User Manual UMAX030520
Version 4
Firmware: V4.xx
EA: 4.9.74.0+

USER MANUAL

1 Analog Signal Output CAN Controller

P/N: AX030520

In Europe:
Axiomatic Technologies Oy
Höytämöntie 6
33880 Lempäälä - Finland
Tel. +358 3 3595 600
Fax. +358 3 3595 660
www.axiomatic.fi

In North America:
Axiomatic Technologies Corporation
5915 Wallace Street
Mississauga, ON Canada L4Z 1Z8
Tel. 1 905 602 9270
Fax. 1 905 602 9279
www.axiomatic.com

ACRONYMS

CAN	Controller Area Network
CANopen	CAN-based higher layer protocol supported by CAN in Automation (CiA)
DM	Diagnostic message. Defined in J1939/73 standard
EA	Electronic Assistant®. PC application software from Axiomatic, primary designed to view and program Axiomatic control configuration parameters (setpoints) through CAN bus using J1939 Memory Access Protocol
ECU	Electronic control unit
EMI	Electromagnetic Interference
LSB	Less Significant Byte
PC	Personal Computer
PGN	Parameter Group Number. Defined in J1939/73 standard
RS-232	PC serial port interface
SAE J1939	CAN-based higher level protocol designed and supported by Society of automobile Engineers (SAE)
USB	Universal Serial Bus
UTP	Un-shielded twisted pair

TABLE OF CONTENTS

1 INTRODUCTION

The following manual describes the controller software architecture, network functionality, setpoint and firmware programming of the 1 Analog Signal Output CAN Controller. It also contains technical specifications and installation instructions to help users build a custom solution on the base of this controller.

The user should check whether the application firmware installed in the converter is covered by this user manual. It can be done through CAN bus using Axiomatic Electronic Assistant® (EA) software. 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 converter application firmware V4.xx. Updates specific to the user manual are done by adding letters: A, B, …, Z to the user manual version number.

The controller supports SAE J1939 CAN interface. It is assumed, that the user is familiar with the J1939 group of standards; the terminology from these standards is widely used in this manual.

2 CONTROLLER DESCRIPTION

The controller is designed to monitor application signals transmitted on the CAN bus by various ECUs using one universal analog signal output. The output can be individually programmed to output voltage or current in the user-defined output range. The ECU application signals can be pre-processed before being output in case an advanced logic is required. Embedded voltage and current monitoring circuits are available to monitor the actual output signal.

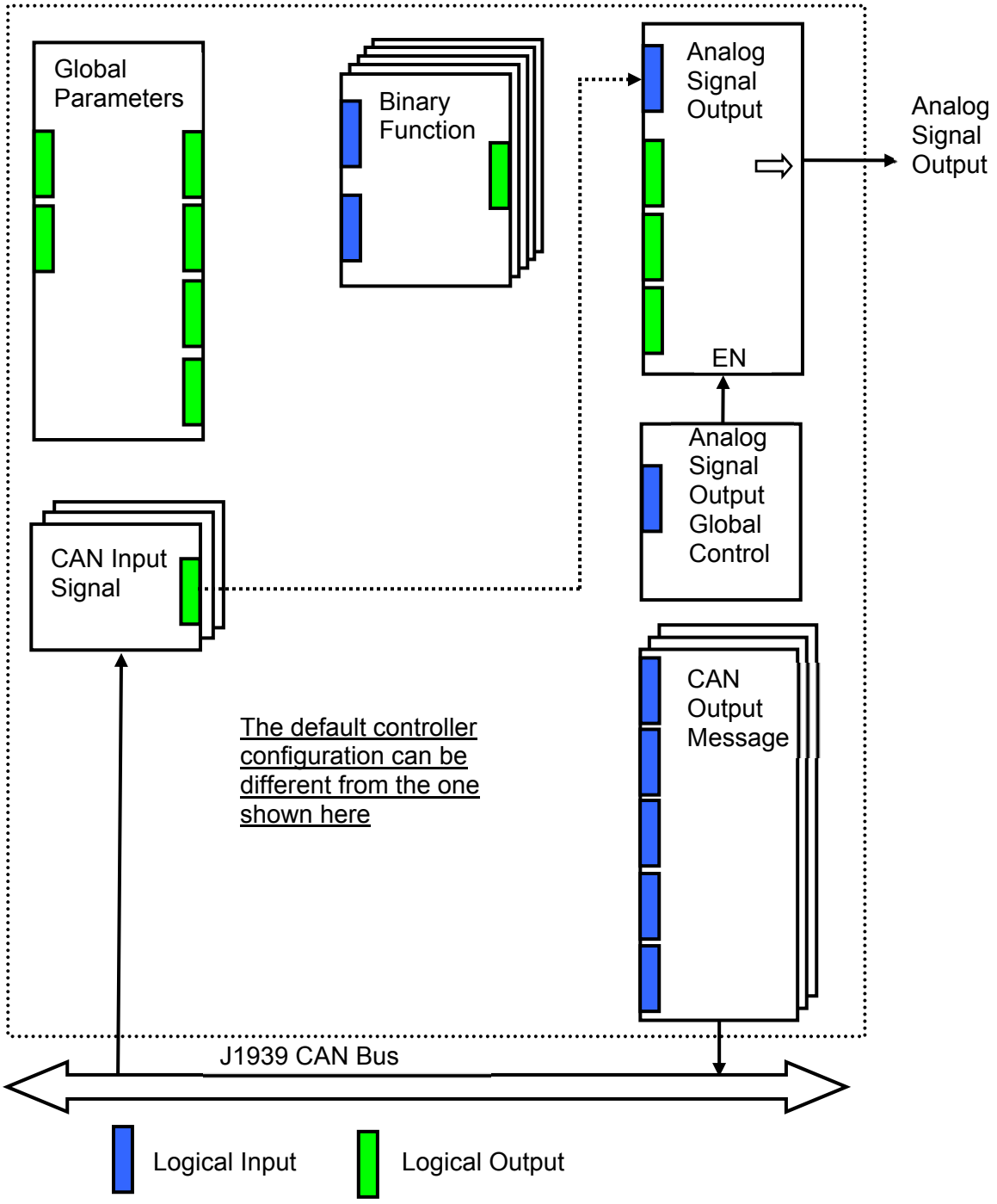
The 1 Analog Signal Output CAN Controller belongs to a family of Axiomatic user-customizable smart controllers. The programmable internal architecture provides users with an ultimate flexibility, allowing them to build their own custom controller with a required functionality from a set of predefined internal function blocks using PC-based Axiomatic EA software.

All application programming is performed through CAN interface, without disconnecting the controller from the user's system.

Besides reading application signals transmitted on the CAN bus, the controller can also transmit CAN application messages carrying signals internally generated by the controller. This feature can be used for monitoring the output signal and for debugging purposes.

3 CONTROLLER FUNCTION BLOCKS

From the software prospective, the controller consists of a set of internal function blocks, which can be individually programmed and arbitrarily connected together to achieve the required system functionality, see Figure 1.



As an example, the logical output of the CAN Input Signal function block is connected to the logical input of the Analog Signal Output function block, providing a direct path for the CAN input signal to the controller signal output.

Figure 1.The Controller Internal Structure.

Each function block is absolutely independent and has its own set of programmable parameters, or setpoints. The setpoints can be viewed and changed through CAN using Axiomatic Electronic Assistant® (EA) software.

There are two types of the controller function blocks. One type represents the controller hardware resources, for example the analog signal output block. The other type is purely logical – these function blocks are included to program the user defined functionality of the controller. The number and function diversity of these function blocks are only limited by the system resources of the internal microcontroller. They can be added or modified on the customer's request to accommodate user-specific requirements.

The user can build virtually any type of a custom control by logically connecting inputs and outputs of the function blocks. This approach gives the user an absolute freedom of customization and an ability to fully utilize the controller hardware resources in a user's application.

Depending on the block functionality, a function block can have: logical inputs, logical outputs or any combinations of them. The connection between logical inputs and outputs is defined by logical input setpoints. The following rules apply:

- A logical input can be connected to any logical output using a logical input setpoint.
- Two or more logical inputs can be connected to one logical output.
- Logical outputs do not have their own setpoints controlling their connectivity. They can only be chosen as signal sources by logical inputs.

To provide data flow between logical inputs and outputs, all logical output signals are normalized to [0;1] data range using the following equation:

$$Yn = (Y - Ymin) / (Ymax - Ymin),$$

where: Yn – normalized output value,
Y – original output value,
Ymax – maximum output value,
Ymin – minimum output value.

The original output values are restored, if necessary, at the logical inputs using the following reverse linear transformation:

$$X = Xn \bullet (Xmax - Xmin) + Xmin,$$

where: X – original restored input value,
Xn – normalized input value, Xn=Yn,
Xmax – maximum input value, Xmax=Ymax,
Xmin – minimum input value, Xmin=Ymin.

All function blocks have (Xmax, Xmin) and (Ymax, Ymin) setpoint pairs controlling the normalization process. They will be called "normalization parameters" further in the setpoint descriptions.

For discrete logical inputs and outputs the normalization parameters are not required, since the discrete signals can take only two values: {0,1}. When a regular logical output of a function block is

connected to a discrete logical input, it is assumed that the input values below 0.5 represent state 0 and above 0.5 – state 1:

Discrete Logical Input	Logical State
< 0.5	0
≥ 0.5	1

For additional flexibility, in a majority of function blocks, logical input signals can be inverted using the following inversion function:

Inv(Xn,I), I∈ {Yes,No},

Inv(Xn,I)={1-Xn, if I=Yes; Xn, if I=No}

In addition to signal values in the range of [0;1], the logical inputs and outputs also carry information on the state of the data source. This information can show that the source is not available or there is an error in data, or the data source is in a special state.

When the data source does not carry a valid data, the output signal value is always set to 0 and the inversion operation on the signal in suppressed. In this case, instead of the signal value, the logical signal carries a signal state code, associated with its signal state, see the table below:

Signal State	Signal Value, Xn	Signal State Code	Inverted Signal Value	
			Xn'=Inv(Xn,Yes)	Xn'=Inv(Xn,No)
Valid Data	[0;1]	0	1-Xn	Xn
Special	0	0…4294967295 (0…0xFFFFFFFF) – Special State Code	0	0
Error	0	0…4294967295 (0…0xFFFFFFFF) – Error Code	0	0
Not Available	0	0	0	0

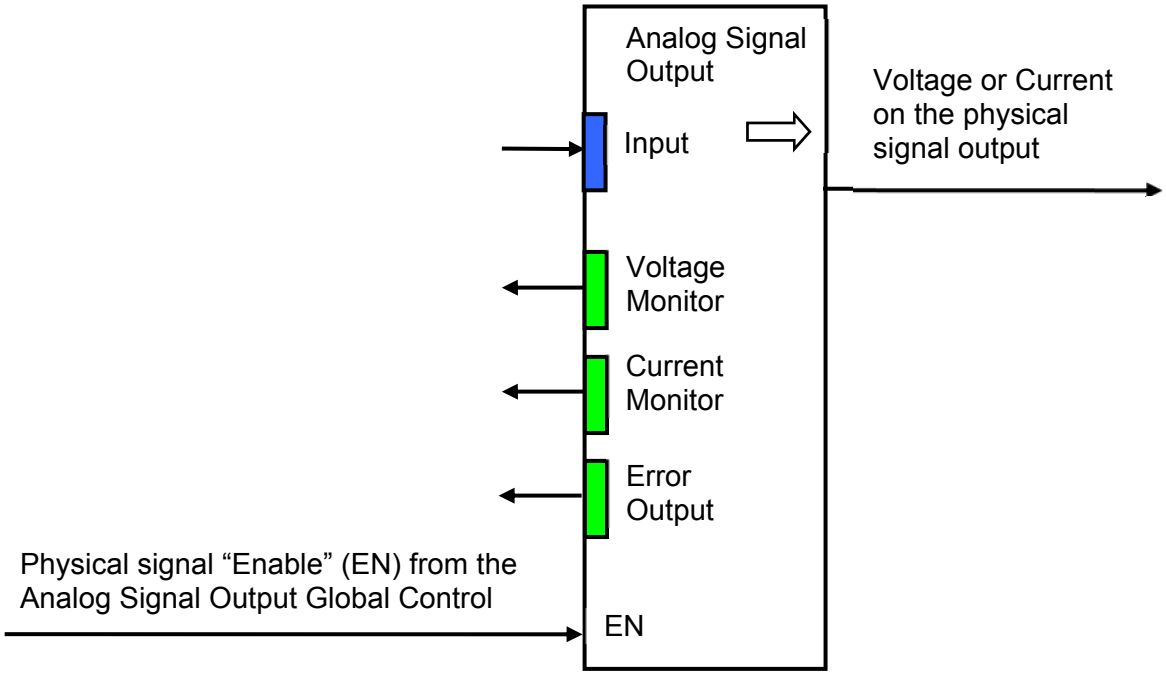
The states of the data source other than the "Valid Data" are primary used by CAN function blocks to report that a CAN input signal is absent on the bus, is out of range, etc. Other function blocks usually use only the "Error" state to show an error condition.

1.1 Analog Signal Output

There is one Analog Signal Output function block representing analog signal output of the controller. The function block can be programmed to output voltage or current in the user-defined range. The output signal can be enabled or disabled through the Analog Signal Output Global Control function block.

The controller has internal hardware circuits measuring voltage and current on the physical signal output. These parameters are available as logical signals on the voltage and current monitor logical outputs. In addition, the function block has a separate error logical output that flags an error condition when the output signal is beyond the specified error range.

The Analog Signal Output function block has one logical input receiving a normalized output signal and three logical outputs monitoring the state of the physical signal output. The block is also internally connected to the Analog Signal Output Global Control function block, which enables or disables the physical signal output.



The function block setpoints are presented in the following table:

Name	Default Value	Range	Units	Description
Input Source	CAN Input Signal #1	Any logical output of any function block or "Not Connected"	–	Defines an input signal source of the analog signal output.
Input Inversion	No	{Yes, No}	–	Defines whether the input signal from the Input Source is inverted.
Output Mode	Output Voltage	{Output Voltage, Output Current}	–	Specifies an output mode of the analog signal output.
Vmax – Maximum Output Voltage	5.0	[-10…10], but Vmax>Vmin	V	Normalization parameters for Output Voltage mode.
Vmin – Minimum Output Voltage	0	[-10…10], but Vmin<Vmax	V	
Imax – Maximum Output Current	20.0	[-20…20], but Imax>Imin	mA	Normalization parameters for Output Current mode.
Imin – Minimum Output Current	4.0	[-20…20], but Imin<Imax	mA	
Vmax – Voltage Monitor	12.0	–	V	Normalization Parameters for Voltage Monitor output. Read only parameters.
Vmin – Voltage Monitor	-12.0	–	V	
Imax – Current Monitor[1]	25.0	–	mA	

Name	Default Value	Range	Units	Description
Imin – Current Monitor[1]	-25.0	–	mA	Normalization Parameters for Current Monitor output. Read only parameters.
Output Error Threshold	5.0	[0…100]	%	Maximum relative error of the signal on the analog signal output. The Error Output is switched from “0” to “1” when the output error exceeds this value.
Output Error Delay	0.02	[0…300]	s	Delay before the Error Output changes its condition.
Disable Output On Error	No	{Yes, No}	–	If "Yes", Signal Output will be disabled on an error condition until the next power-up.
Disable Output On Error Delay	5.0	[0…300]	s	Delay before the Signal Output is disabled.

[1] Current Monitor logical output can be in an error state with an error code equal to 0 if voltage on the signal output is above or below the current monitor voltage range (±8 [V]).

The Output Error Threshold setpoint defines the output signal error relatively to the signal range specified by normalization parameters of the output signal. When the signal output is in the voltage output mode, these parameters are: Vmax – Maximum Output Voltage and Vmin – Minimum Output Voltage. In the current mode, they are: Imax – Maximum Output Current and Imin – Minimum Output Current.

The Error Output logical signal is set to “1” with the Output Error Delay time when the output signal error exceeds the Output Error Threshold value. For example, for the voltage output mode:

ErrorOutput = {1, if $|Vout'-Vmon|>\Delta$ for OutputErrorDelay time;
0, if $|Vout'-Vmon|\leq\Delta$ for OutputErrorDelay time },

Δ = (Vmax-Vmin)*OutputErrorThreshold/100.
Vout’ – Voltage expected on the output and defined by the logical input.
Vmon – Voltage measured on the output and available on the Voltage Monitor logical output.

The Output Error Delay time should allow the signal output and the monitor circuits to settle down after a new signal value is commanded to the output. This settling time is approximately 5…20 ms depending on the required monitoring accuracy. The application signal update rate and the slew rate should be also taken into consideration when defining this setpoint value.

When the Disable Output On Error setpoint is set to “Yes”, the signal output will be disabled (set to zero) and the ErrorOutput will be set to “1”until the next power-up, if the error condition on the output stays continuously for the time defined by the Disable Output On Error Delay setpoint. For the voltage output mode:

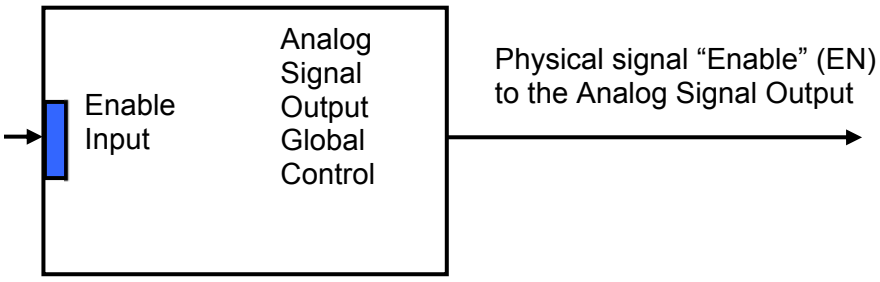
Vout’ = 0, ErrorOutput=1, if $|Vout'-Vmon|>\Delta$ for DisableOutputOnErrorDelay time.

For the current output, the error condition triggering the Error Output logical signal is either the output current error or the current monitor error. The current monitor error appears when the voltage on the signal output is too high for the current monitor to function properly. It is usually due to an open circuit on the output.

When the current monitor is in the error condition, the Current Monitor logical output is in the error state with an error code equal to 0. It is different from the Voltage Monitor output, which always shows the valid data.

1.2 Analog Signal Output Global Control

The Analog Signal Output Global Control function block is used to globally enable or disable the analog signal output of the controller. It has one logical input to control the analog signal output.



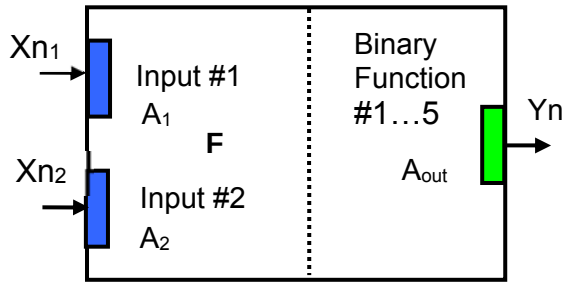
The function block setpoints are defined as follows:

Name	Default Value	Range	Units	Description
Enable Input Source	Constant Output = 1.0	Any logical output of any function block or “Not Connected”	–	Defines an input signal source to enable the signal output.
Enable Input Inversion (EA alias: Enable Input Source Inversion)	No	{Yes, No}	–	Defines whether the input signal from the Enable Input Source is inverted.

The Enable Input is connected to the Constant Output = 1.0 to enable the analog signal output by default.

1.3 Binary Function

There are five Binary Function blocks added to the controller to support advanced CAN signal monitoring algorithms. Each Binary Function block takes two logical input signals, scales them, and performs an arithmetic or logical operation on the scaled inputs. Then it outputs the result, which can be scaled as well.



The normalized output signal Yn of the Binary Function block can be presented by the following formula:

$$Yn=Clip(Y),$$

$$Y = A_{out} \bullet F[A_1 \bullet Xn_1, A_2 \bullet Xn_2],$$

where:

Clip(Y)={Y, if $0 \le Y \le 1$; 0, if $Y<0$; 1, if $Y>1$} – clipping function;
Xn_1, Xn_2 – normalized signal values of the input sources (can be inverted);
A_1, A_2 – input scale coefficients;
A_{out} – output scale coefficient;
F[x, y] – binary function of the scaled input signals: $x=A_1 \bullet Xn_1$, $y=A_2 \bullet Xn_2$.

In case one of the input sources is not connected, the output signal of the function block is not available and its signal value is equal to Yn=0.

The Binary Function block has the following set of setpoints:

Name	Default Value	Range	Units	Description
Input #1 Source	Not Connected	Any logical output of any function block or “Not Connected”	–	Source of the input #1 signal
Input #1 Inversion	No	{Yes, No}	–	Specifies, whether to invert the input #1 signal
Input #1 Scale	1.0	Any value	–	Input #1 signal scale coefficient
Input #2 Source	Not Connected	Any logical output of any function block or “Not Connected”	–	Source of the input #2 signal
Input #2 Inversion	No	{Yes, No}	–	Specifies, whether to invert the input #2 signal
Input #2 Scale	1.0	Any value	–	Input #2 signal scale coefficient
Function	+	{+, ∗,÷, Max, Min, OR, AND, XOR, <, ≤, =, >, ≥}	–	Binary function of the input #1 scaled signal and the input #2 scaled signal
Output Scale	1.0	Any value	–	Output signal scale coefficient

The binary functions F[x,y] have the following implementation specifics.

In the division function, to avoid ambiguity in dividing by 0, the dividend and the divisor are not allowed to be less than δ:

$$F^{(\div)}[x,y] = \max(x,\delta)/\max(y,\delta),$$

where: δ =1.0E-6 is a specially introduced computational constant.

For logical functions {OR, AND, XOR} values $X_i \geq 0.5$ (i=1,2) are treated as 1 (true) and $X_i < 0.5$ – as 0 (false).

To minimize influence of computational errors during normalization, comparison functions {$\leq$, =, $\geq$ } are defined the following way:

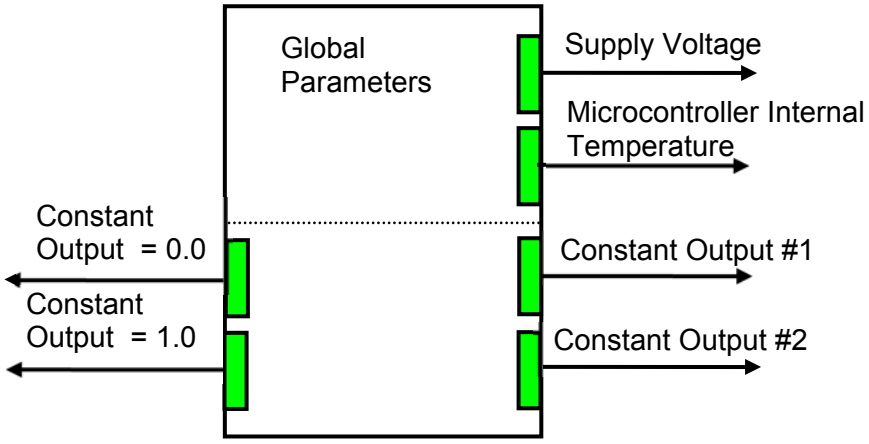
$$F^{(\leq)}[x,y] = \{1, \text{ if } x \leq y+\delta;\ 0, \text{ if } x > y+\delta\},$$

$$F^{(=)}[x,y] = \{1, \text{ if } |x-y| \leq \delta;\ 0, \text{ if } |x-y| > \delta\},$$

$$F^{(\geq)}[x,y] = \{1, \text{ if } x \geq y-\delta;\ 0, \text{ if } x < y-\delta\}.$$

1.4 Global Parameters

The Global Parameters function block gives the user access to the controller supply voltage and the microcontroller internal temperature as well as to a set of four constant logical outputs. These outputs can be used by other function blocks as constant input sources. For example, they can be used to set up threshold values in Binary Function blocks.



Two out of four constant logical outputs are user programmable. Other two represent logical one and logical zero outputs.

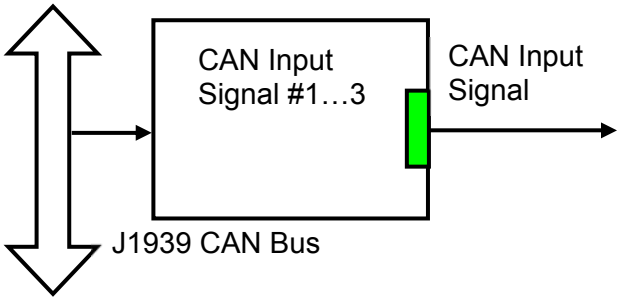
Please note, that the supply voltage, provided by the Global Parameters function block, is not the voltage on the controller power supply pins. It is an internal voltage measured after the reverse polarity protection and filtering circuit. It is always less than the actual power supply voltage by approximately 0.4…0.8 V.

The setpoints for the Global Parameters function block are presented in the following table:

Name	Default Value	Range	Units	Description
Constant Output #1	0.0	[0…1]	–	Logical output with a constant value.
Constant Output #2	0.0	[0…1]	–	Logical output with a constant value.
Vsmax – Max Supply Voltage	70	–	V	Normalization parameters for the inclinometer supply voltage. Read only parameters.
Vsmin – Min Supply Voltage	0	–	V	
Tmax – Max Microcontroller Temperature	150	–	°C	Normalization parameters for the microcontroller embedded temperature sensor. Read only parameters.
Tmin – Min Microcontroller Temperature	-50	–	°C	

1.5 CAN Input Signals

There are three CAN Input Signal function blocks supported by the controller. Each function block can be programmed to read single-frame CAN messages and extract CAN signal data presented in virtually any user-defined signal data format. The function block then outputs the signal data to its logical output for processing by other function blocks of the controller.



The CAN Input Signal function block has an ability to filter out signals transmitted only from a selected address. This way, it can be bound to a specific ECU on the CAN network. It can also automatically reset the input signal data in case the signal has been absent or lost for more than a specific period of time.

CAN application specific messages transmitted by the controller itself are also processed by this function block. The only difference in processing of the internal messages is that they are not sampled from the CAN bus and therefore their processing does not depend on a state of the CAN bus.

The setpoints of the CAN Input Signal function block are presented in the following table:

Name	Default Value	Range	Units	Description
Signal Type	Undefined	{Undefined, 1-Bit Discrete, 2-Bit Discrete, 4-Bit Discrete, 1-Byte Continuous, 2-Byte	–	Type of the CAN input signal

Name	Default Value	Range	Units	Description
		Continuous, 3-Byte Continuous, 4-Byte Continuous}		
PGN	65280	Any J1939 PGN value	–	PGN of the single frame CAN messages carrying the CAN input signal
PGN From Selected Address	No	{No, Yes}	–	Only CAN messages from the selected address will be accepted, if “Yes”
Selected Address	0	[0; 253]	–	Address of the ECU transmitting CAN messages carrying the CAN input signal
Data Position Byte	1	[1; 8]	–	Input signal data position byte within the CAN message data frame. LSB for continuous input signals
Data Position Bit	1	[1; 8]	–	Less significant input signal data position bit within the “Data Position Byte” for discrete input signals[1]
Resolution	1	Any value	Signal Units / Bit	CAN continuous signal resolution
Offset	0	Any value	Signal Units	CAN continuous signal offset
Signal Max Value	1	Any value, but: Signal Max Value > Signal Min Value	Signal Units	Normalization parameters for the CAN input signal. Valid only for continuous signals
Signal Min Value	0	Any value, but: Signal Min Value < Signal Max Value	Signal Units	
Autoreset Time	500	[0; 10000]	ms	Time interval, after which the output signal will be automatically reset to “Not Available”, if a new CAN message, carrying the signal, has not arrived. If 0 – autoreset is disabled.

[1]Discrete input signals should be within the “Data Position Byte” borders, not split between the adjacent bytes.

By default, the output of the first CAN Input Signal function block is connected to the input of the Analog Signal Output function block. It provides the simplest controller configuration with a direct control of the signal output by the CAN input signal. The second and third CAN Input Signal function blocks, not connected by default, can be engaged in more complicated CAN signal acquisition and processing algorithms involving Binary Function blocks and other controller resources.

According to the J1939/71 standard, CAN signals can carry not only signal values, but also special indicators, including: error indicator, “signal not available” indicator, etc. CAN signal types, supported by the controller, have the following CAN signal code mapping to the controller logical signals:

CAN Signal Type	CAN Signal Code	Logical Signal		
		State	Value	Signal State Code
1-Bit Discrete*	0…1	Valid Data	0…1 (=CANSignalCode)	0
2-Bit Discrete	0…1	Valid Data	0…1 (=CANSignalCode)	0
	2	Error	0	0
	3	Not Available	0	0
4-Bit Discrete*	0…1	Valid Data	0…1 (=CANSignalCode)	0
	2…13 (0x02…0x0D)	Special	0	0…11 =CANSignalCode-2
	14 (0x0E)	Error	0	0
	15 (0x0F)	Not Available	0	0
1-Byte Continuous	0…250 (0…0xFA)	Valid Data	[0;1] - normalized signal code	0
	251…253 (0xFB…0xFD)	Special	0	0…2 =CANSignalCode-251
	254 (0xFE)	Error	0	0
	255 (0xFF)	Not Available	0	0
2-Byte Continuous	0…64255 (0…0xFAFF)	Valid Data	[0;1] - normalized signal code	0
	64256…65023 (0xFB00…0xFDFF)	Special	0	0…767 =CANSignalCode-64256
	65024…65279 (0xFExx)	Error	0	0…255 =CANSignalCode-65024
	65280…65535 (0xFFxx)	Not Available	0	0
3-Byte Continuous	0…16449535 (0…0xFAFFFF)	Valid Data	[0;1] - normalized signal code	0
	16449536…16646143 (0xFB0000…0xFDFFFF)	Special	0	0…196607 =CANSignalCode-16449536
	16646144…16711679 (0xFExxxx)	Error	0	0…65535 =CANSignalCode-16646144
	16711680…16777215 (0xFFxxxx)	Not Available	0	0
4-Byte Continuous	0…4211081215 (0… 0xFAFFFFFF)	Valid Data	[0;1] - normalized signal code	0
	4211081216… 4261412863 (0xFB000000… 0xFDFFFFFF)	Special	0	0…50331647 =CANSignalCode-4211081216

CAN Signal Type	CAN Signal Code	Logical Signal		
		State	Value	Signal State Code
	4261412864… 4278190079 (0xFExxxxxx)	Error	0	0…16777215 =CANSignalCode-4261412864
	4278190080… 4294967295 (0xFFxxxxxx)	Not Available	0	0

*CAN signal code mapping for these types is specific to this control.

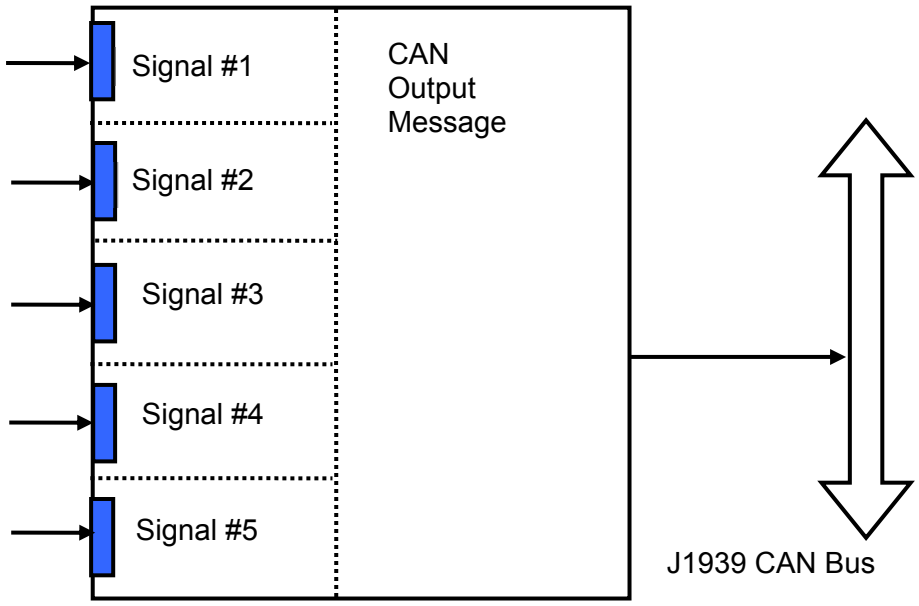
This mapping closely follows the J1939/71 standard for the 2-bit Discrete and all continuous CAN signal types, dividing the CAN code in similar ranges to represent different states of the signal. For the 1-bit and 4-bit Discrete signal types there are no generic rules specified by the J1939/71 standard to encode special indicators. The control uses its own mapping scheme for these types.

The J1939 standard does not specify how to encode the error codes and parameter specific indicators within the special indicator ranges. The control uses its own simple way of encoding, converting parameter specific and error indicators into absolute signal state codes. This allows to receive and transmit the same codes using different CAN signal types in a consistent way.

For example, if the logical signal is in the “Error” state with the error code equal to 1, the CAN signal code carrying this error will be 650251 (0xFE01) for the “2-Byte Continuous” CAN signal type or 4261412865 (0xFE00 0001) – for the “4-Byte Continuous” CAN signal type. See also the CAN Output Message function block for reverse conversion of the logical signals into the CAN signal codes.

1.6 CAN Output Messages

There are three CAN Output Message function blocks, which allow the controller to send three independent single frame application specific CAN messages to the CAN bus. The messages can be sent continuously or upon request. Each message contains up to five user defined CAN signals.



The CAN Output Message does not have a specific destination address. In case the message PGN is presented in the PDU1 format, the message is sent to the global address.

The setpoints of the CAN Output Message function block are presented in the following table:

Name	Default Value	Range	Units	Description
PGN	65281	Any J1939 PGN value	–	CAN output message PGN
Transmission Enable	No	{Yes, No}		Enables the CAN output message transmission
Transmission Rate	0	[0;10000]		CAN output message transmission rate. If 0 – transmission is upon request
Signal #1 Type	Undefined	{Undefined, 1-Bit Discrete, 2-Bit Discrete, 4-Bit Discrete, 1-Byte Continuous, 2-Byte Continuous, 3-Byte Continuous, 4-Byte Continuous}	–	Type of the CAN output signal #1
Signal #1 Source	Not Connected	Any logical output of any function block or “Not Connected”	–	Source of the CAN output signal #1
Signal #1 Inversion	No	{Yes, No}	–	Specifies, whether to invert the output signal #1
Signal #1 Data Position Byte	1	[1; 8]	–	Signal #1 data position byte within the CAN message data frame. LSB for continuous output signals
Signal #1 Data Position Bit	1	[1; 8]	–	Less significant signal #1 data position bit within the “Signal #1 Data Position Byte” for discrete output signals[1]
Signal #1 Resolution	1	Any value, except 0	Signal Units / Bit	CAN output signal #1 resolution. Valid only for continuous signals
Signal #1 Offset	0	Any value	Signal Units	CAN output signal #1 offset. Valid only for continuous signals
Signal #1 Max Value	1	Any value, but: Signal #1 Max Value > Signal #1 Min Value	Signal Units	Normalization parameters for the CAN output signal #1. Valid only for continuous signals
Signal #1 Min Value	0	Any value, but: Signal #1 Min Value < Signal #1 Max Value	Signal Units	
Signal #2 Type	Undefined	{Undefined, 1-Bit Discrete, 2-Bit Discrete, 4-Bit Discrete, 1-Byte Continuous, 2-Byte Continuous, 3-Byte	–	Type of the CAN output signal #2

Name	Default Value	Range	Units	Description
		Continuous, 4-Byte Continuous}		
Signal #2 Source	Not Connected	Any logical output of any function block or “Not Connected”	–	Source of the CAN output signal #2
Signal #2 Inversion	No	{Yes, No}	–	Specifies, whether to invert the output signal #2
Signal #2 Data Position Byte	1	[1; 8]	–	Signal #2 data position byte within the CAN message data frame. LSB for continuous output signals
Signal #2 Data Position Bit	1	[1; 8]	–	Less significant signal #2 data position bit within the “Signal #2 Data Position Byte” for discrete output signals[1]
Signal #2 Resolution	1	Any value, except 0	Signal Units / Bit	CAN output signal #2 resolution. Valid only for continuous signals
Signal #2 Offset	0	Any value	Signal Units	CAN output signal #2 offset. Valid only for continuous signals
Signal #2 Max Value	1	Any value, but: Signal #2 Max Value > Signal #2 Min Value	Signal Units	Normalization parameters for the CAN output signal #2. Valid only for continuous signals
Signal #2 Min Value	0	Any value, but: Signal #2 Min Value < Signal #2 Max Value	Signal Units	
Signal #3 Type	Undefined	{Undefined, 1-Bit Discrete, 2-Bit Discrete, 4-Bit Discrete, 1-Byte Continuous, 2-Byte Continuous, 3-Byte Continuous, 4-Byte Continuous}	–	Type of the CAN output signal #3
Signal #3 Source	Not Connected	Any logical output of any function block or “Not Connected”	–	Source of the CAN output signal #3
Signal #3 Inversion	No	{Yes, No}	–	Specifies, whether to invert the output signal #3
Signal #3 Data Position Byte	1	[1; 8]	–	Signal #3 data position byte within the CAN message data frame. LSB for continuous output signals
Signal #3 Data Position Bit	1	[1; 8]	–	Less significant signal #3 data position bit within the “Signal #3 Data Position Byte” for discrete output signals[1]

Name	Default Value	Range	Units	Description
Signal #3 Resolution	1	Any value, except 0	Signal Units / Bit	CAN output signal #3 resolution. Valid only for continuous signals
Signal #3 Offset	0	Any value	Signal Units	CAN output signal #3 offset. Valid only for continuous signals
Signal #3 Max Value	1	Any value, but: Signal #3 Max Value > Signal #3 Min Value	Signal Units	Normalization parameters for the CAN output signal #3. Valid only for continuous signals
Signal #3 Min Value	0	Any value, but: Signal #3 Min Value < Signal #3 Max Value	Signal Units	
Signal #4 Type	Undefined	{Undefined, 1-Bit Discrete, 2-Bit Discrete, 4-Bit Discrete, 1-Byte Continuous, 2-Byte Continuous, 3-Byte Continuous, 4-Byte Continuous}	–	Type of the CAN output signal #4
Signal #4 Source	Not Connected	Any logical output of any function block or “Not Connected”	–	Source of the CAN output signal #4
Signal #4 Inversion	No	{Yes, No}	–	Specifies, whether to invert the output signal #4
Signal #4 Data Position Byte	1	[1; 8]	–	Signal #4 data position byte within the CAN message data frame. LSB for continuous output signals
Signal #4 Data Position Bit	1	[1; 8]	–	Less significant signal #4 data position bit within the Signal #4 Data Position Byte for discrete output signals[1]
Signal #4 Resolution	1	Any value, except 0	Signal Units / Bit	CAN output signal #4 resolution. Valid only for continuous signals
Signal #4 Offset	0	Any value	Signal Units	CAN output signal #4 offset. Valid only for continuous signals
Signal #4 Max Value	1	Any value, but: Signal #4 Max Value > Signal #4 Min Value	Signal Units	Normalization parameter for the CAN output signal #4. Valid only for continuous signals
Signal #4 Min Value	0	Any value, but: Signal #4 Min Value < Signal #4 Max Value	Signal Units	
Signal #5 Type	Undefined	{Undefined, 1-Bit Discrete, 2-Bit Discrete, 4-Bit Discrete, 1-Byte Continuous, 2-Byte Continuous, 3-Byte	–	Type of the CAN output signal #5

Name	Default Value	Range	Units	Description
		Continuous, 4-Byte Continuous}		
Signal #5 Source	Not Connected	Any logical output of any function block or “Not Connected”	–	Source of the CAN output signal #5
Signal #5 Inversion	No	{Yes, No}	–	Specifies, whether to invert the output signal #5
Signal #5 Data Position Byte	1	[1; 8]	–	Signal #5 data position byte within the CAN message data frame. LSB for continuous output signals
Signal #5 Data Position Bit	1	[1; 8]	–	Less significant signal #5 data position bit within the “Signal #5 Data Position Byte” for discrete output signals[1]
Signal #5 Resolution	1	Any value, except 0	Signal Units / Bit	CAN output signal #5 resolution. Valid only for continuous signals
Signal #5 Offset	0	Any value	Signal Units	CAN output signal #5 offset. Valid only for continuous signals
Signal #5 Max Value	1	Any value, but: Signal #5 Max Value > Signal #5 Min Value	Signal Units	Normalization parameter for the CAN output signal #5. Valid only for continuous signals.
Signal #5 Min Value	0	Any value, but: Signal #5 Min Value < Signal #5 Max Value	Signal Units	

[1]CAN discrete signals should be within the “Data Position Byte” borders, not split between the adjacent bytes.

The logical signals can carry not only signal values but also error and special codes reflecting different states of the logical signal. The logical signals are converted into CAN signal codes the same way as in the CAN Input Signal function block, closely following the J1939/71 standard when possible. See the table below:

CAN Signal Type	Logical Signal			CAN Signal Code
	State	Value	Signal State Code	
1-Bit Discrete	Valid Data	[0;1]	0	0, if Value<0.5 1, if Value≥0.5
	Special*	0	0…4294967295 (0…0xFFFFFFFF)	1
	Error*	0	0…4294967295 (0…0xFFFFFFFF)	1
	Not Available*	0	0	1
2-Bit Discrete	Valid Data	[0;1]	0	0, if Value<0.5 1, if Value≥0.5
	Special*	0	0…4294967295 (0…0xFFFFFFFF)	3 (Same as “Not Available”)
	Error	0	0…4294967295	2

CAN Signal Type	Logical Signal			CAN Signal Code
	State	Value	Signal State Code	
			(0…0xFFFFFFFF)	
	Not Available	0	0	3
4-Bit Discrete*	Valid Data	[0;1]	0	0, if Value<0.5 1, if Value≥0.5
	Special	0	0…4294967295 (0…0xFFFFFFFF)	2…13 (0x02…0x0D) =SignalStateCode+2, if SignalStateCode<12 =13, if SignalStateCode ≥12
	Error	0	0…4294967295 (0…0xFFFFFFFF)	14 (0x0E)
	Not Available	0	0	15 (0x0F)
1-Byte Continuous	Valid Data	[0;1]	0	0…250 (0…0xFA) – calculated from the Value using normalization parameters
	Special	0	0…4294967295 (0…0xFFFFFFFF)	251…253 (0xFB…0xFD) = SignalStateCode+251, if SignalStateCode<3, =253, if SignalStateCode ≥3
	Error	0	0…4294967295 (0…0xFFFFFFFF)	254 (0xFE)
	Not Available	0	0	255 (0xFF)
2-Byte Continuous	Valid Data	[0;1]	0	0…64255 (0…0xFAFF) – calculated from the Value using normalization parameters
	Special	0	0…4294967295 (0…0xFFFFFFFF)	64256…65023 (0xFB00…0xFDFF) = SignalStateCode+64256, if SignalStateCode<768, =65023, if SignalStateCode ≥768
	Error	0	0…4294967295 (0…0xFFFFFFFF)	65024…65279 (0xFExx) = SignalStateCode+65024, if SignalStateCode<256, =65279, if SignalStateCode ≥256
	Not Available	0	0	65535 (0xFFFF)
3-Byte Continuous	Valid Data	[0;1]	0	0…16449535 (0…0xFAFFFF) – calculated from the Value using normalization parameters
	Special	0	0…4294967295 (0…0xFFFFFFFF)	16449536…16646143 (0xFB0000…0xFDFFFF) = SignalStateCode+16449536, if SignalStateCode<196608, =16646143, if SignalStateCode ≥196608
	Error	0	0…4294967295 (0…0xFFFFFFFF)	16646144…16711679 (0xFExx) = SignalStateCode+16646144, if SignalStateCode<65536, =16711679, if SignalStateCode ≥65536
	Not Available	0	0	16777215 (0xFFFFFF)

CAN Signal Type	Logical Signal			CAN Signal Code
	State	Value	Signal State Code	
4-Byte Continuous	Valid Data	[0;1]	0	0…4211081215 (0… 0xFAFFFFFF) – calculated from the Value using normalization parameters
	Special	0	0…4294967295 (0…0xFFFFFFFF)	4211081216… 4261412863 (0xFB000000… 0xFDFFFFFF) =SignalStateCode+4211081216, if SignalStateCode<50331648, =4261412863, if SignalStateCode ≥50331648
	Error	0	0…4294967295 (0…0xFFFFFFFF)	4261412864… 4278190079 (0xFExxxxxx) =SignalStateCode+4261412864, if SignalStateCode<16777216, =4278190079, if SignalStateCode ≥16777216
	Not Available	0	0	4294967295 (0xFFFFFFFF)

*Conversion rules are specific to this control. They are not defined by the J1939/71 standard.

4 NETWORK SUPPORT

The controller is designed to work on the J1939 CAN network. When connected to the network or upon power up, it automatically recognizes the network connection, claims a network address, and then starts a network communication.

The network part of the controller is compliant with Bosch CAN protocol specification, Rev.2.0, Part B, and the following J1939 standards:

ISO/OSI Network Model Layer	J1939 Standard
Physical	J1939/11 – Physical Layer, 250K bit/s, Twisted Shielded Pair. Rev. SEP 2006. J1939/15 - Reduced Physical Layer, 250K bits/sec, Un-Shielded Twisted Pair (UTP). Rev. AUG 2008.
Data Link	J1939/21 – Data Link Layer. Rev. DEC 2006
	The controller supports Transport Protocol for: Commanded Address messages (PGN 65240), ECU identification messages -ECUID (PGN 64965), and software identification messages -SOFT (PGN 65242). It also supports responses on PGN Requests (PGN 59904).
Network	J1939, Appendix B – Address and Identity Assignments. Rev. FEB 2010. J1939/81 – Network Management. Rev. 2003-05.
	The controller is an Arbitrary Address Capable ECU. It can dynamically change its network address in real time to resolve an address conflict with other ECUs. The controller supports: Address Claimed Messages (PGN 60928), Requests for Address Claimed Messages (PGN 59904) and Commanded Address Messages (PGN 65240).
Transport	N/A in J1939.
Session	N/A in J1939.
Presentation	N/A in J1939.
Application	J1939/71 – Vehicle Application Layer. Rev. SEP 2013
	The controller can receive application specific PGNs with input signals and transmit application specific PGNs with up to five output signals. All application specific PGNs are user programmable.
	J1939/73 – Application Layer – Diagnostics. Rev. FEB 2010
	Memory access protocol (MAP) support: DM14, DM15, DM16 messages used by EA to program configuration parameters.

1.7 J1939 Name and Address

Upon connecting to the network, before sending and receiving any application data, the controller claims its network address with the unique J1939 Name. The Name fields are presented in the table bellow:

Field Name	Field Length	Field Value	User Programmable
Arbitrary Address Capable	1 bit	1 (Capable)	No

Field Name	Field Length	Field Value	User Programmable
Industry Group	3 bit	0 (Global)	No
Vehicle System Instance	4 bit	0 (First Instance)	No
Vehicle System	7 bit	0 (Nonspecific System)	No
Reserved	1 bit	0	No
Function	8 bit	66 (I/O Controller)	No
Function Instance	5 bit	19 (AX030520, 1 Analog Signal Output CAN Controller, Axiomatic proprietary)	No
ECU Instance	3 bit	0 (First Instance)	Yes
Manufacturer Code	11 bit	162 (Axiomatic Technologies Corp.)	No
Identity Number	21 bit	Calculated on the base of the microcontroller unique ID	No

The user can change the controller *ECU Instance* using EA to accommodate multiple controllers on the same CAN network.

The controller takes its network *ECU Address* from a pool of addresses assigned to self-configurable ECUs. The address is pre-set to 154, but the controller can change it during an arbitration process or upon receiving a commanded address message. The new address value is then stored in a non-volatile memory and is used during the next address claim procedure. The user can also change the controller *ECU Address* using EA, if necessary.

1.8 Slew Rate Control

To adjust the controller to the parameters of the CAN physical network, the controller has a setpoint controlling the CAN transceiver slew rate. It can be set to “Fast” or “Slow” slew rate according to the following table:

Setpoint Value	Slew Rate
Fast	19 V/µs
Slow	4 V/ µs

For the majority of J1939 CAN applications the slow slew rate is preferable due to the reduced EMI of the transceiver.

1.9 Network Bus Terminating Resistors

An absence of the CAN bus terminating resistors is the most common source of the CAN bus communication errors.

The controller does not have an embedded 120 Ohm CAN bus terminating resistor. The appropriate resistors should be installed externally on both ends of the CAN twisted pair cable according to the J1939/11 or J1939/15 standards.

Even if the length of the CAN network is short and the signal reflection from both ends of the cable can be ignored, at least one 120 Ohm resistor is required for the majority of CAN transceivers to operate properly.

1.10 Network Configuration Parameters

The following table summarizes the EA configuration parameters controlling the controller CAN network functionality:

Name	Default Value	Range	Units	Description
ECU Instance Number	0	[0…7]	–	ECU Instance field of the J1939 ECU Name.
ECU Address	154	[0…253]		ECU network address
Slew Rate	Slow	{Slow, Fast}	–	Slew rate control of the CAN transceiver

5 CONFIGURATION PARAMETERS

The converter configuration parameters can be viewed and changed using the standard J1939 memory access protocol through the CAN bus using Axiomatic PC-based Electronic Assistant® (EA) software.

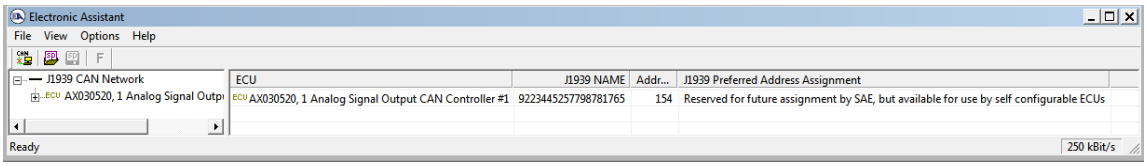
1.11 Electronic Assistant Software

Axiomatic provides PC-based Electronic Assistant® (EA) software, together with a USB-CAN converter, as a kit P/N AX070502, to communicate with a wide range of Axiomatic products, including this converter. Please also refer to the EA user manual UMAX07050X for the description of the EA and for the network connection troubleshooting.

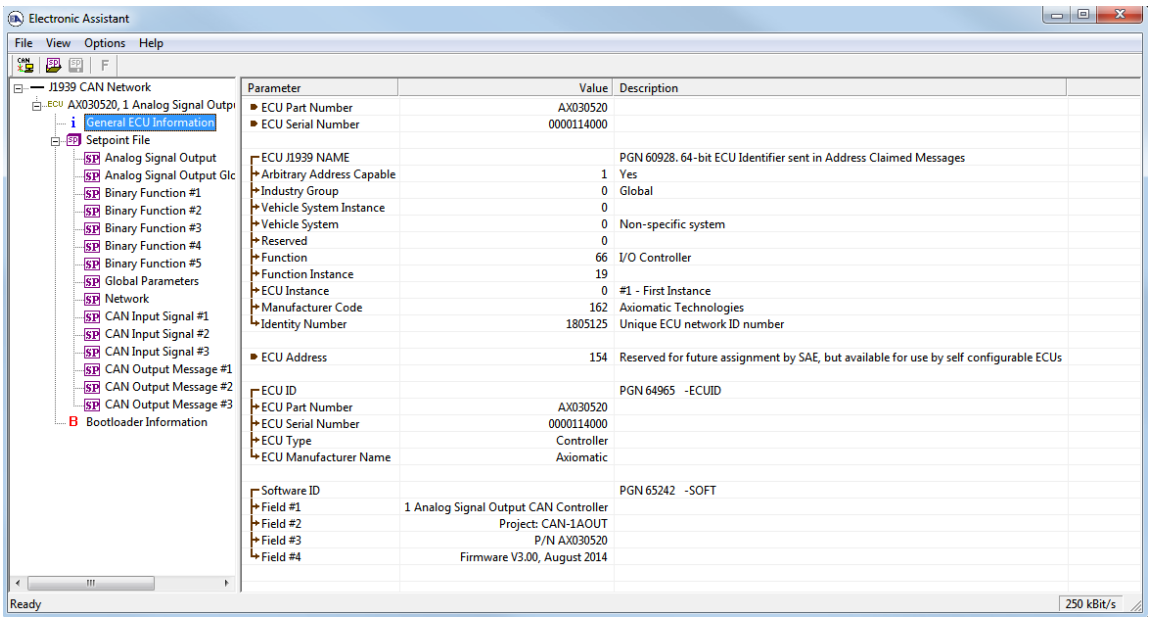
The user should use EA software version 4.9.74.0 or higher, which supports this converter firmware. The most recent EA software version can be downloaded from www.axiomatic.com web site.

Before connecting to the converter, the user should first check whether the baud rate in EA is set to the default 250kBit/s (displayed in the bottom-right corner of the EA screen).

Upon connection, the EA will show the converter on the list of controls that are present on the J1939 CAN network. If there is only one converter on the network, the following screen will appear:



The user can then browse through the ECU parameters, read *General ECU Information* and *Bootloader Information* groups, view and modify configuration parameters:

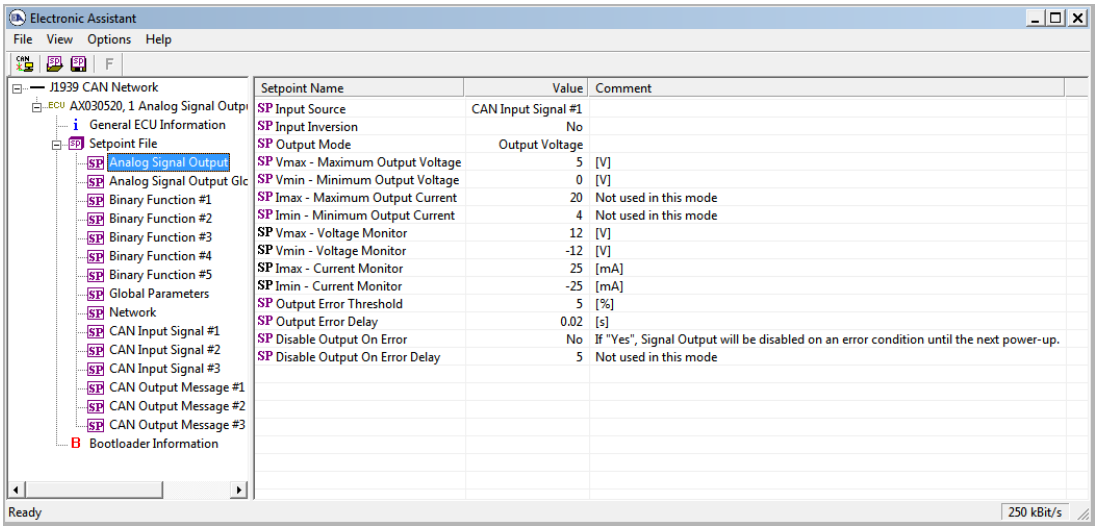


The configuration parameters are grouped by function blocks. Please, refer to the appropriate section of this manual describing the required function block.

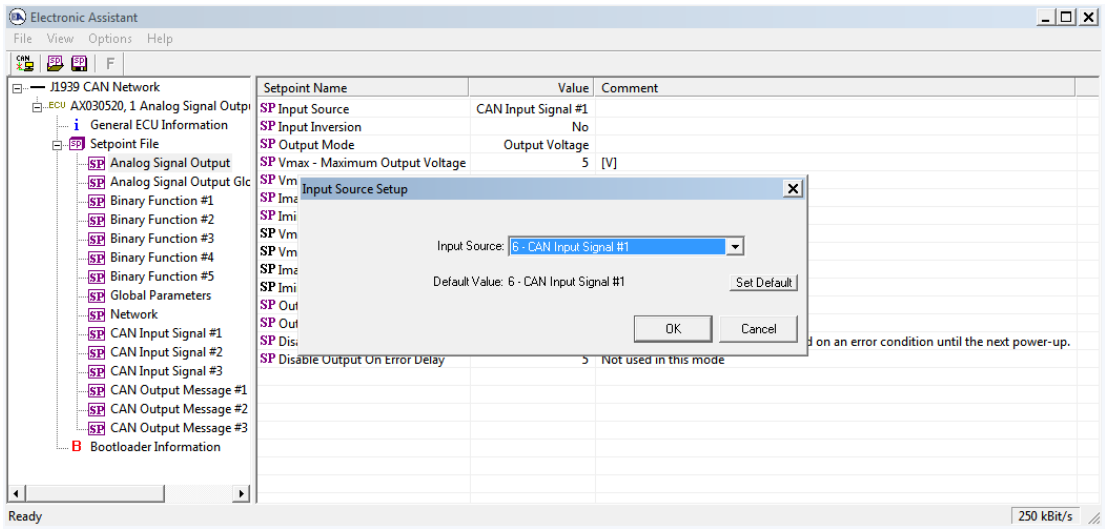
In the *General ECU Information* group, the user will see the version number of the application firmware. Please, make sure that the user manual version number matches with the most significant part of the application firmware version number. Otherwise, a different user manual is required to work with this converter.

1.12 Function blocks in EA

Each converter function block is presented by its own setpoint group in the *Setpoint File* main group. Individual configuration parameters (setpoints) of a function block can be accessed through the function block setpoint group:



The user can view and, when necessary, change configuration parameters by double-clicking on the appropriate setpoint name. A pop-up dialog window will appear:



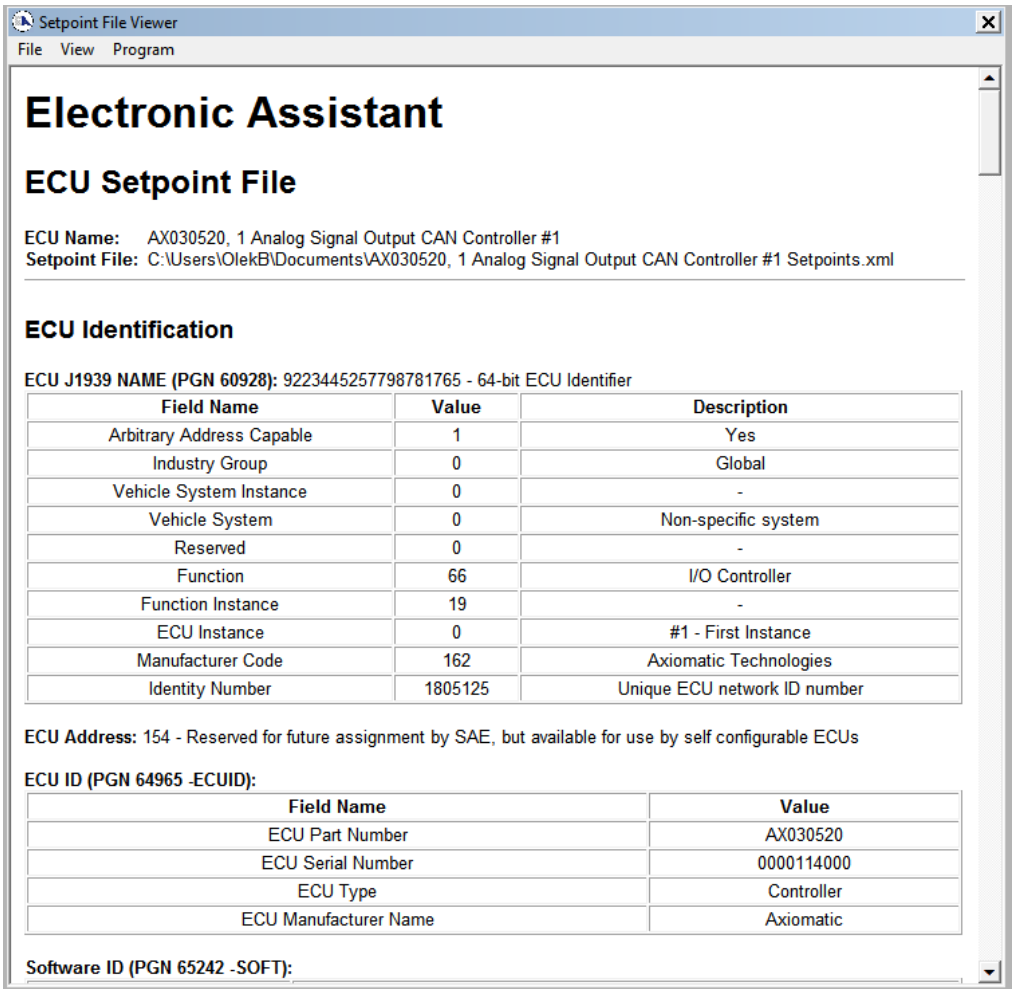
If the user changes the configuration parameter, the new value will be stored in a non-volatile memory and used immediately by the converter.

The converter will perform an internal reset of all function blocks after each change of the configuration parameters. If the new configuration parameter affects the CAN network identification, the converter will reclaim its network address with a new network identification message.

1.13 Setpoint File

The EA can store all converter configuration parameters in one setpoint file and then flash them into the converter in one operation.

The setpoint file is created and stored on disk using a command *Save Setpoint File* from the EA menu or toolbar. The user then can open the setpoint file, view or print it, and also flash configuration parameters from the setpoint file into the converter.



The CAN network identification and “read-only” configuration parameters are not transferrable using this operation. Also, the converter will perform one or several internal resets of all function blocks during the setpoint flashing operation.

There can be small differences in configuration parameters between different versions of the application firmware. It is recommended that the user manually inspect all configuration parameters after flashing if the setpoint file was created by a different version of the application firmware.

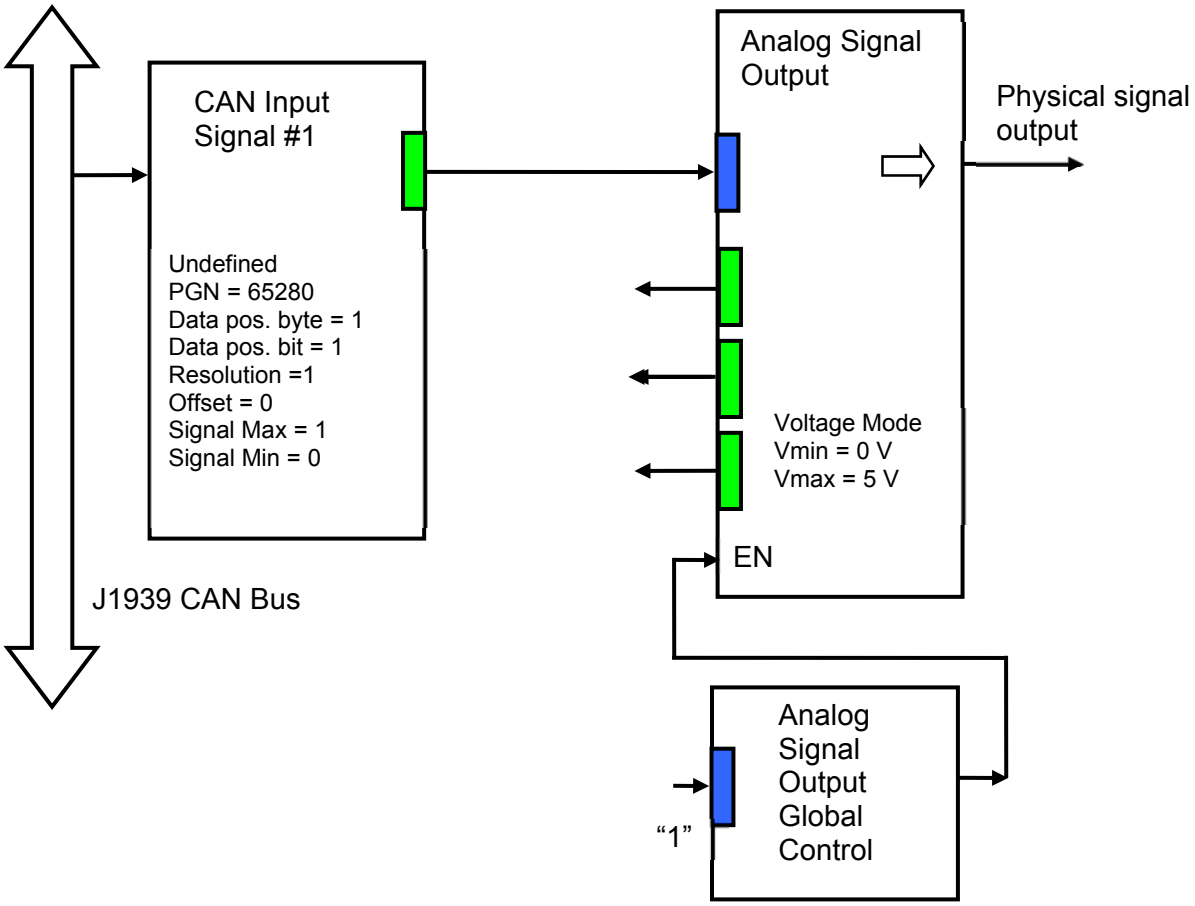
A setpoint file containing default configuration parameters is available upon request.

1.14 Default Configuration Parameters

The controller is preprogrammed by the manufacturer with default values of configuration parameters. These values can be found for each internal function block in the Controller Function Blocks section of this manual.

In the default controller configuration, the signal output is enabled and set to the voltage output mode with 0…5V voltage range. It is also connected to the CAN Input Signal #1 function block (Figure 2). The CAN Input Signal #1 function block is disabled through the *Signal Type* setpoint, which is set to the “Undefined” value.

This configuration does not provide any useful system functionality. It is intended to be used by users only as a template to build a user-specific system configuration.



Unused function blocks and Global Parameters function block are not shown.
The Analog Signal Output is at 0V independently of the CAN input signals. Uses can use this configuration as a template to build their own system configurations.

Figure 2. The Block Diagram of the Default Controller Configuration.

1.15 Controller Configuration Example

The controller should be configured to perform the required system functionality before being used in the user system. A detailed description of the controller configuration process is presented below, as an example.

1.15.1 User Requirements

Let the controller be used to output a displacement angle of a platform relatively to ground as a voltage with 100mV/Deg resolution. We will assume that the platform is equipment with an inclinometer sensor transmitting pitch and roll angles over a CAN network.

1.15.2 Programming Steps

First, create a block diagram of the required controller configuration using the controller function blocks (Figure 1). Then, configure the controller Analog Signal Output function block.

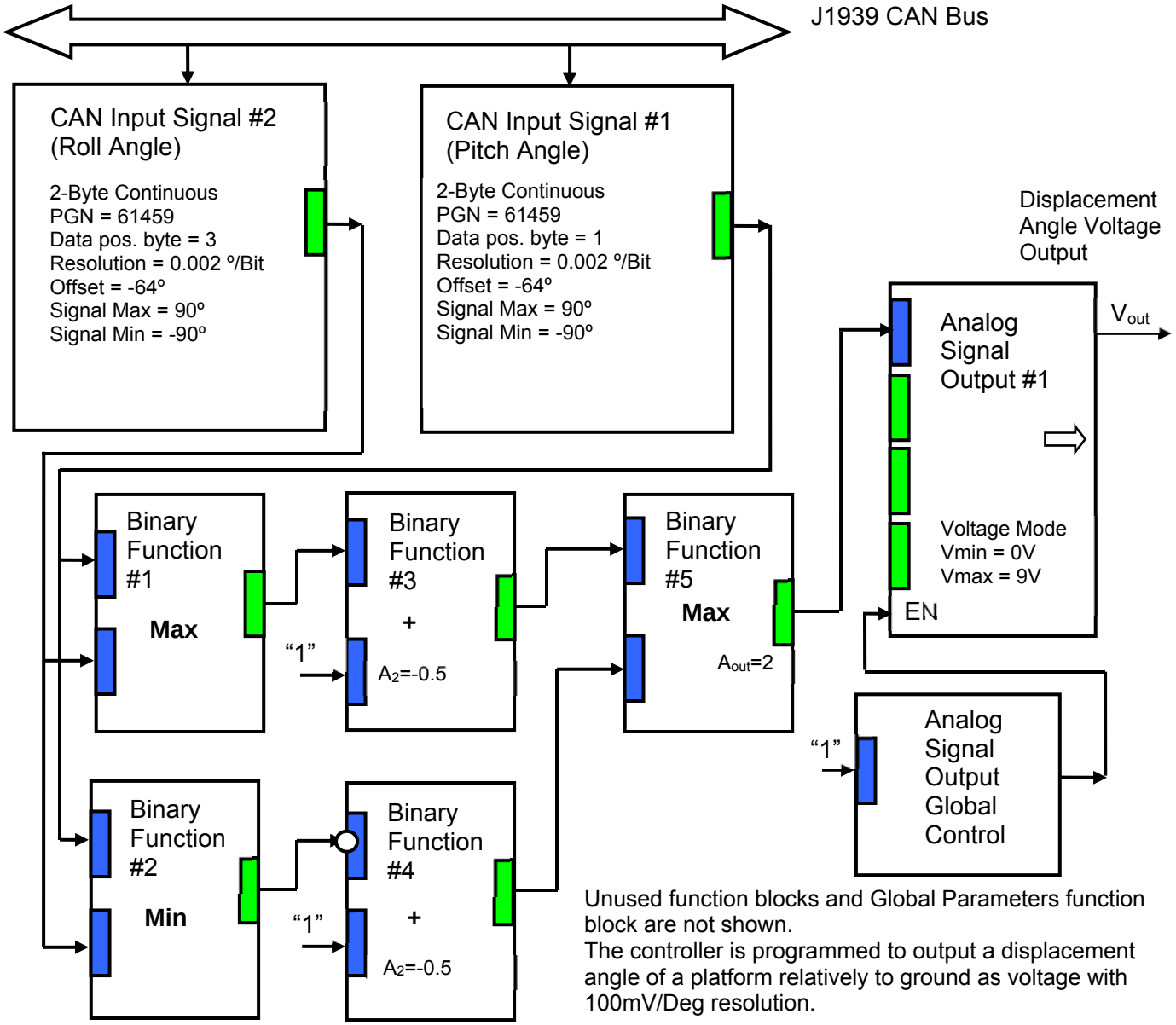
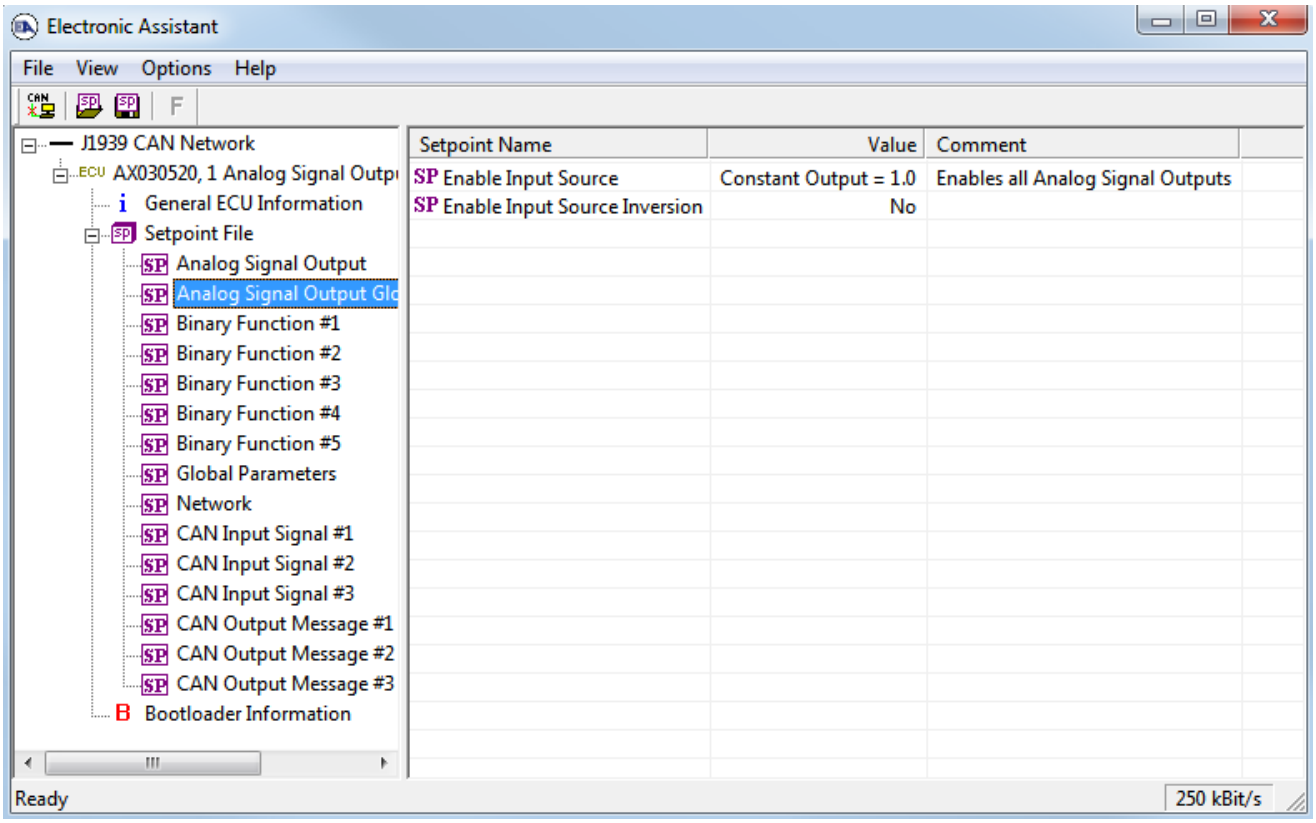


Figure 3. The Block Diagram of the Controller Configuration for the Controller Configuring Example.

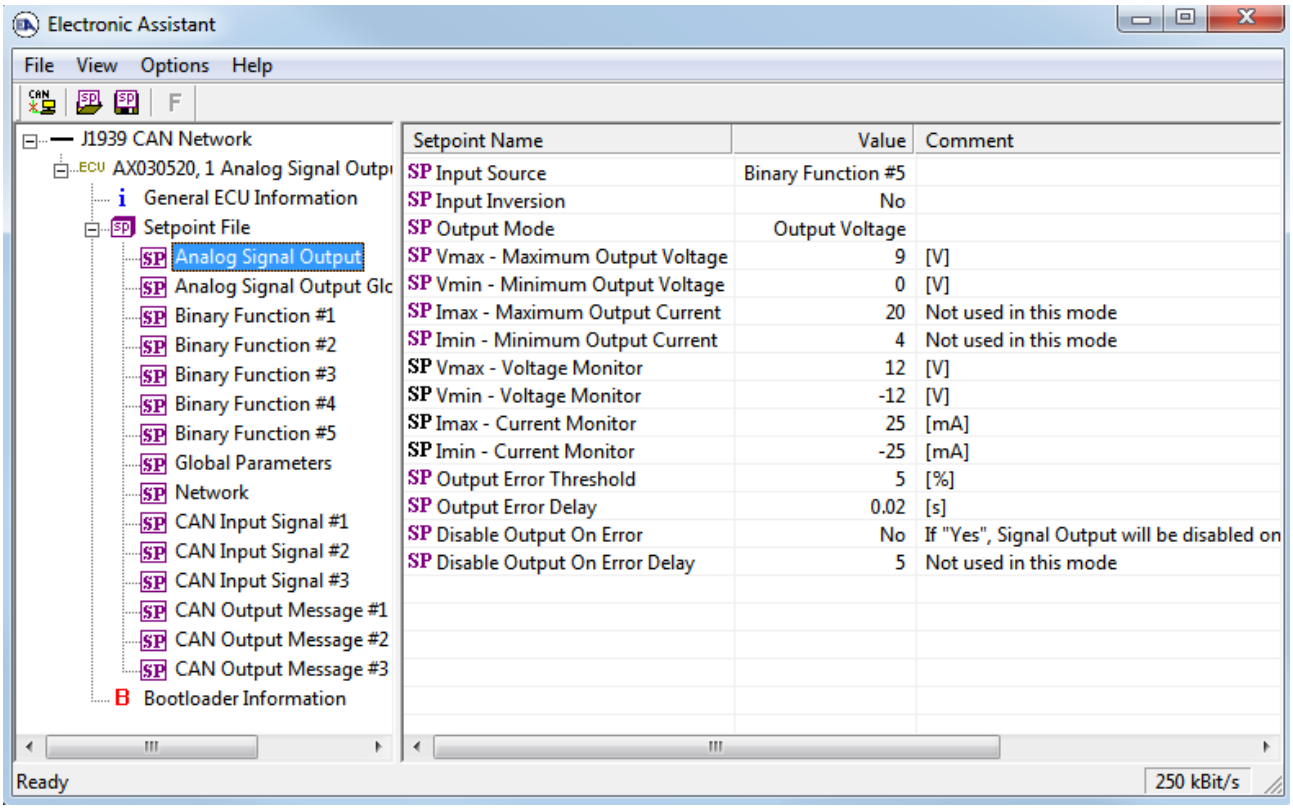
Start configuration with enabling the analog signal output through the Analog Signal Output Global Control function block by setting the *Enable Input Source* configuration parameter to "1" using the Constant Output = 1.0 logical output from the Global Parameters function block.



Then, configure the Analog Signal Output function block. Set the *Output Mode* setpoint to “Output Voltage”, the *Vmin – Minimum Output Voltage* to 0V, and the *Vmax – Maximum Output Voltage* to +9V:

100 [mV/Deg] * 90 [Deg] = 9000 [mV] = 9 [V].

Also, connect the logical input of this function block to the logical output of the Binary Function #5 function block using the *Input Source* configuration parameter. Finally, ensure that you do not invert the logical input data and the *Input Inversion* is set to “No”.



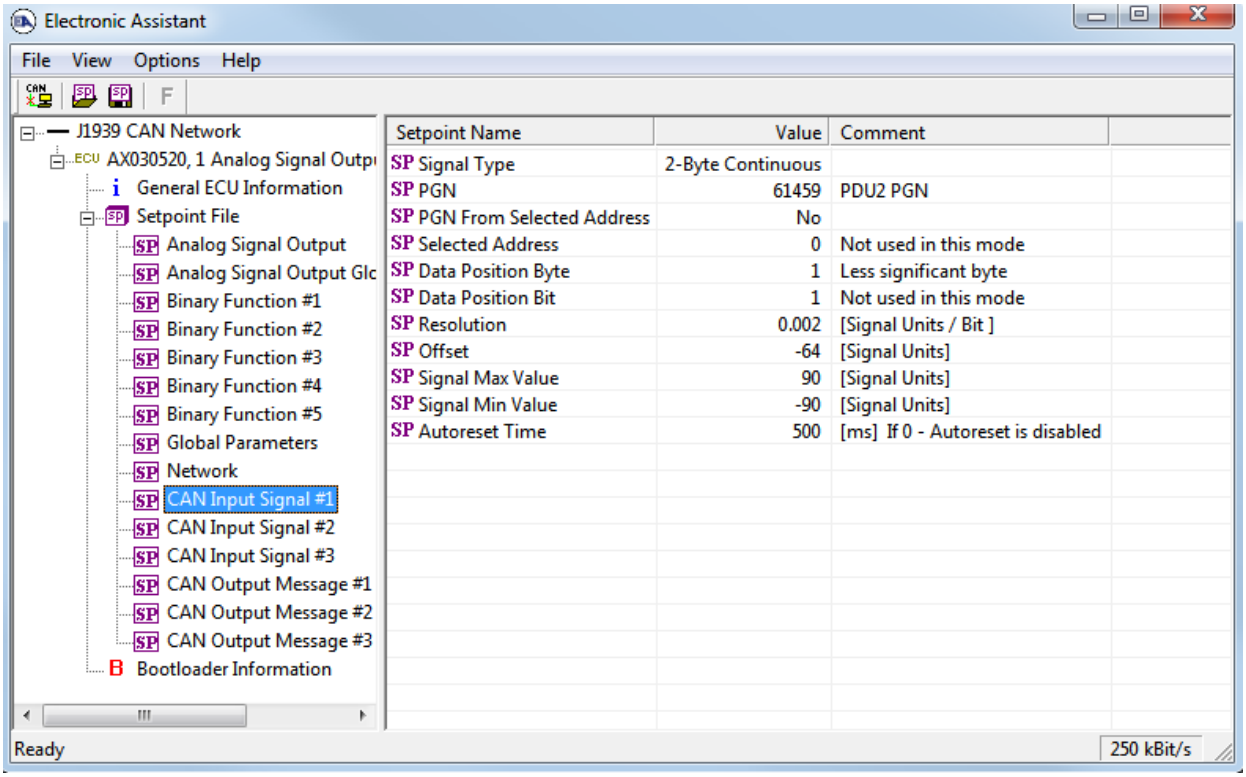
Now, configure the CAN Input Signal #1 and CAN Input Signal #2 function blocks to accept pitch and roll signals from the CAN bus. According to the J1939/71 standard, these signals are transmitted in PGN 61459 (Slope Sensor Information) as SPN 3318 (Pitch Angle) and SPN 3319 (Roll Angle), see the following table:

	SPN 3318 Pitch Angle	**SPN 3319 Roll Angle**
Description	The angle between the vehicle x-axis and the ground plane.	The angle between the vehicle y-axis and the ground plane.
Data Length	2 bytes	2 bytes
Start Position in the PGN data frame	1	3
Resolution	0.002 deg/bit, -64 offset	0.002 deg/bit, -64 offset
Data Range	-64 to 64.51 deg	-64 to 64.51 deg
Operational Range	same as data range	same as data range

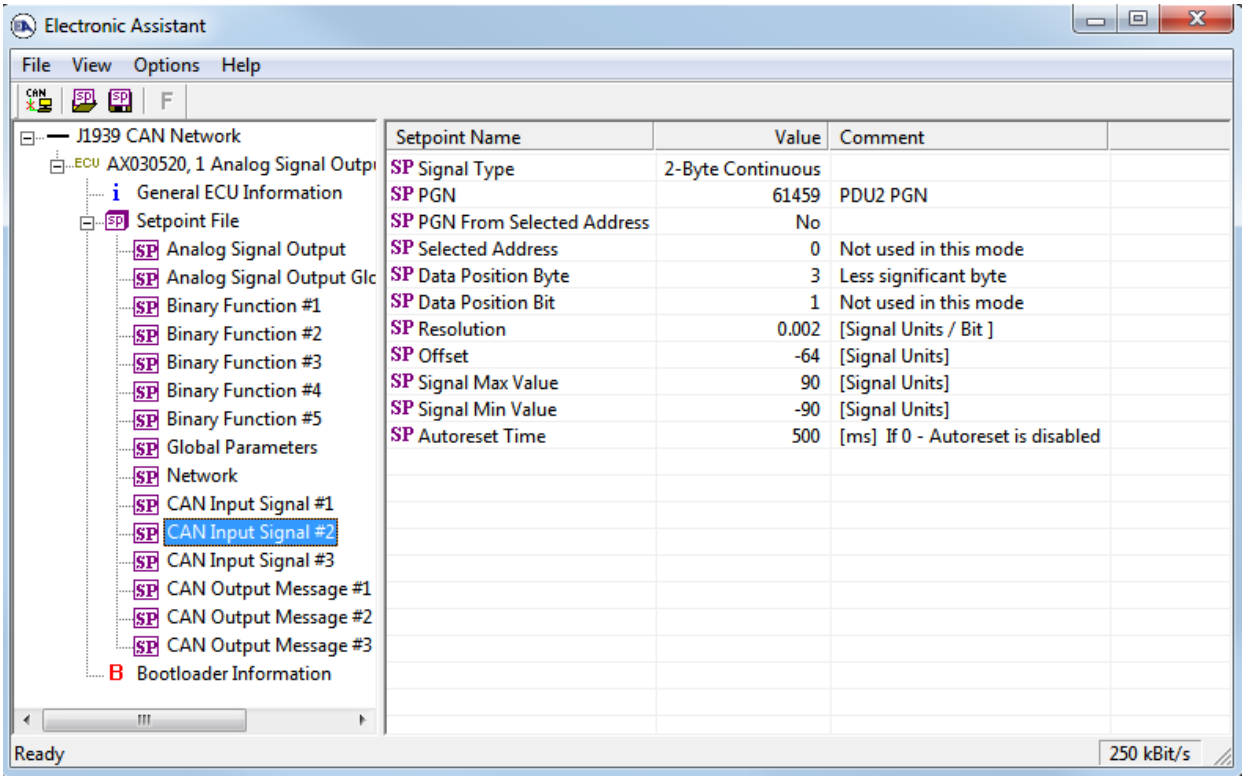
First, in the CAN Input Signal #1 function block, set *Signal Type* to “2-Byte Continuous” to meet the data length parameter of the pitch angle signal. Then, set the *PGN* to 61459 (Slope Sensor Information), the *Data Position Byte* – to 1, the *Resolution* – to 0.002 [Signal Units/Bit] and the *Offset* – to -64 [Signal Units].

Keep the default value “No” for the *PGN From Selected Address* configuration parameter unless you have several slope sensors on the network and the PGN filtering is required. Also, keep *Autoreset Time* at the default value of the 500ms to reset the logical output signal of the function block when the CAN signal is absent.

Finally, set normalization setpoints *Signal Max Value* and *Signal Min Value* to +90º and -90º, to cover the entire data range of the pitch angle signal:

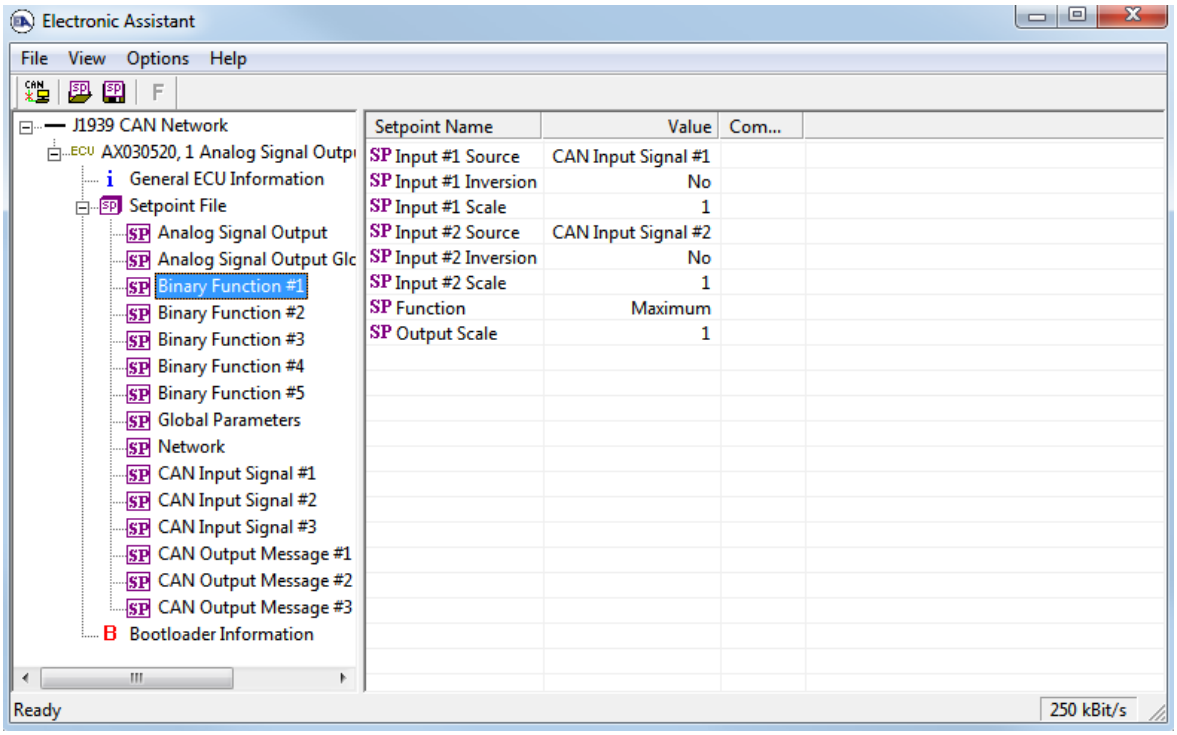


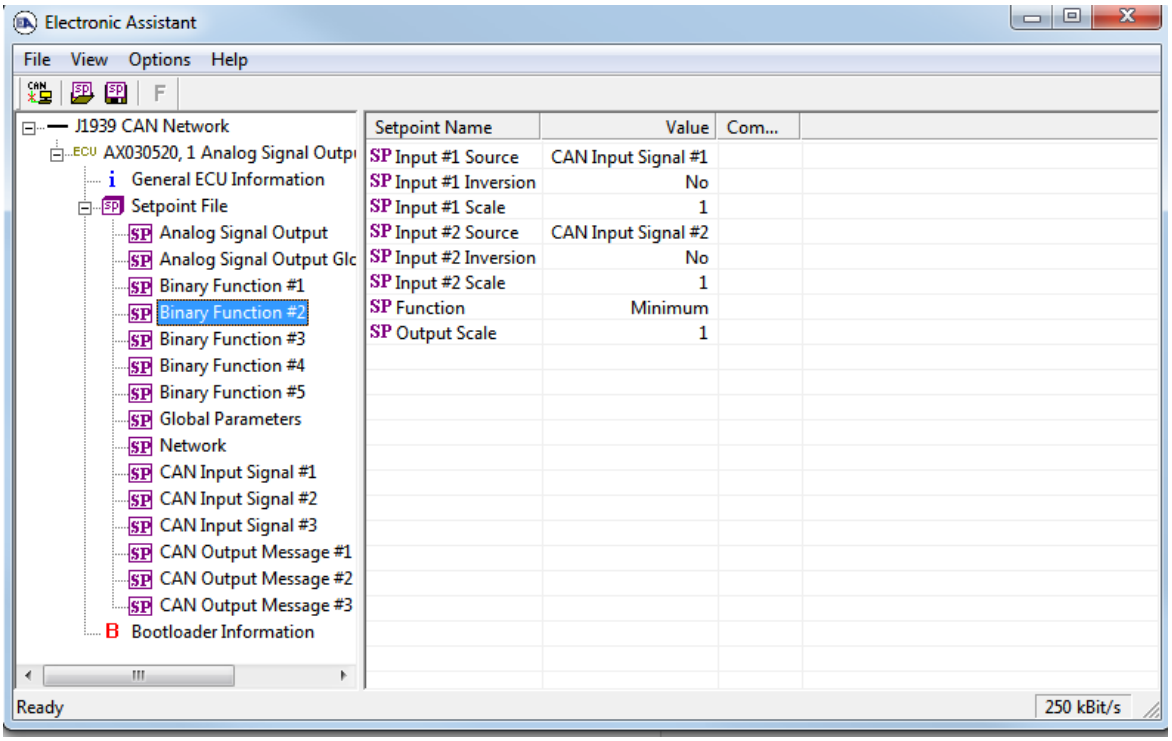
Similarly configure the CAN Input Signal #2 function block for receiving the roll angle signal:



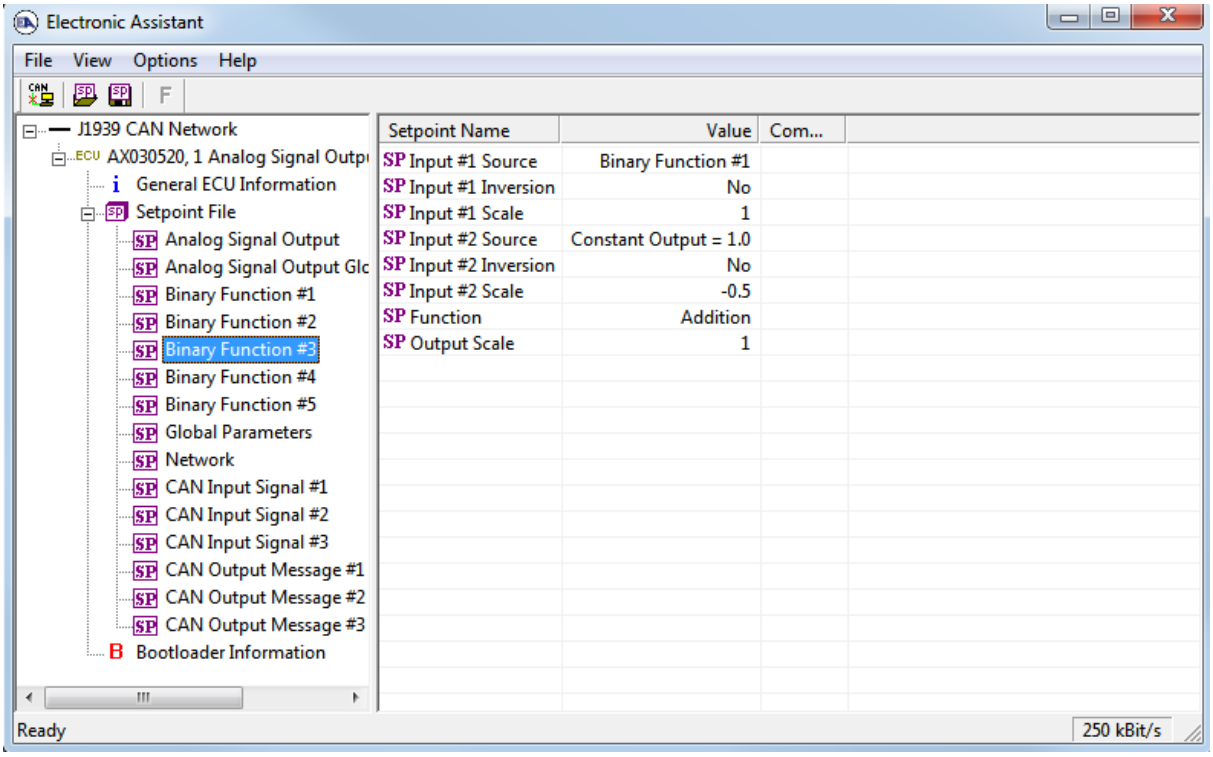
Now, when the data acquisition part of the pitch and roll CAN signals is set up, configure the data processing part of the controller to generate the displacement signal output.

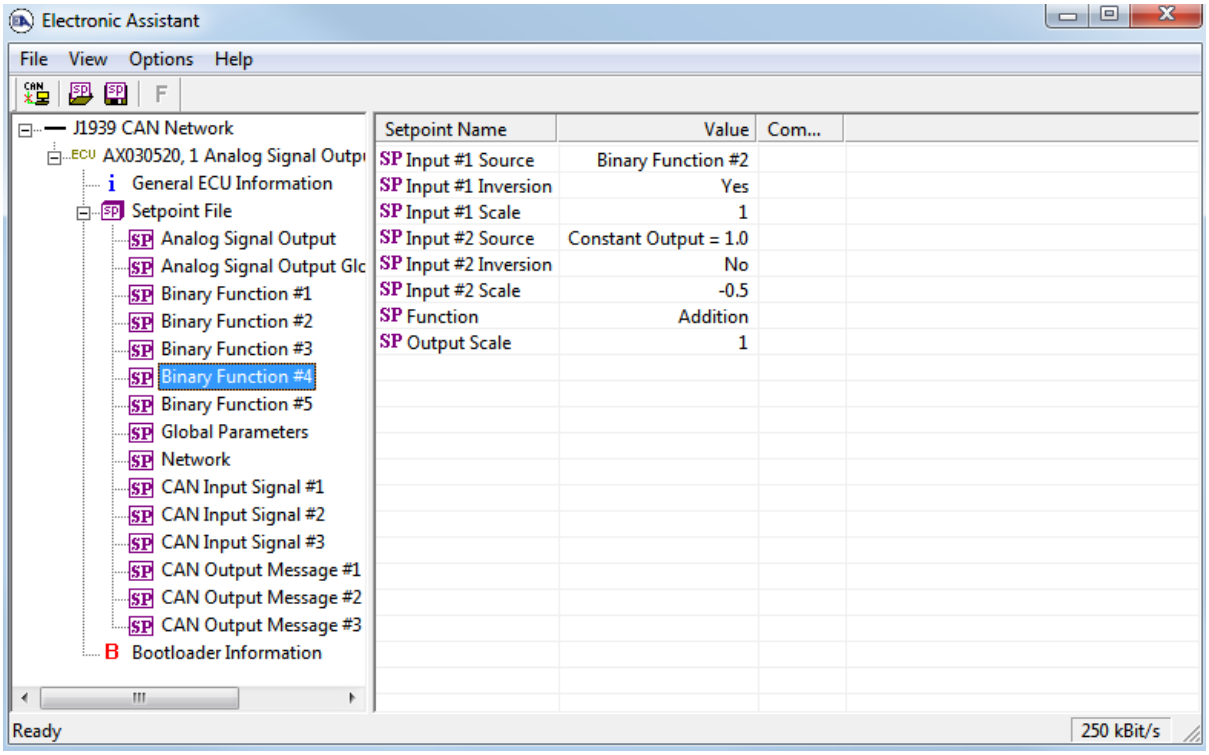
First, configure the Binary Function #1 and Binary Function #2 function blocks to calculate minimum and maximum angle values the following way:



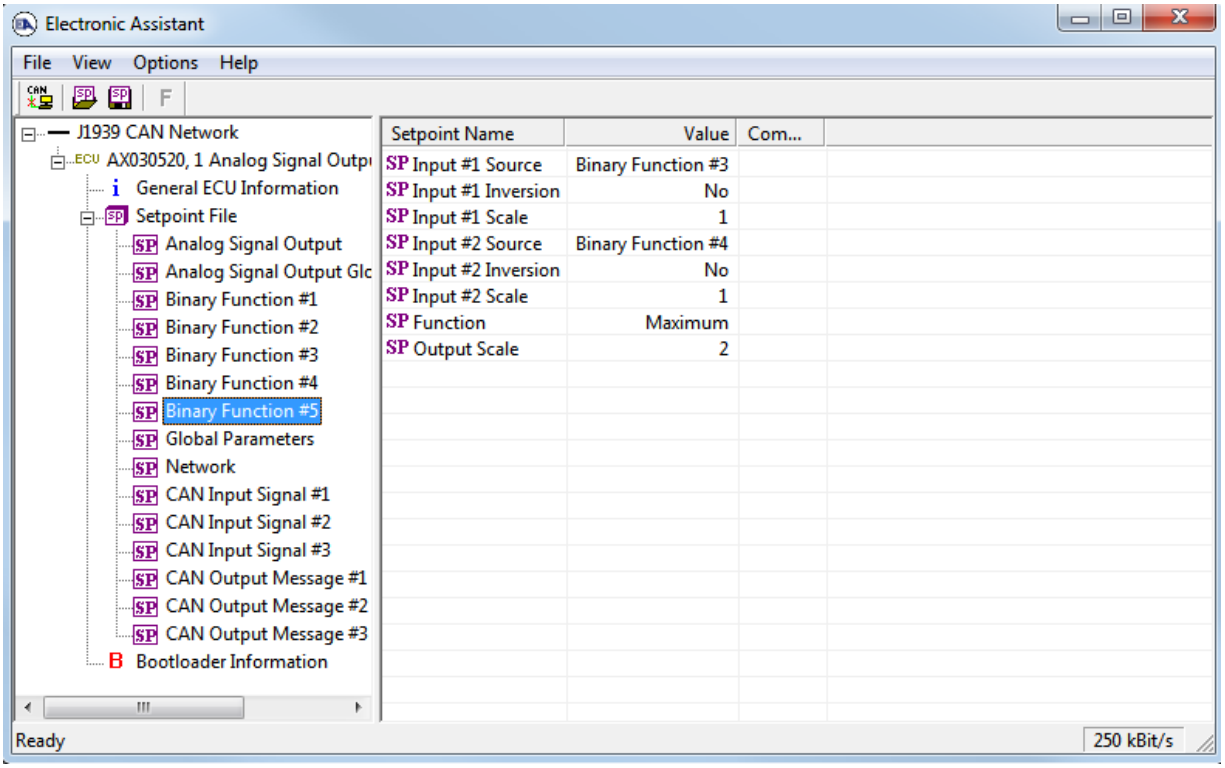


Then, calculate the displacement angle for positive and negative angle displacement using the Binary Function #2 and #3 function blocks:





Finally, calculate the maximum displacement angle using the Binary Function #5:



The calculated displacement angle value will go directly to the Analog Signal Output function block, as it has already been configured in Analog Signal Output.

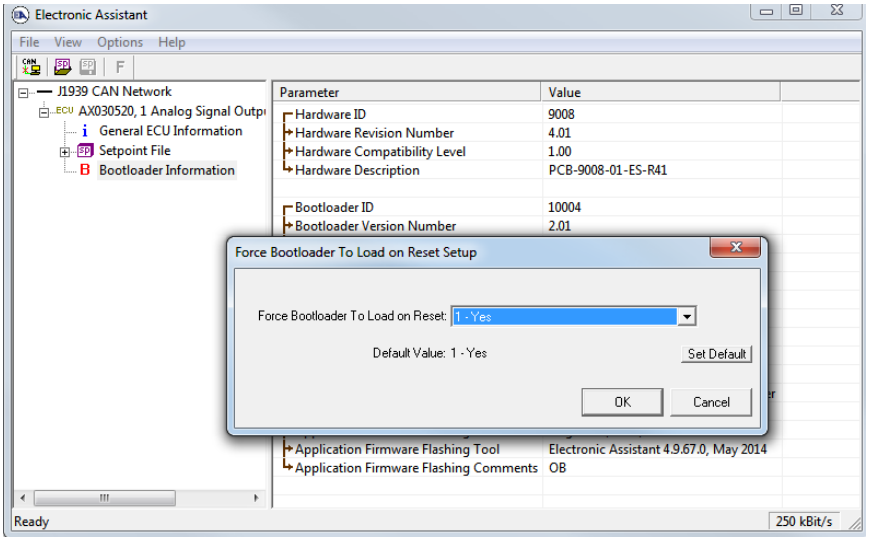
The controller configuration parameters are now programmed. The user can save the controller configuration into a setpoint file for the future reference or for configuring of the other controllers.

6 FLASHING NEW FIRMWARE

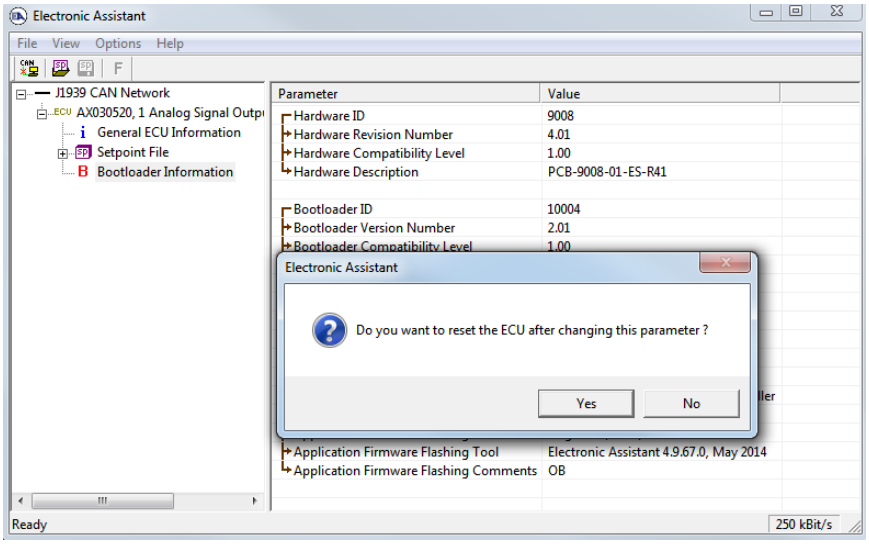
The user can flash the new controller firmware in the field using the unit embedded bootloader. This feature is supported starting from the firmware version 3.xx and higher.

Please note, that the bootloader version 2.xx, shipped in units with the application firmware version 3.xx, is not compatible with the newer bootloader versions. The user should contact Axiomatic to receive an application firmware file compatible with the old bootloader version to update the controller firmware.

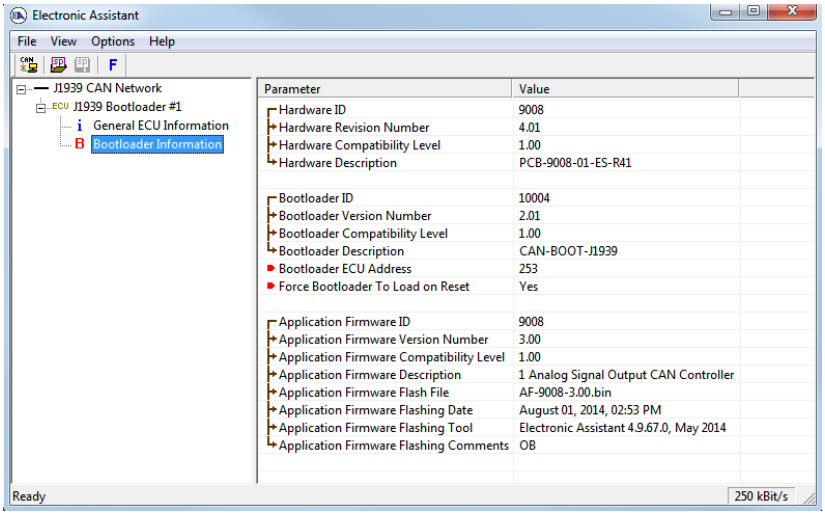
To flash the new firmware, the user should activate the embedded bootloader. To do so, start the EA and in the *Bootloader Information* group screen click on the *Force Bootloader to Load on Reset* parameter. The following dialog will appear:



The EA will prompt the user to change the *Force Bootloader to Load on Reset* parameter flag to “Yes”. This will automatically activate the bootloader on the next ECU reset. After accepting the change, the next screen will ask the user if the reset is actually required. Select “Yes”.

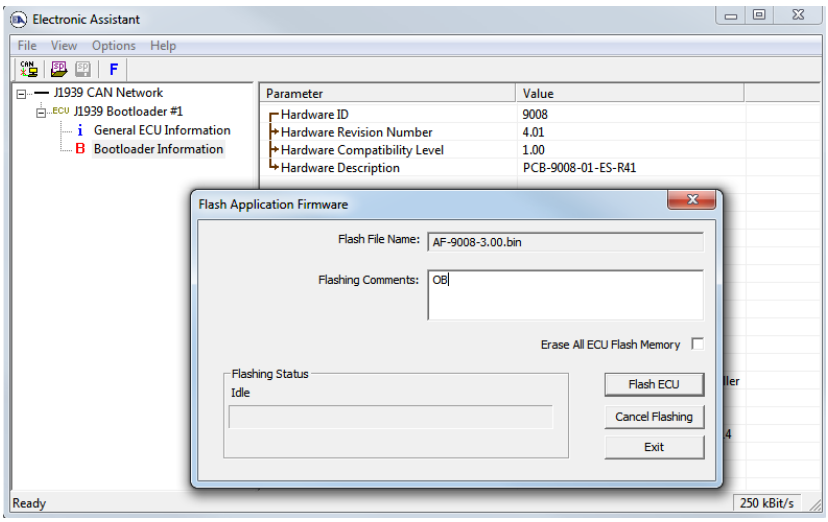


After automatic reset, instead of *AX140601, LIN - J1939 CAN Converter*, the user will see *J1939 Bootloader* ECU in the *J1939 CAN Network* top level group in the EA. This means that the bootloader is activated and ready to accept the new firmware. All the bootloader specific information: controller hardware, bootloader details and the currently installed application firmware remains the same in the bootloader mode and the user can read it in the *Bootloader Information* group screen. The information can be slightly different for different versions of the bootloader.



At this point, the user can return to the installed controller firmware by changing the *Force Bootloader to Load on Reset* flag back to “No” and resetting the ECU.

To flash the new firmware, the user should click on F toolbar icon or from the *File* menu select the *Open Flash File* command. The *Open Application Firmware Flash File* dialog will appear. Pick up the flash file with the new converter firmware and confirm selection by pressing the *Open* button. The *Flash Application Firmware* dialog window will appear[1].

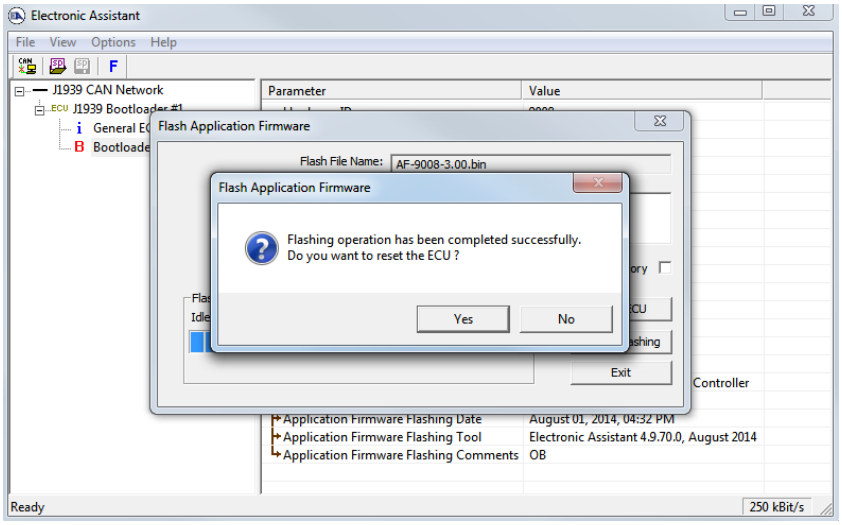


[1] In this example, instead of the new firmware the old firmware is being simply re-flashed.

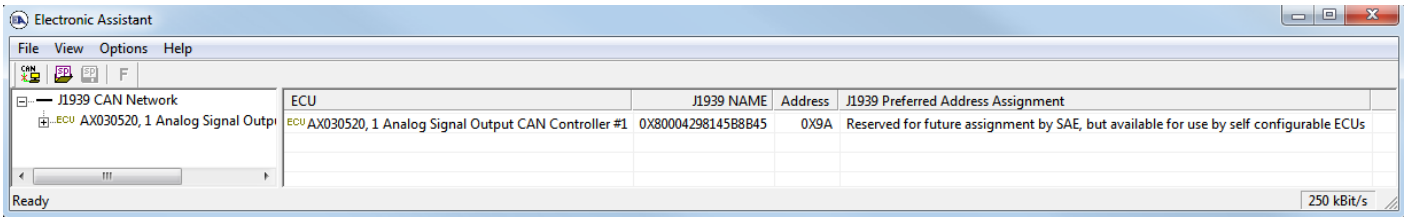
Now the user can add any comments to the flashing operation in the *Flashing Comments* field. They will be stored in the *Bootloader Information* group after flashing.

The user can also check the *Erase All ECU Flash Memory* flag to erase all configuration parameters set by the old firmware and force the controller to load the default values after flashing the new firmware.

Select the *Flash ECU* button to start flashing. A reminder that the old application firmware will be destroyed by the flashing operation will appear. Press “Ok” to continue and watch the dynamics of the flashing operation in the *Flashing Status* field. When flashing is done, the following screen will appear prompting the user to reset the ECU.



Select “Yes” and see the ECU running the new firmware. This will indicate that the flashing operation has been performed successfully.



For more information, see the *J1939 Bootloader* section of the EA user manual.

7 TECHNICAL SPECIFICATIONS

Input Specifications

Power Supply Input - Nominal	12V, 24V or 48VDC nominal (9…60 VDC power supply range)
Protection	Surge and reverse polarity protection are provided.
Input	CAN Messages, SAE J1939 {CANopen® available on request} The CAN signal can be filtered to accept messages from a single address on the network permitting a link to a specific ECU. There are three CAN Input Signal function blocks supported by the controller. Each function block can be programmed to read single-frame CAN messages and extract CAN signal data presented in virtually any user-defined signal data format. The function block then outputs the signal data to its logical output for processing by other function blocks of the controller. (Refer to Figure 1.0.) By default, the output of the first CAN Input Signal function block is connected to the input of the Analog Signal Output function block. It provides the simplest controller configuration with a direct control of the signal output by the CAN input signal. The second and third CAN Input Signal function blocks, not connected by default, can be engaged in more complicated CAN signal acquisition and processing algorithms involving Binary Function function blocks and other controller resources. The Electronic Assistant® (EA) is used to set up CAN signal acquisition and processing algorithms.

Output Specifications

CAN	The controller can send a single frame application specific CAN message to the network continuously or on request. Using the EA, the user can configure this feature.
Analog Outputs	1 analog signal output Refer to Table 1.0.
Ground Connection	1 Analog GND connection is provided.
Protection for Output + Terminal	Fully protected against short circuit to ground and short circuit to power supply rail. Unit will fail safe in the case of a short circuit condition, self-recovering when the short is removed.

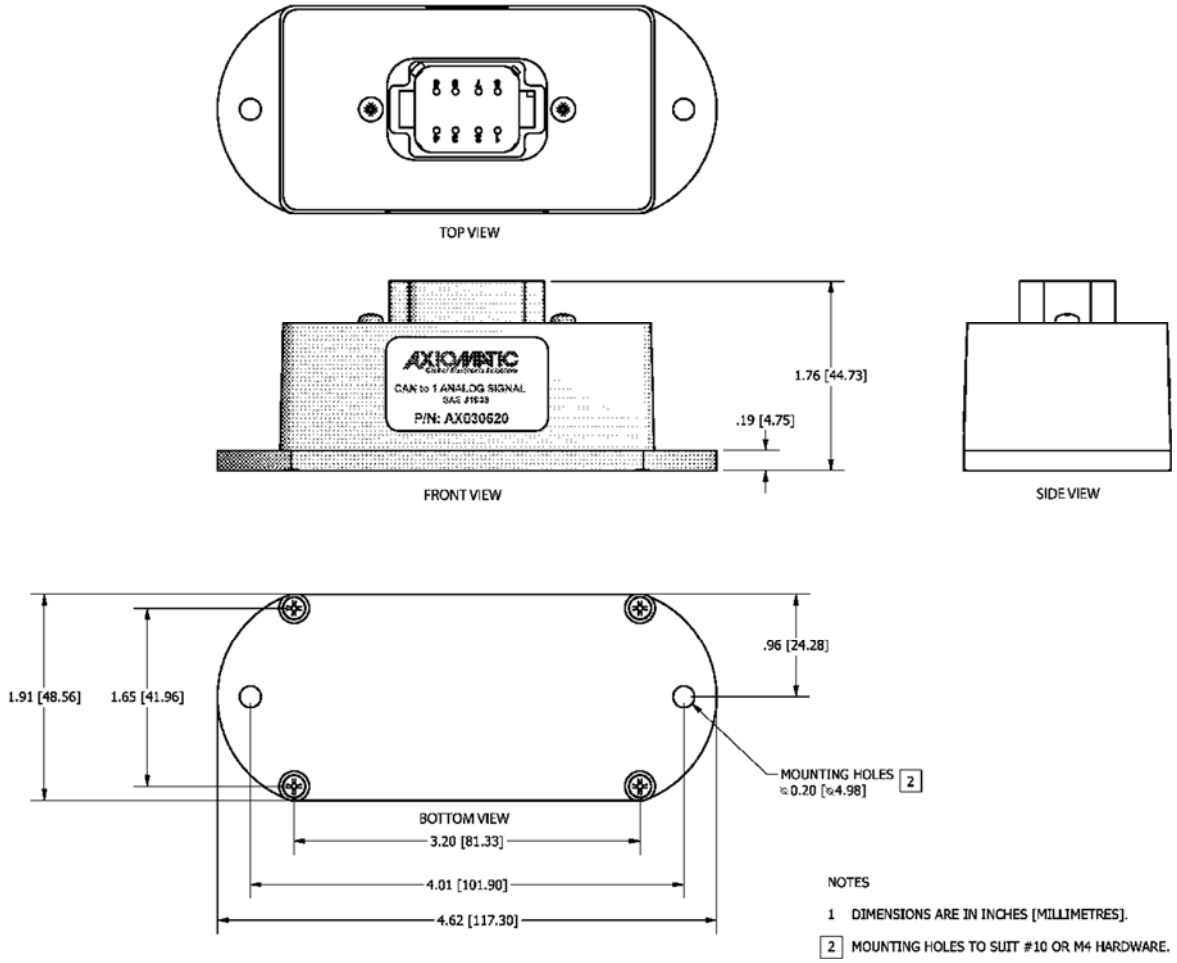
Table 1.0 - Outputs

Analog Output	1 analog signal output with embedded voltage and current monitoring circuits Using the Electronic Assistant®, the user selects: • the output mode (voltage or current); • the minimum and maximum values for the output signal from the +/-10V or +/-20 mA range. Standard analog signal ranges are supported, including: 0-5V; 0-10V; +/-5V; +/-10V; 0-20mA; or 4-20 mA. The output can be globally enabled or disabled.
Output Accuracy	≤0.5% at: +/-5V,+/-10V,+/-20mA For all other output ranges an absolute accuracy is defined by one of the above ranges within which the output range is located. For example, for 0..5V range the absolute accuracy is the same as for +/-5V range. For 0…6V range, it will be defined by +/-10V range.
Output Resolution	0.015% (>12 bit)
Voltage Monitoring Range	+/-12V
Current Monitoring Range	+/-25mA Voltage on the load should be within +/-8V voltage range.
Voltage and Current Monitoring Accuracy	≤ 1% for voltage monitoring ≤ 2% for current monitoring

General Specifications

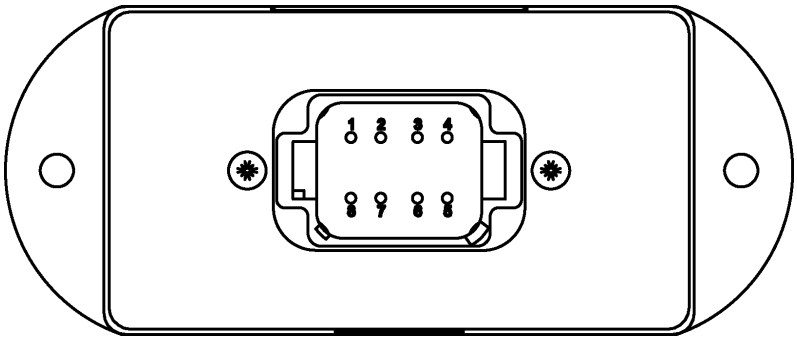
Microprocessor	32-bit, 128 KByte flash program memory
Control Logic	Standard embedded software is provided. (Application-specific control logic or factory programmed setpoints are available on request.) The controller belongs to a family of Axiomatic smart controllers with programmable internal architecture. This provides users with an ultimate flexibility, allowing them to build their own custom controller with a required functionality from a set of predefined internal function blocks using the PC-based Axiomatic Electronic Assistant® software tool. Application programming is performed through CAN interface, without disconnecting the controller from the user's system.
CAN	1 CAN port (SAE J1939) (CANopen® on request)
Slew Rate	To adjust the controller to the CAN physical network, the slew rate can be configured as fast or slow.
Monitoring and Debugging	Besides reading application signals transmitted on the CAN bus, the controller can also transmit CAN application messages carrying signals internally generated by the controller. This feature can be used for monitoring the analog signal output and for debugging purposes.
User Interface (PC-based)	The controller setpoints can be viewed and programmed using the standard J1939 memory access protocol through the CAN port and the PC-based Axiomatic Electronic Assistant®. For default setpoints, refer to the User Manual. The EA can store all controller setpoints in one setpoint file and then flash them into the controller in one operation. The setpoint file is created and stored on disk using a command *Save Setpoint File* from the EA menu or toolbar. The user then can open the setpoint file, view or print it and flash the setpoint file into the controller. The Electronic Assistant® for *Windows* operating systems comes with a royalty-free license for use on multiple computers. It requires an USB-CAN converter to link the device's CAN port to a *Windows*-based PC. An Axiomatic USB-CAN Converter AX070501 is available as part of the Axiomatic Configuration KIT. **P/N: AX070502**, the Axiomatic Configuration KIT includes the following. USB-CAN Converter P/N: AX070501 1 ft. (0.3 m) USB Cable P/N: CBL-USB-AB-MM-1.5 12 in. (30 cm) CAN Cable with female DB-9 P/N: CAB-AX070501 AX070502IN CD P/N: CD-AX070502, includes: Electronic Assistant® software; EA & USB-CAN User Manual UMAX07050X; USB-CAN drivers & documentation; CAN Assistant (Scope and Visual) software & documentation; and the SDK Software Development Kit.
Typical Quiescent Current Draw	54 mA @ 12VDC, 29 mA @ 24VDC. 17 mA @ 48 VDC
Settling Time	≤ 5 mSec. (0…95%)
Weight	0.65 lbs. (0.29 kg)
Operating Conditions	-40 to 85 °C (-40 to 185 °F)
Storage Temperature	-55 to 125 °C (-67 to 257°F)
Protection	IP67 PCB is conformal coated and protected by the housing.
Packaging and Dimensions	Encapsulated Cast Aluminum housing with mounting holes 4.62 x 1.91 x 1.76 inches (117.30 x 48.56 x 44.73 mm) L x W x H including integral connector

DIMENSIONAL DRAWING



Mounting	Mounting holes – The controller accepts 2 #10 or M4 screws. The CAN wiring is considered intrinsically safe. The power wires are not considered intrinsically safe and so in hazardous locations, they need to be located in conduit or conduit trays at all times. The module must be mounted in an enclosure in hazardous locations for this purpose. All field wiring should be suitable for the operating temperature range. Install the unit with appropriate space available for servicing and for adequate wire harness access (6 inches or 15 cm) and strain relief (12 inches or 30 cm).
Network Termination	It is necessary to terminate the network with external termination resistors. The resistors are 120 Ohm, 0.25W minimum, metal film or similar type. They should be placed between CAN_H and CAN_L terminals at both ends of the network.

Electrical Connections



Deutsch DT series 8 pin plug (DT15-08PA)

Mating plug KIT: Axiomatic P/N AX070112
(Comprised of Deutsch IPD P/n's: DT06-08SA socket, wedge W8S, 7 solid contact sockets 0462-201-16141 and 1 sealing plug 114017.)

16-18 AWG wire is recommended for use with sockets 0462-201-16141.

Use dielectric grease on the pins when installing the controller.
Wiring to these mating plugs must be in accordance with all applicable local codes. Suitable field wiring for the rated voltage and current must be used. The rating of the connecting cables must be at least 70°C. Use field wiring suitable for both minimum and maximum ambient temperature.

PIN #	FUNCTION
1	POWER +
8	POWER -
2	NOT USED
7	CAN SHIELD
3	ANALOG SIGNAL OUTPUT
6	CAN_L
4	AGND
5	CAN_H

8 REVISION HISTORY

User Manual Version	Firmware version	Electronic Assistant® (EA) version	Date	Author	Modifications
-	4.xx	-	March 22, 2018	A Wilkins	Updated connector p/n and mating plug p/n
4	4.xx	V4.9.74.00 or higher	May 20, 2015	Olek Bogush	Updated Flashing New Firmware section to reflect incompatibility of the new application firmware versions with the old bootloader.
3	3.xx	4.9.70.0 or higher	August 5, 2014	Olek Bogush	Added support for 3-Byte Continuous CAN signals. Added flashing new software functionality. Updated the entire user manual.
Rev. B for Firmware 2.xx	2.xx	3.0.34.1	May 19, 2011	Olek Bogush	In Technical Specification section updated Output Accuracy and Voltage and Current Monitoring Accuracy subsections.
Rev. A for Firmware 2.xx	2.xx	3.0.34.1	Dec 16, 2010	Olek Bogush	Added voltage and current monitors to the signal output. Added an error logical output and a disable on error feature. Analog Signal Output function block was rewritten. Added two additional CAN Output Message function blocks. Updated Technical Specifications section.
Rev. F for Firmware 1.xx	1.xx	3.0.29.0 or higher	Oct 14, 2010	Olek Bogush	Explained discrete logical inputs in the Controller Architecture section.
					Added in Technical Specifications to make a UM.
Rev. E for Firmware 1.xx	1.xx	3.0.29.0 or higher	Sept 2, 2010	Olek Bogush	The internal service port description and usage was clarified in Firmware Flashing section.
Rev. D for Firmware 1.xx	1.xx	3.0.29.0 or higher	June 16, 2010	Olek Bogush	Corrected Inverted Signal Value in a table describing signal inversion. Changed some function block drawings. Clarified Analog Signal Output function block. Clarified Analog Signal Output Global Control function block. Clarified Binary Function block. The inversion function formula Inv(...) was removed from the logical input of the Binary Function function block for simplicity. It was mentioned instead that the input can be inverted. J1939 standard document revisions were updated in the Network Support section. Added hyperlinks to function block names. Updated Fig. 1, 2, 3.
Rev. C for Firmware 1.xx	1.xx	3.0.29.0 or higher	Dec 15, 2009	Olek Bogush	Added detailed description of the data source states other than “Valid Data”. Added rules for conversion of the logical signals and CAN signal codes into each other.

Rev. B for Firmware 1.xx	1.xx	3.0.29.0 or higher	Dec 2, 2009	Olek Bogush	Added a paragraph to the Global Parameters function block description clarifying the supply voltage logical output.
Rev. A for Firmware 1.xx	1.xx	3.0.29.0 or higher	Oct 19, 2009	Olek Bogush	Initial release.

The user manual before version 3 (firmware 3.xx) was maintained as a programming manual and had a different naming convention.