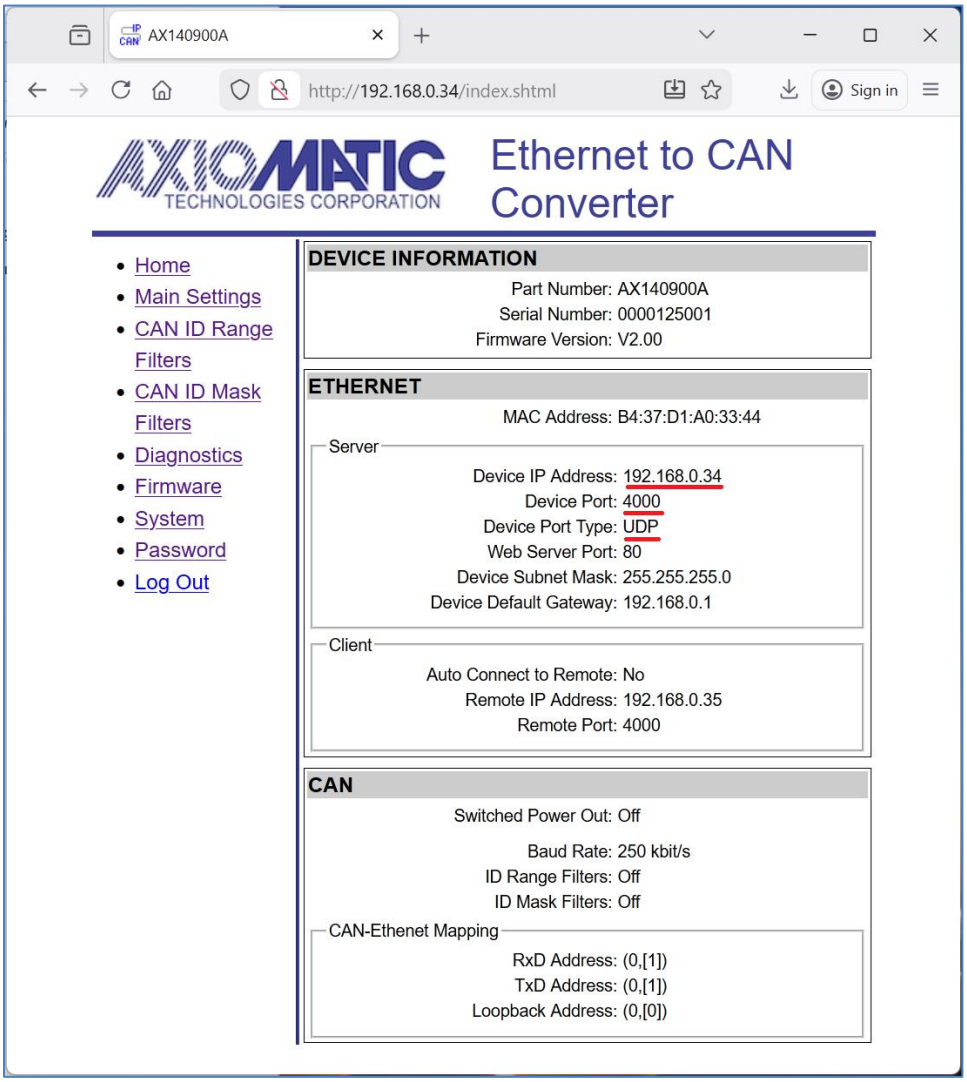


Figure 42. The Converter Server Configuration Example

[1]The Firmware Version number may be different from the firmware version described in the manual.

7.1.1.2 Client Configuration

In the client configuration, the user should set *Device IP Address*, *Device Port*, *Device Subnet Mask* and *Device Default Gateway* to the appropriate values received from the network administrator the same way as with the server configuration. After that, the user should set the *Device Port Type, Remote IP Address* and *Remote Port* to match the settings of the converter in the server mode and activate the client mode by setting the *Auto Connect to Remote* to *Yes*.

The CAN network setup is done similarly to the server mode; the *Baud Rate* is set to the desired baud rate (not necessarily the same as on the server) and the *Loopback Channel* is set to 0.

An example of the converter configuration in the client mode is presented in Figure 43.

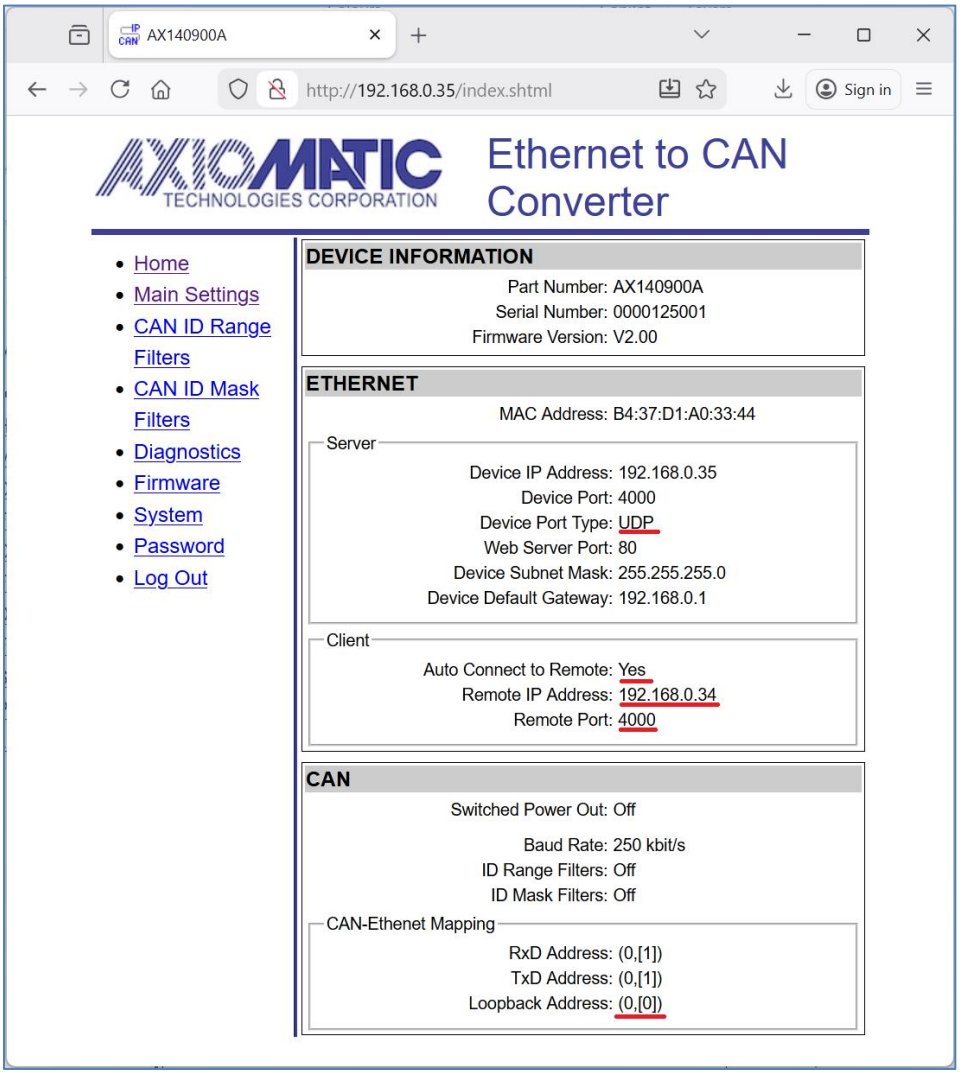


Figure 43. The Converter Client Configuration Example

The values of the *Device Port Type, Remote IP Address* and *Remote Port* of the client on Figure 43 are the same as the *Device Port Type*, *Device IP Address* and *Device Port* of the server in Figure 42.

Please note, that if the converters are connected over the internet, the *Remote IP Address* of the client will be a public IP address of the server, not the internal server IP address presented as the *Device IP Address* on Figure 42. A network administrator on the server side will be required to configure port forwarding to open the internet access to the converter in the server mode.

When the converter is configured as a client, it will still act as a server accepting connections on the *Device Port* from other clients. This adds versatility to the converter configurations since the same converter can be used together with both client and server communication nodes. As

an example, the user can establish an unlimited number of daisy-chain client-server connections, see: Figure 44.

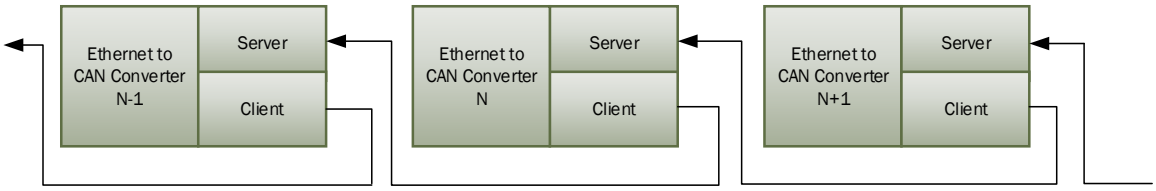


Figure 44. Daisy-Chain Converter Connection

8 CONVERTER DISCOVERY

In case the IP address and/or web server port is unknown or has been lost, the user can recover them using Axiomatic Windows console application `AxioDisc.exe`.

8.1 Axiomatic Discovery Application

The `AxioDisc.exe` application uses a proprietary discovery protocol and can recover IP locations of Axiomatic converters on the LAN. The application is available upon request.

The `AxioDisc.exe` application sends a UDP request to the broadcast IP address `255.255.255.255`, port `35100`, and waits for the responses from devices located on the same physical link of the LAN as the PC.

The converter discovery response includes the device *MAC Address*, *Device IP Address*, *Web Server Port*, *Device Port*, *Device Port Type*, the converter *Part Number* and *Serial Number*, see Figure 45.

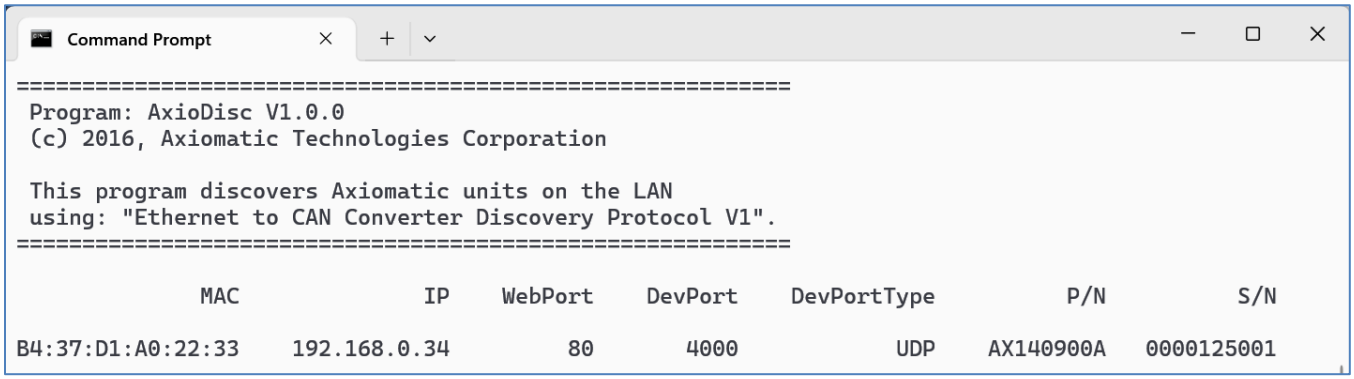


Figure 45. `AxioDisc.exe` Converter Discovery Application

In case the application does not see the device, the user should ensure that the IP network where the device is located is the only one that is enabled and active. All other network connections including Wi-Fi, VNP, virtual adapters, etc., should be temporarily disabled through the PC *Network Connections* settings.

The `AxioDisc.exe` application can run on Windows starting from Win XP SP3. It was tested on Win XP SP3, Win 7, Win 10 and Win 11. In case the application cannot find standard dlls, the Visual C++ Redistributable for Visual Studio 2015 x86 must be installed on the user's computer from the Microsoft website:
https://www.microsoft.com/en-ca/download/details.aspx?id=48145

The Axiomatic proprietary discovery protocol is supported by the CAN-ENET Software Support Package, P/N AX140910. The Software Support Package can be used by third party software developers to implement network discovery of the device.

9 TECHNICAL SPECIFICATIONS

9.1 Power Supply

9.1.1 Input

Power supply input is located on the Ethernet connector. The power supply uses automotive battery power. It is not compatible with the PoE (Power over Ethernet) IEEE 802.3 standard.

Table 15. Power Supply Input

Parameter	Value	Remarks
Supply Voltage	9…36 VDC	12V, 24V – nominal
Power Consumption	2W	Maximum at 12V
LED Indicator	Power	Red-green bicolor LED. Shared with Power supply output
Protection	Under Voltage Shutdown Over Voltage Shutdown Reverse Polarity Transients	< 6 V > 37V 12V Load Dump

9.1.2 Output

Power supply output is located on the CAN connector.

Table 16. Power Supply Output

Parameter	Value	Remarks
Voltage Output	9…36 VDC	Pass-through power from the power supply input.
Current Output	0.7A	Maximum pass-through current.
Voltage Drop	1V	Maximum
LED Indicator	Power	Red-green bicolor LED. Shared with Power supply input
Protection	Overcurrent Short to Battery Short to Ground	> 1A

9.1.3 Power LED Indicator

Table 17. Power LED

LED Color	State Description
Off	No power
Green	Power is on
Red	Power is on. Error on the power supply output

9.2 Ethernet

Table 18. Ethernet Parameters

Parameter	Value	Remarks
Number of Ports	1	
Port Type	10BASE-T, 100BASE-TX	IEEE 802.3i/u IEEE 802.3 compliant auto-negotiation
Speed	100 Mbps, 10 Mbps	Auto-configuration and full-duplex supported
MDIX	Auto-MDIX	Auto-crossover to eliminate cabling mismatch
LED Indicators	Speed, Link/Activity	Green LED
Protocols	Ethernet IEEE 802.3, IP, ICMP, ARP, UDP, TCP, HTTP, Proprietary[1]	CAN messages are transmitted using a proprietary application protocol running on top of the user selectable UDP or TCP transport protocol [1]. The

Parameter	Value	Remarks
		internal web server uses HTTP protocol. The unit supports a proprietary discovery protocol[2] [2].
Server Mode	Up to 10 bi-directional simultaneous connections	Up to 9 connections, if the Client mode is enabled
Client Mode	1 remote bi-directional connection	Auto-connect to a remote server, if connection is dropped or temporarily unavailable. Client mode can be disabled.
Web server	Provided	Hosts a website and a RESTful API for the converter configuration, diagnostics, and flashing application firmware. Password protected
RESTful API	Provided	Requires Basic HTTP authentication, as per RFC 7617
Internal Diagnostics	Health Status[1]	Internal health status of the converter is transmitted in heartbeat messages [1,3]. It is also available from the web server.

[1] Supported by *CAN-ENET Software Support Package* (SSP), P/N AX140910, v3.0.0+
[2] Axiomatic Windows console application `AxioDisc.exe` can be used for the converter discovery.

Reference documents describing proprietary protocols and *Health Status* field format are presented below in Table 20. The documents are available upon request.

Table 19. Reference Documents

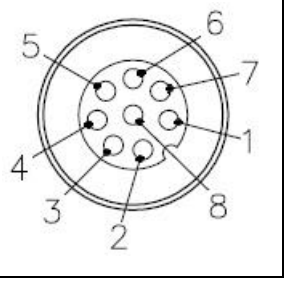
Reference Number	Document Name
[1]	O. Bogush, "Ethernet to CAN Converter Communication Protocol. Document version: 6", Axiomatic Technologies Corporation, April 22, 2025
[2]	O. Bogush, "Ethernet to CAN Converter Discovery Protocol. Document version: 1A", Axiomatic Technologies Corporation, April 5, 2021
[3]	O. Bogush, "Ethernet to CAN Converter Health Status. Document version: 4", Axiomatic Technologies Corporation, April 22, 2025
[4]	O. Bogush, “Ethernet to CAN Converter Password Reset Procedure. Document version: 1”, Axiomatic Technologies Corporation, December 1, 2025

9.2.1 Ethernet Connector

M12 socket, 8-pin, A-coded, female connector, Phoenix Contact, P/N: 1441817.

Table 20. Ethernet Connector Pinout

PIN #	Description
1	BAT + (9-36V)
2	BAT – (GND)
3	BAT – (GND)
4	ETH_TX –
5	ETH_RX +
6	ETH_TX +
7	BAT + (9-36V)
8	ETH_RX –



Use A-coded mating connectors compliant with IEC 61076-2-101:2012 or AX070531 accessory cable, see Table 28.

9.2.2 Ethernet LEDs

Table 21. Speed (10/100) LED

LED Color	State Description
Off	Link is up at 10 Mbps or no link
Green	Link is up at 100 Mbps

Table 22.Link/Activity LED

LED Color	State Description
Off	No link
Green	Link is up. No activity on the link
Blinking Green	Link is up and active

9.3 CAN

Table 23. CAN Parameters

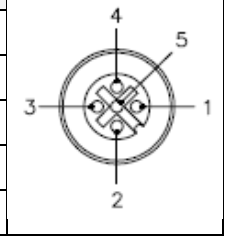
Parameter	Value	Remarks
Number of Ports	1	
Port Type	High Speed, ISO 11898-2 compatible	Twisted pair, up to 1 Mbit/s. Shield connection is provided if shielded cable is used. An external 120 Ohm termination resistor is required.
Baud Rate	1000, 666.6(6), 500, 250, 125, 100, 83.3(3), 50, 20, 10	[kbit/s]. Programmable through the web interface
Protocol	CAN Bosch 2.0A and B	Data frames and remote frames with standard and extended CAN ID are supported. CAN ID range and mask filtering is provided.

9.3.1 CAN Connector

M12 socket, 5-pin, A-coded, female connector, Phoenix Contact, P/N: 1441778.

Table 24. CAN Connector Pinout

PIN #	Description
1	CAN_SHIELD
2	POUT + (Switch Output)
3	POUT – (GND)
4	CAN_H
5	CAN_L



Use mating A-coded connectors compliant with IEC 61076-2-101:2012 or AX070532 accessory cable, see Table 28.

9.4 General Specifications

Table 25. General Specifications

Parameter	Value	Remarks
Operating Temperature	-40…+85 °C	Industrial temperature range
Storage Temperature	-40…+85 °C	
Environmental Protection	IP67	IEC 60529
Vibration	Sine sweep, 5-200 Hz, 8.9G peak, 2.5 hr (15 sweeps), each axis.	Custom profile

Parameter	Value	Remarks
	Random, 10-1014 Hz, 6.86 Grms, 5.0 hr, each axis.	
Shock	50G peak, 5 shocks, each axis	Custom profile
Size	4.19in x1.82in x1.32in (107mm x 47mm x 34 mm)	See dimensional drawing
Weight	0.15 lb (0.068 kg)	
Compliance	RoHS Directive	

Table 26. Electromagnetic Compatibility (EMC)

Standard	Description
ISO 13766-1:2018	Earth-moving and building construction machinery - Electromagnetic compatibility (EMC) of machines with internal electrical power supply - Part 1: General EMC requirements under typical electromagnetic environmental conditions
EN 61000-6-4:2005	Emission Standard for Industrial Environments
EN 61000-6-2:2007	Generic Standards – Immunity for Industrial Environments

9.5 Accessories

Table 27. Software Accessories

Axiomatic P/N	Description
AX140910	CAN-ENET Software Support Package (SSP), v3.0.0+
N/A	Axiomatic Windows console application `AxioDisc.exe`. Used for converter discovery on the LAN

Software accessories can be downloaded from www.axiomatic.com, log-in section.

Table 28. Hardware Accessories

Axiomatic P/N	Description
AX070531	Ethernet and Power Cable - 1.7m (5.5 ft.), 8-pin M12 A-coded, Unterminated Leads, Ethernet Jack. Temperature rating -40…+75 °C. Can be used for experimenting
AX070532	CAN Cable - 1.5 m (5 ft.), 5-pin M12 A-coded, Unterminated Leads. Temperature rating -40…+105 °C. Can be used for experimenting

9.6 Housing

Injection molded enclosure and cover. Material: PA66, 30% glass fiber reinforced, flame retardant UL 94 V-0. Ultrasonically welded. For dimensional drawing, see Figure 46.

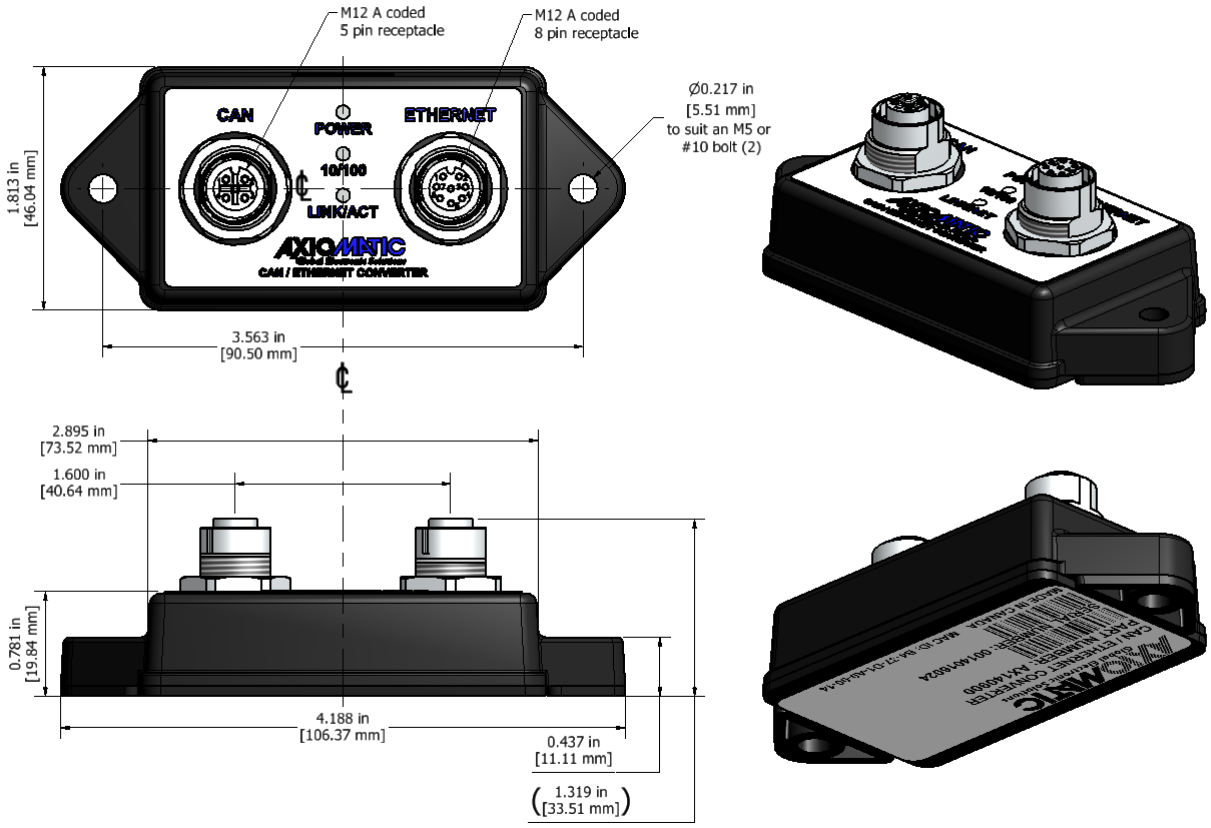


Figure 46. Dimensional Drawing

10 APPENDIX A. Third Party Software License Notices

This section contains Third Party Software License Notices and/or Additional Terms and Conditions for licensed third-party software components included in the Ethernet to CAN Converter firmware.

Table 29. Third Party Software License Notices

Third Party Software	License Notice/Terms
STM32F4xx CMSIS V2.6.11 / 25-April-2025	This software component is provided to you as part of a software package and applicable license terms are in the Package_license file. If you received this software component outside of a package or without applicable license terms, the terms of the Apache-2.0 license shall apply. You may obtain a copy of the Apache-2.0 at: https://opensource.org/licenses/Apache-2.0
STM32F4xx HAL Drivers V1.8.3 / 31-May-2024	Copyright (c) 2017 STMicroelectronics. All rights reserved. This software component is provided to you as part of a software package and applicable license terms are in the Package_license file. If you received this software component outside of a package or without applicable license terms, the terms of the BSD-3-Clause license shall apply. You may obtain a copy of the BSD-3-Clause at: https://opensource.org/licenses/BSD-3-Clause
STMicroelectronics microcontroller support software	COPYRIGHT(c) 2015 STMicroelectronics Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
FreeRTOS Kernel V11.1.0	Copyright (C) 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved. MIT License Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in

Third Party Software	License Notice/Terms
	the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
LwIP v2.1.2	Copyright (c) 2001-2004 Swedish Institute of Computer Science. All rights reserved. BSD license Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. The license is available at: http://lwip.wikia.com/wiki/License
Base64 Library in C	MIT License Copyright (c) 2016 Joe DF (joedf@ahkscript.org) Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in

Third Party Software	License Notice/Terms
	the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
coreJSON v3.3.0	Copyright (C) 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-Identifier: MIT Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Third party software version numbers are shown for application firmware 1.00. Higher version numbers can be used in subsequent releases.

11 VERSION HISTORY

User Manual Version	Firmware version	Date	Author	Modifications
2	2.xx	Feb 25, 2026	Olek Bogush	• Allowed colon symbol in passwords. Updated Password Update Web Page and Applying New Firmware subsections. • Removed redundant [Controller] configuration parameter group from configuration files. Updated *Configuration File Format* subsection. • Corrected *Command Resource* description in RESTful API. Updated *Commands* subsection in *RESTful Interface* section. • Updated Axiomatic logo in embedded website snapshots. • Updated Finnish office address on the front page.
1	1.xx	Dec 1, 2025	Olek Bogush	• Initial release

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 C
33880 Lempäälä
FINLAND
TEL: +358 10 337 5751
www.axiomatic.com
salesfinland@axiomatic.com