User Manual UMAX03050x
Version 2

USER MANUAL

4 Analog Signal Output
CAN (SAE J1939) Controller

P/N: AX030500 (250 kBit/s)
P/N: AX030502 (500 kBit/s)
P/N: AX030503 (1 MBit/s)

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) Note: CANopen® is a registered community trade mark of CAN in Automation e.V.
DM	Diagnostic message. Defined in J1939/73 standard
EA	Electronic Assistant™. PC application software from Axiomatic, primary designed to view and program Axiomatic control 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: architecture, functionality, and application programming of the 4 Analog Signal Output CAN (SAE J1939) Controller. It also contains technical specification 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 on the controller is covered by this user manual. It can be done using Axiomatic 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 controller application firmware V2.xx. Updates specific to the user manual are done by adding letters to the user manual version number, see Version History section.

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.

The baud rate of the CAN interface is not adjustable. The user should order the controller part number with the necessary baud rate. Application firmware for a unit with one baud rate cannot be flashed in a unit with a different baud rate.

2 CONTROLLER DESCRIPTION

The controller is designed to monitor application signals transmitted on the CAN bus by various ECUs using four universal analog signal outputs. Each of the outputs 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 monitoring logic is required.

Besides monitoring application signals, the controller can also transmit a CAN application message carrying signals internally generated by the controller. This feature can be used for monitoring and debugging purposes.

2.1 Hardware Block Diagram

The controller contains: four independent universal signal output channels, a CAN port and an isolated power supply, which isolates signal output channels and the CAN port from the power supply lines. An embedded 32-bit microcontroller provides necessary functionality of the controller.

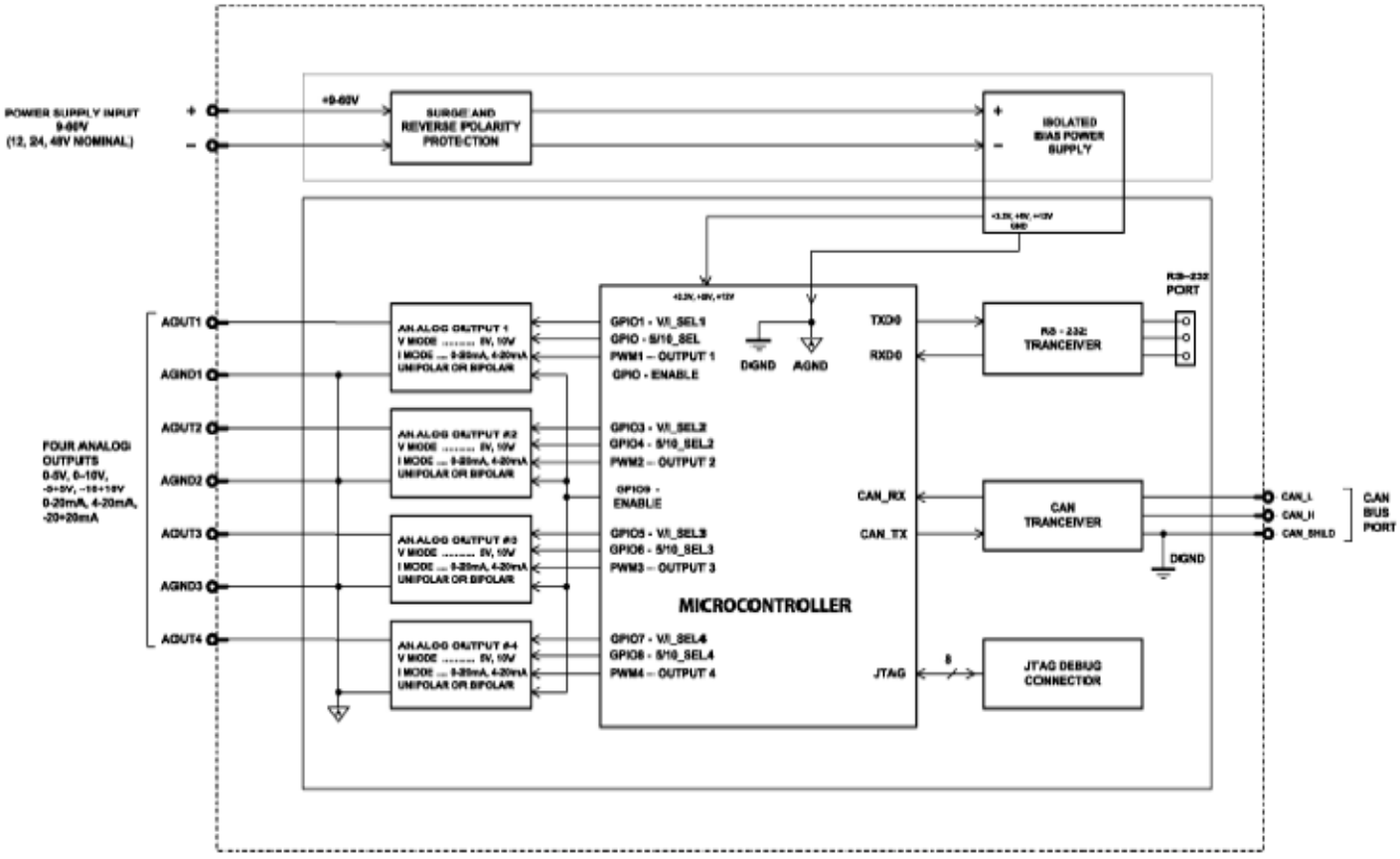


Figure 1. The Controller Hardware Block Diagram.

The controller has a wide range of protections features. The controller power supply has a transient and reverse polarity protection, and all signal outputs have overvoltage and overcurrent protection.

2.2 Software Organization

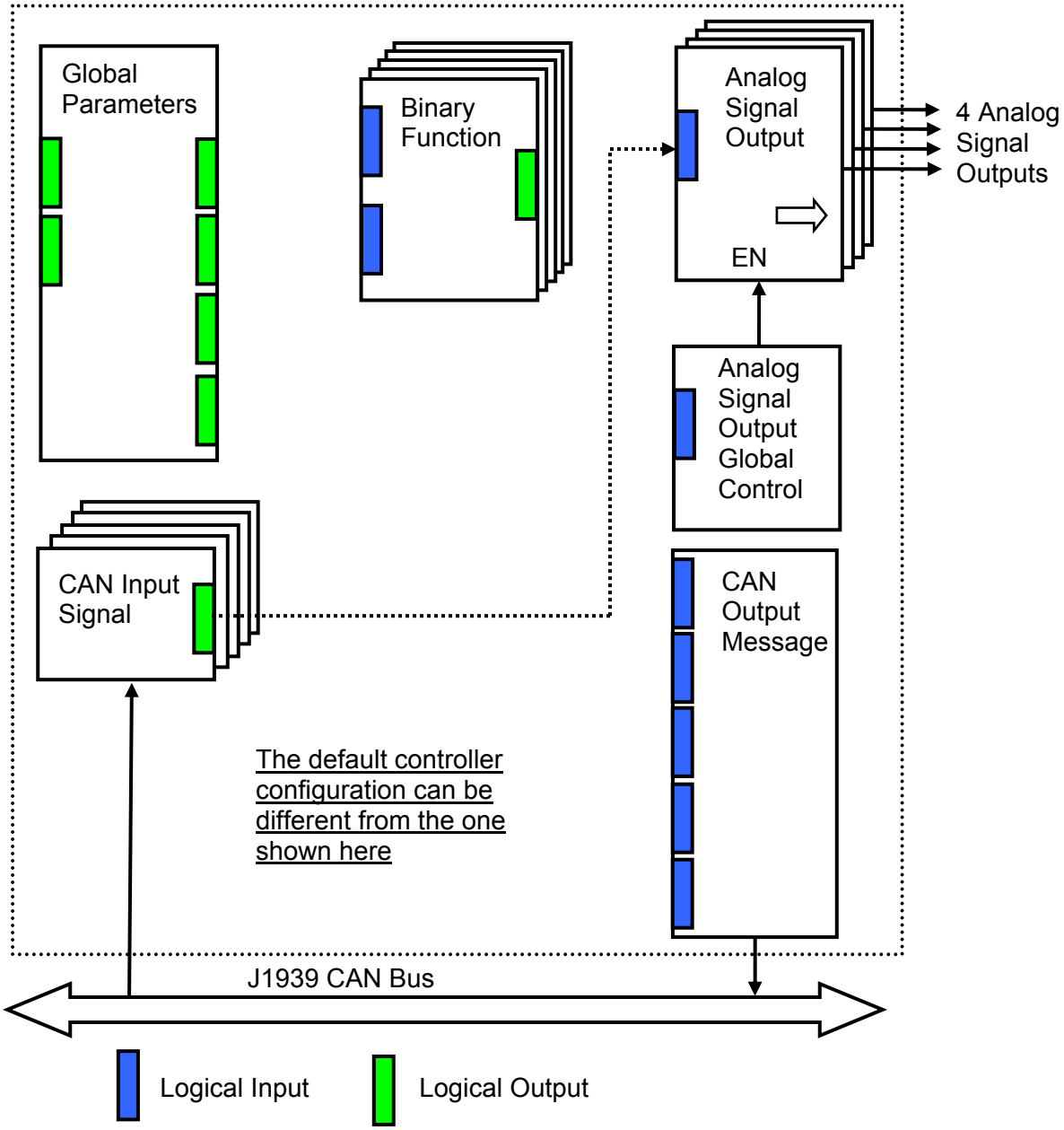
The 4 Analog Signal Output CAN Controller belongs to a family of Axiomatic smart controllers with programmable internal architecture. This architecture allows building a control algorithm based on a set of predefined internal configurable function blocks without the need for custom software.

The user can set the controller structure and configure the controller function blocks using PC-based Axiomatic Electronic Assistant® (EA) software through CAN interface, without disconnecting the controller from the user's system.

Starting from the firmware version 2.00 and EA 4.4.42.0, the controller application firmware can be updated the same way using EA in the field, see Flashing New Firmware section.

3 CONTROLLER ARCHITECTURE

From the software perspective, 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 2.



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 2.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.

3.1 Function Blocks

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 functional 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:

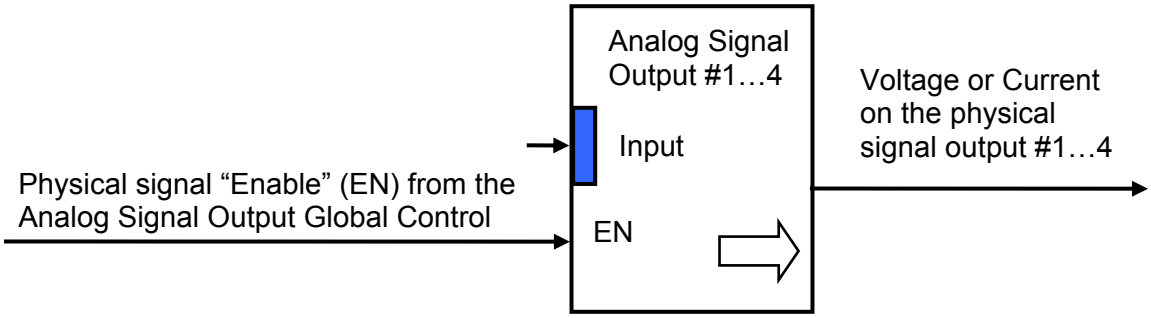
Signal State	Signal Value, Xn	Signal State Code	Inverted Signal Value	
			Xn'=Inv(Xn,Yes)	Xn'=Inv(Xn,No)
Valid Data	[0;1]	0	1-Xn	Xn
Special	0	0…4294967295 (0…0xFFFFFFFF) – Special State Code	0	0
Error	0	0…4294967295 (0…0xFFFFFFFF) – Error Code	0	0
Not Available	0	0	0	0

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.

3.2 Analog Signal Output

There are four Analog Signal Output function blocks representing analog signal outputs of the controller. Each function block can be programmed to output voltage or current in the user-specified range. All output signals can be globally enabled or disabled through the Analog Signal Output Global Control function block.

The Analog Signal Output function block has one logical input receiving a normalized signal to be output from the physical signal output. It is internally connected with the Analog Signal Output Global Control function block, which enables or disables all physical outputs.



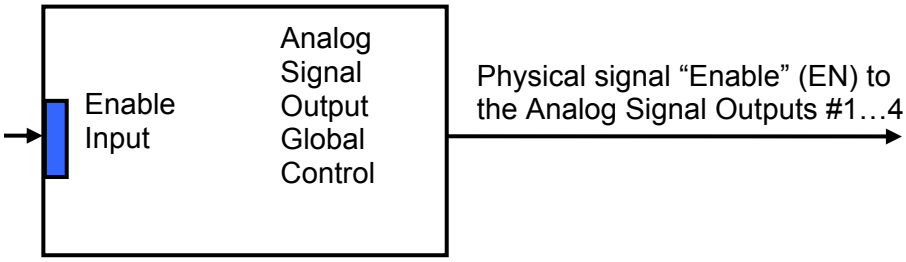
The function block setpoints are presented in the following table:

Name	Default Value	Range	Units	Description
Input Source	CAN Input Signal #1…4[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	

[1] CAN Input Signal number is the same as the number of the Analog Signal Output function block.

3.3 Analog Signal Output Global Control

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



The function block setpoints are defined as follows:

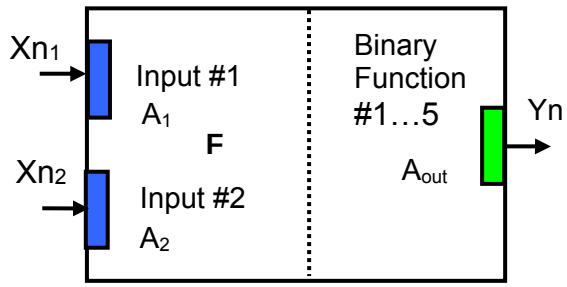
Name	Default Value	Range	Units	Description
Enable Input Source	Constant Output = 1.0	Any logical output of any function block or “Not Connected”	–	Defines an input signal source to enable all signal outputs.

Name	Default Value	Range	Units	Description
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 all analog signal outputs by default.

3.4 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. 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.

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	[-1…1] or Any value*	–	Input #1 signal scale coefficient
Input #2 Source	Not Connected	Any logical output of any function block or “Not	–	Source of the input #2 signal

Name	Default Value	Range	Units	Description
		Connected"		
Input #2 Inversion	No	{Yes, No}	–	Specifies, whether to invert the input #2 signal
Input #2 Scale	1.0	[-1…1] or 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	[-1…1] or Any value*	–	Output signal scale coefficient

*Any scale value can be programmed using EA version 3.0.29.0 or later.

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 {≤, =, ≥ } are defined the following way:

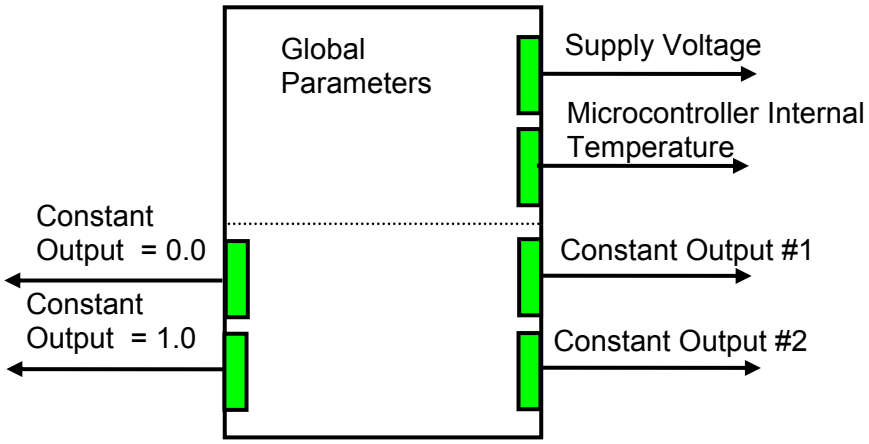
$$F^{(\leq)}[x,y] = \{1, \text{ if } x \leq y+\delta;\ 0, \text{ if } x > y+\delta\},$$

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

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

3.5 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.

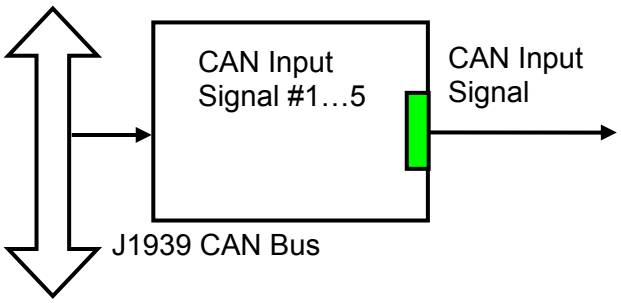
Please note, that the supply voltage sensing circuit is absent on recent controller hardware and the supply voltage logical output is set to “Not Available” state.

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	

3.6 CAN Input Signals

There are five 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 Continuous, 4-Byte Continuous}	–	Type of the CAN input signal
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	CAN continuous signal offset

Name	Default Value	Range	Units	Description
			Units	
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 outputs of the first four CAN Input Signal function blocks are connected to the appropriate inputs of the Analog Signal Output function blocks. It provides the simplest controller configuration with a direct control of the signal outputs by the CAN input signals. The fifth CAN Input Signal function block, not used 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	0…64255	Valid Data	[0;1] - normalized	0

CAN Signal Type	CAN Signal Code	Logical Signal		
		State	Value	Signal State Code
Continuous	(0…0xFAFF)		signal code	
	64256…65023 (0xFB00…0xFDFF)	Special	0	0…267 =CANSignalCode-64256
	65024…65279 (0xFExx)	Error	0	0…255 =CANSignalCode-65024
	65280…65535 (0xFFxx)	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
	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.

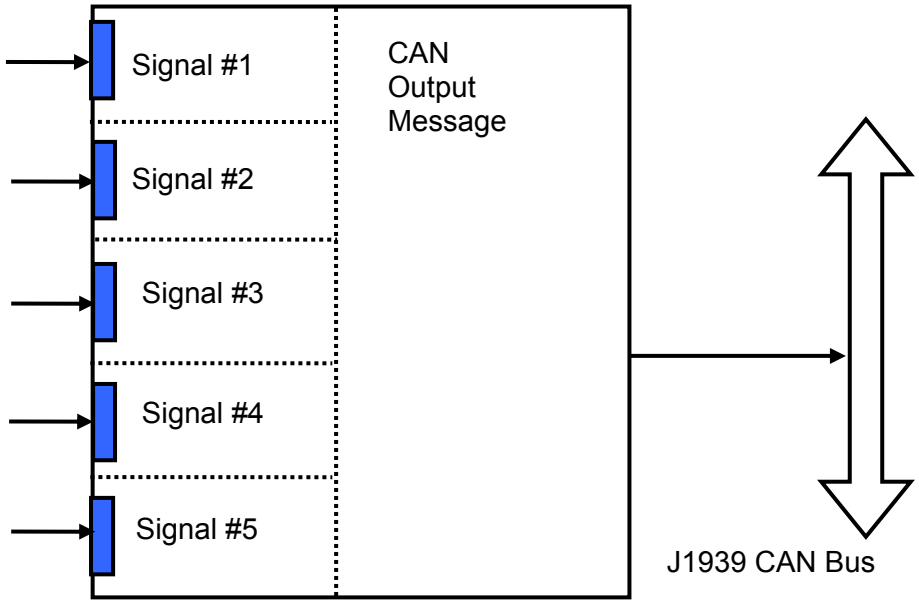
This mapping closely follows the J1939/71 standard for the 2-bit Discrete and all continuous CAN signal types, dividing the CAN code in similar ranges to represent different states of the signal. For the 1-bit and 4-bit Discrete signal types there are no generic rules specified by the J1939/71 standard 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 for reverse conversion of the logical signals into the CAN signal codes.

3.7 CAN Output Message

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



The message does not have a specific destination address. In case the PGN of the message 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, 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]

Name	Default Value	Range	Units	Description
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, 4-Byte Continuous}	–	Type of the CAN output signal #2
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, 4-Byte Continuous}	–	Type of the CAN output signal #3

Name	Default Value	Range	Units	Description
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]
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, 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

Name	Default Value	Range	Units	Description
				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, 4-Byte Continuous}	–	Type of the CAN output signal #5
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

CAN Signal Type	Logical Signal			CAN Signal Code
	State	Value	Signal State Code	
				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 (0…0xFFFFFFFF)	2
	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)
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)

CAN Signal Type	Logical Signal			CAN Signal Code
	State	Value	Signal State Code	
				=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.

Several CAN baud rates are supported. The most common J1939 250kBit/s baud rate is supported by units with P/N AX030500. Unit with P/N AX030502 supports J1939 500kBit/s baud rate. For customers requiring the maximum CAN bandwidth, a non-standard 1MBit/s baud rate is supported by the unit with P/N AX030503.

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	*For P/N: AX030500* 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.
	For P/N: AX030502 J1939/14 - Physical Layer, 500 Kbps, OCT2011.
Data Link	J1939/21 – Data Link Layer. Rev. DEC 2006
	The controller supports Transport Protocol for Commanded Address messages (PGN 65240) and software identification -SOFT messages (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. FEB 2010
	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 setpoints.

4.1 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 below:

Field Name	Field Length	Field Value	User Programmable
Arbitrary Address Capable	1 bit	1 (Capable)	No
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	18 (Nineteenth Instance)	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 address from a pool of addresses assigned to self-configurable ECUs. The address is preset to 153, 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 network address using EA, if necessary.

4.2 Slew Rate Control

The controller supporting the standard 250kBit/s baud rate has an ability to adjust the CAN transceiver slew rate for better performance on the physical network. The *Slew Rate* setpoint can be set according to the following table:

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

For the majority of J1939 250kBit/s CAN applications the slow slew rate is preferable due to the reduced EMI of the transceiver.

On the contrary, the fast slew rate is always set for controllers supporting the higher baud rates of 500kBit/s and 1Mbit/s. The user will not be able to change the slew rate from fast to slow using EA in this case.

4.3 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.

4.4 Network Setpoint Group

The following table summarizes the EA programmable setpoints 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	153	[0…253]		ECU Address
Slew Rate	*For 250kBit/s units:* Slow	{Slow, Fast}	–	Slew rate control of the CAN transceiver
	For 500kBit/s and 1Mbit/s units: Fast	Fast		

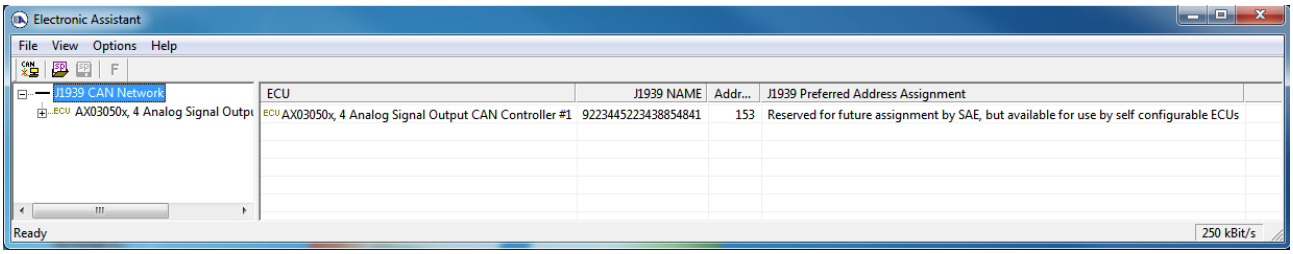
5 SETPOINT PROGRAMMING

The controller setpoints can be viewed and programmed using the standard J1939 memory access protocol through the CAN bus using Axiomatic PC-based Electronic Assistant® (EA) software.

5.1 Electronic Assistant Software

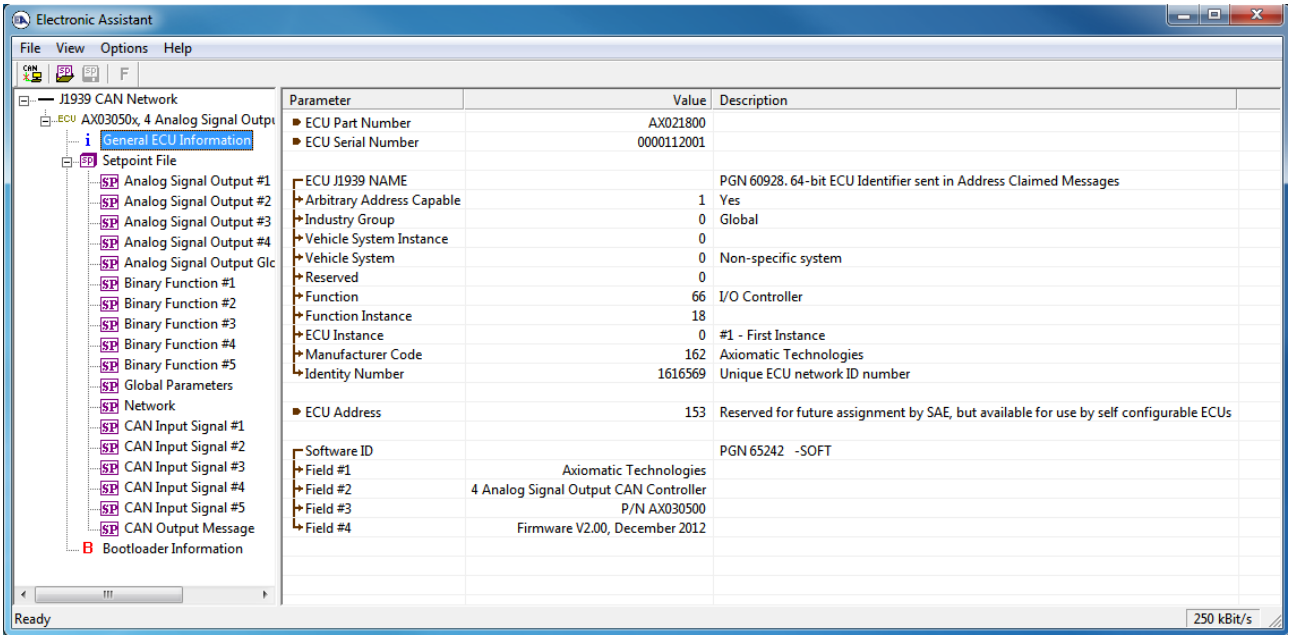
Axiomatic provides PC-based Electronic Assistant® (EA) software, together with a USB-CAN converter, as a kit P/N AX070502, to communicate with a wide range of Axiomatic controls, including this controller. Please also refer to the EA user manual UMAX07050X for the description of the EA and for the network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

To connect to the control, the user should first select the proper baud rate in EA, according to the controller part number. Upon connection, EA will show the controller on the list of controls that are present on the J1939 CAN network. If there is only one controller on the network, the following screen will appear for a 250kBit/s controller:



For 500kBit/s and 1Mbit/s controllers the baud rate in the bottom-right part of the screen will be different.

The user can then browse through the ECU parameters, read general ECU information, and Bootloader Information, view and modify setpoints:

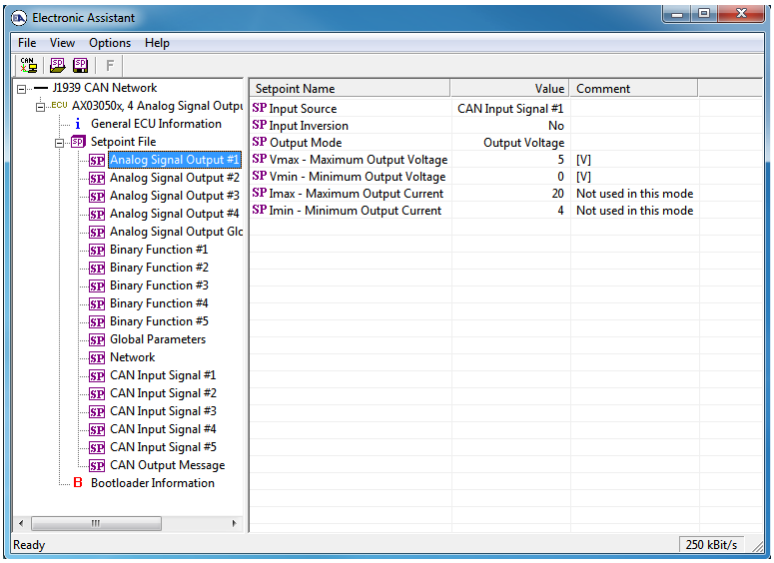


The setpoints are grouped on the base of their functionality. Please, refer to the appropriate sections of this manual describing the required function block or a setpoint group.

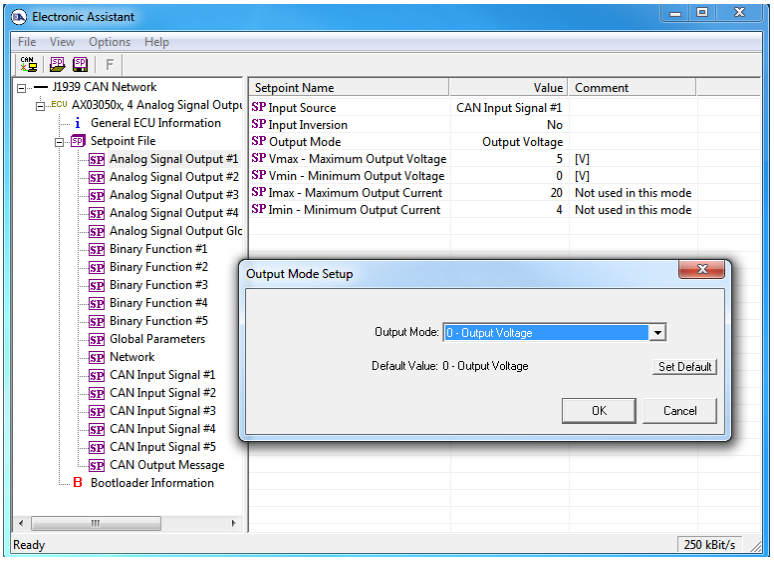
In the General ECU Information group, the user will see the version number of the application firmware. Please, make sure that the user manual version number matches with the most significant part of the application firmware version number. Otherwise, a different user manual is required to work with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

5.2 Function blocks in EA

Each controller function block is presented by its own setpoint group in the Setpoint File main group. Individual setpoints of the function blocks can be accessed through these setpoint groups:



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



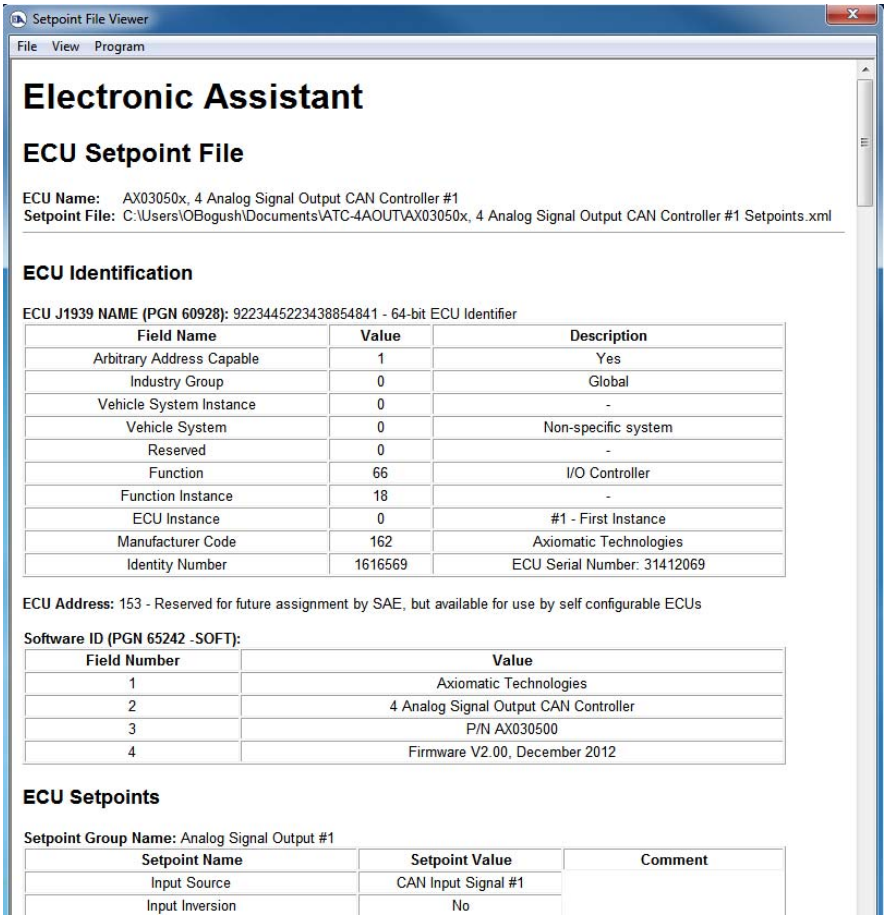
The controller will perform an internal reset of all function blocks after each change of the setpoints. If the new setpoint affects the network identification, the controller will reclaim its network address with a new network identification message, see J1939 Name and Address.

All controller function blocks are described in subsections of the Controller Architecture section. The Network setpoint group is described in the Network Setpoint Group subsection of the Network Support section of this manual.

5.3 Setpoint File

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.


Setpoint File Viewer

File View Program

Electronic Assistant

ECU Setpoint File

ECU Name: AX03050x, 4 Analog Signal Output CAN Controller #1
Setpoint File: C:\Users\OBogush\Documents\ATC-4AOUT\AX03050x, 4 Analog Signal Output CAN Controller #1 Setpoints.xml

ECU Identification

ECU J1939 NAME (PGN 60928): 9223445223438854841 - 64-bit ECU Identifier

Field Name	Value	Description
Arbitrary Address Capable	1	Yes
Industry Group	0	Global
Vehicle System Instance	0	-
Vehicle System	0	Non-specific system
Reserved	0	-
Function	66	I/O Controller
Function Instance	18	-
ECU Instance	0	#1 - First Instance
Manufacturer Code	162	Axiomatic Technologies
Identity Number	1616569	ECU Serial Number: 31412069

ECU Address: 153 - Reserved for future assignment by SAE, but available for use by self configurable ECUs

Software ID (PGN 65242 -SOFT):

Field Number	Value
1	Axiomatic Technologies
2	4 Analog Signal Output CAN Controller
3	P/N AX030500
4	Firmware V2.00, December 2012

ECU Setpoints

Setpoint Group Name: Analog Signal Output #1

Setpoint Name	Setpoint Value	Comment
Input Source	CAN Input Signal #1	
Input Inversion	No	

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

There can be small differences in setpoints between different versions of the application firmware.

It is recommended that the user manually inspect all setpoints after flashing if the setpoint file was created by a different version of the application firmware.

5.4 Default Setpoint Settings

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

In the default configuration, the signal outputs are enabled, all outputs are set to the voltage output mode with 0…5V voltage range and connected to the appropriate CAN input signal function blocks (Figure 3). The CAN input signal function blocks are disabled by default through the *Signal Type* setpoints, which are 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.

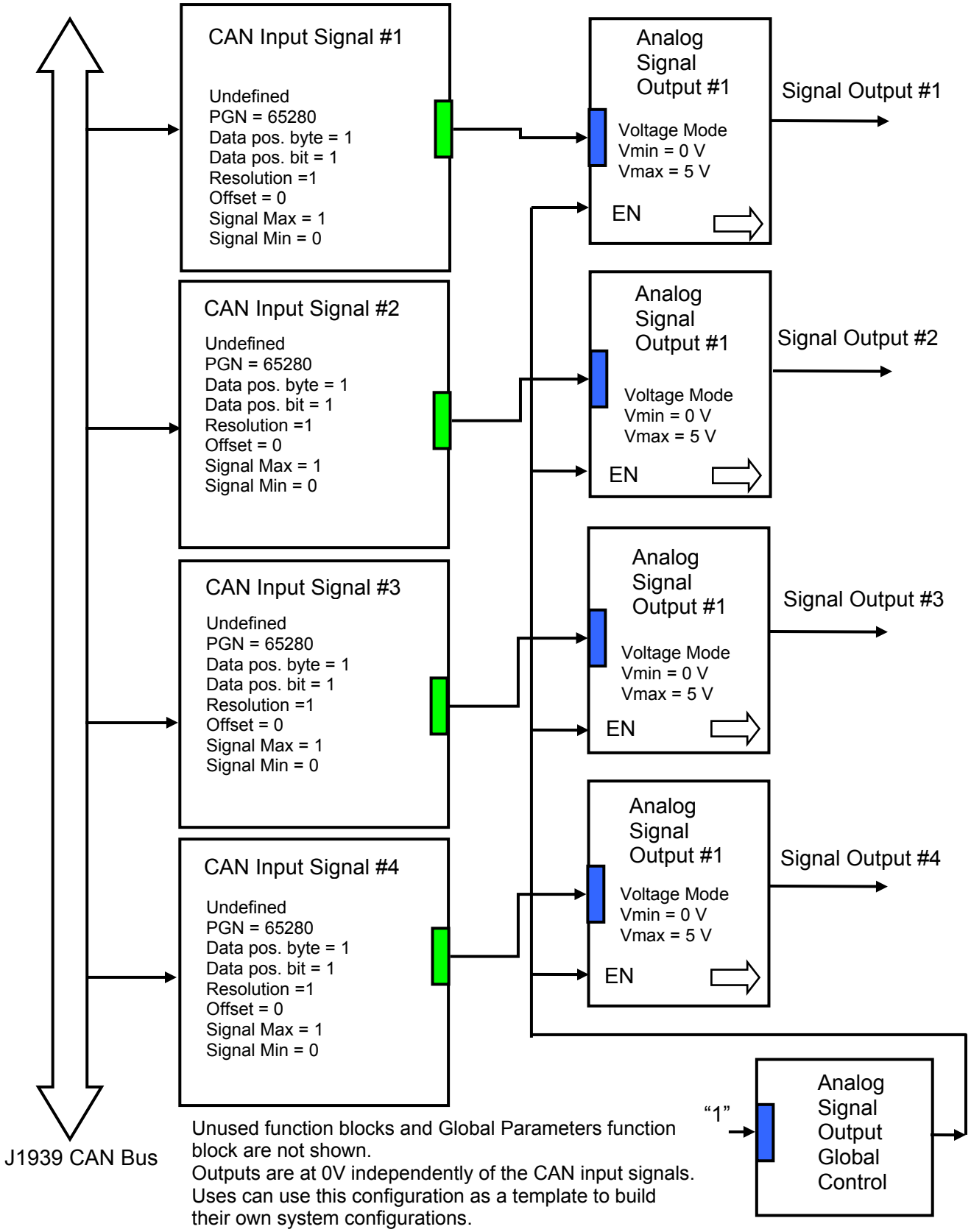


Figure 3. The Block Diagram of the Default Controller Configuration.

5.5 Setpoint Programming Example

The controller should be programmed to perform the required system functionality before being used in the system. A detailed description of the controller setpoint programming process is presented below.

5.5.1 User Requirements

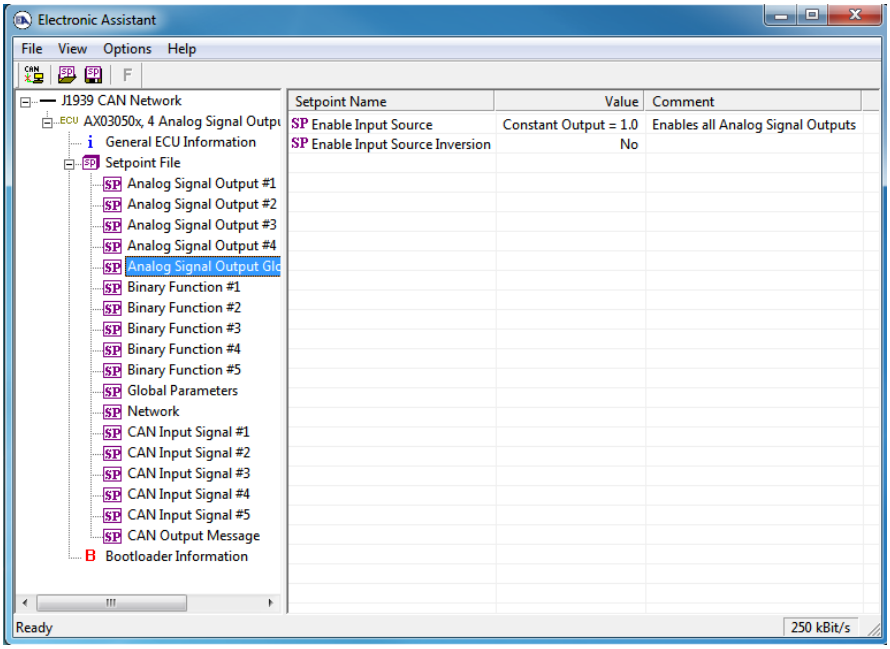
Let the controller output pitch and roll angles from a CAN inclinometer as voltages in the -5…+5 V range. Also, let the controller generate an additional emergency tilt signal when either pitch or roll angle exceeds -30º…30º range. This digital signal should have 0V value in a normal condition and switch to +5V when the pitch or roll angle exceeds the specified limit.

Let us make the following assignments: the pitch angle will be output from the Analog Signal Output #1, the roll angle – from the Analog Signal Output #2, and the emergency output – from the Analog Signal Output #3. Analog Signal Output #4 will be left unused in this application.

5.5.2 Programming Steps

First, create a block diagram of the required controller configuration using the controller function blocks (Figure 4). Then, configure the controller Analog Signal Output function blocks.

Start with enabling all analog signal outputs through the Analog Signal Output Global Control function block by setting the *Enable Input Source* setpoint to “1” using the Constant Output = 1.0 logical output from the Global Parameters function block.



Then, program with the Analog Signal Output #1 function block. Set the *Output Mode* setpoint to “Output Voltage”, the *Vmax – Maximum Output Voltage* to +5V, and the *Vmin – Minimum Output Voltage* to -5V. Also, connect the logical input of this function block to the logical output of the CAN Input Signal #1 function block using the *Input Source* setpoint. Finally, ensure that you do not invert the logical input data and the *Input Inversion* setpoint is set to “No”.

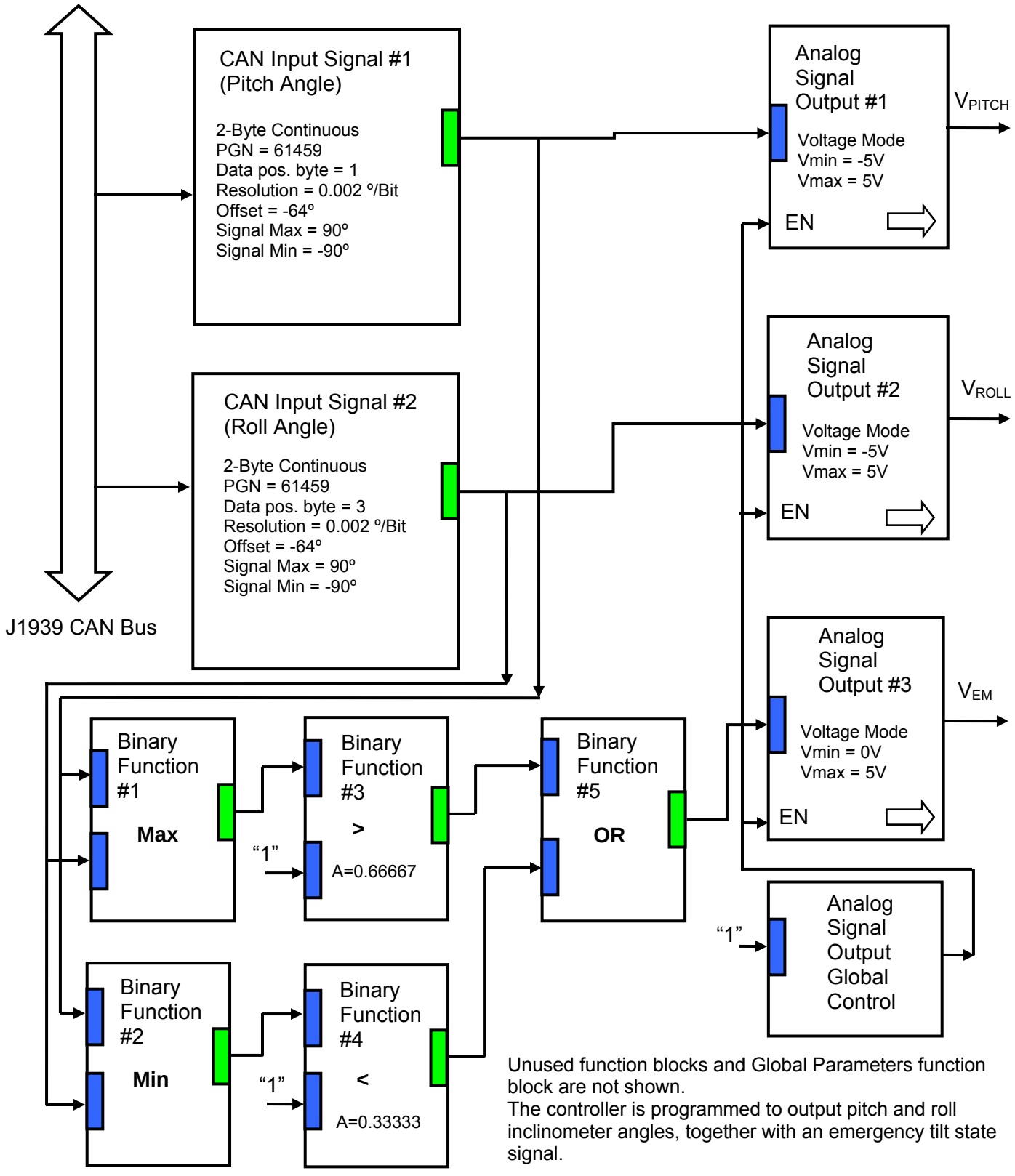
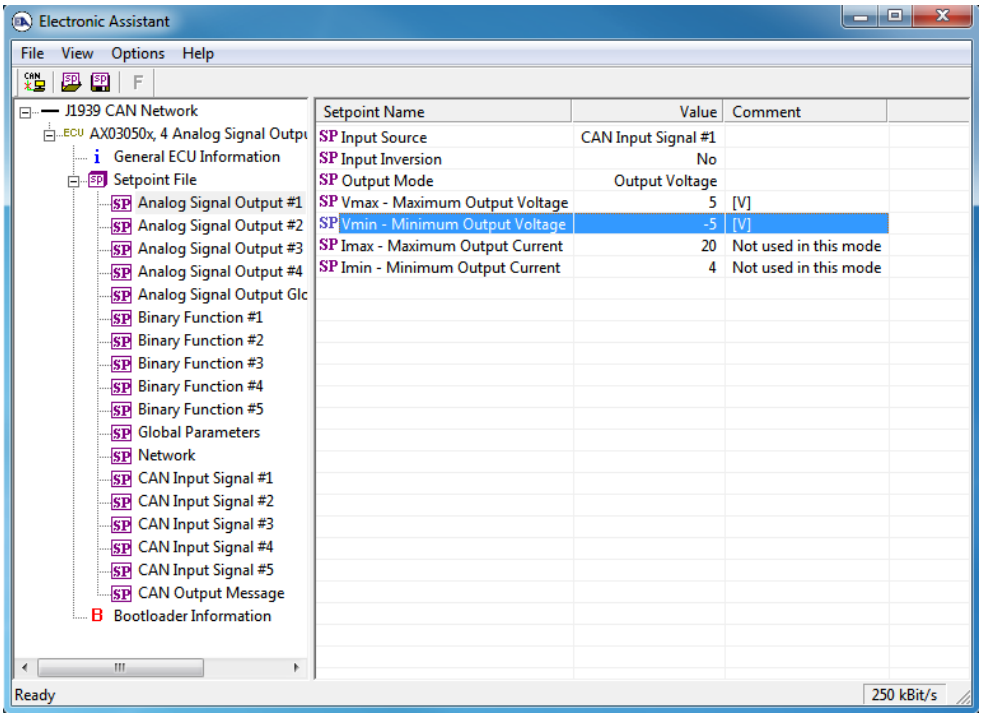
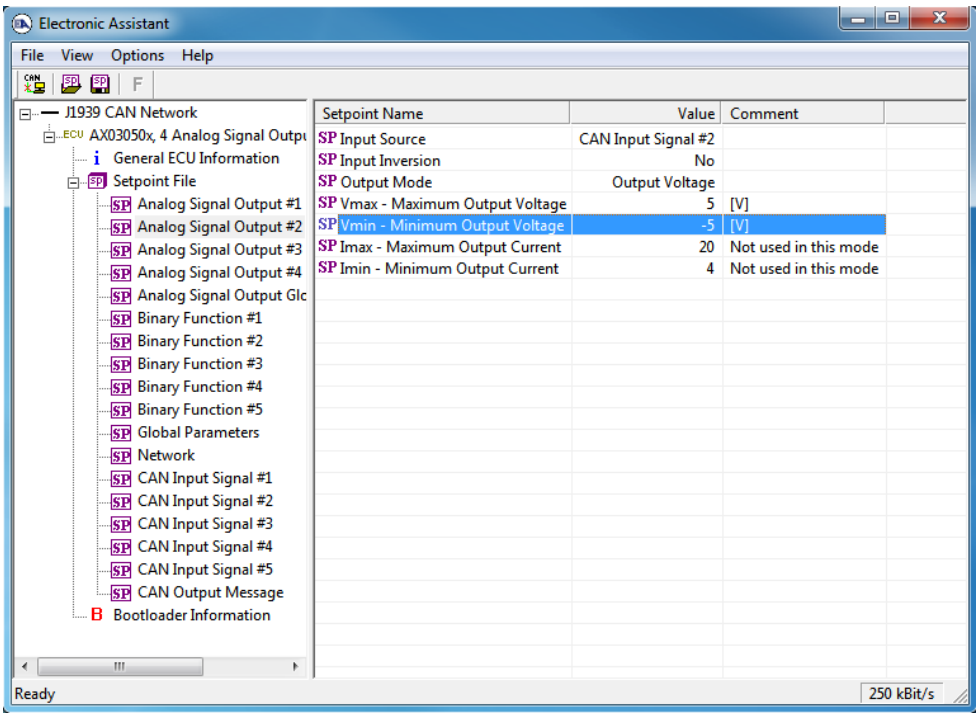


Figure 4. The Block Diagram of the Controller Configuration for the Setpoint Programming Example.



Do the same manipulations with the Analog Signal Output #2 function block, with an exception of connecting its logical input to the logical output of the CAN Input Signal #2 function block.



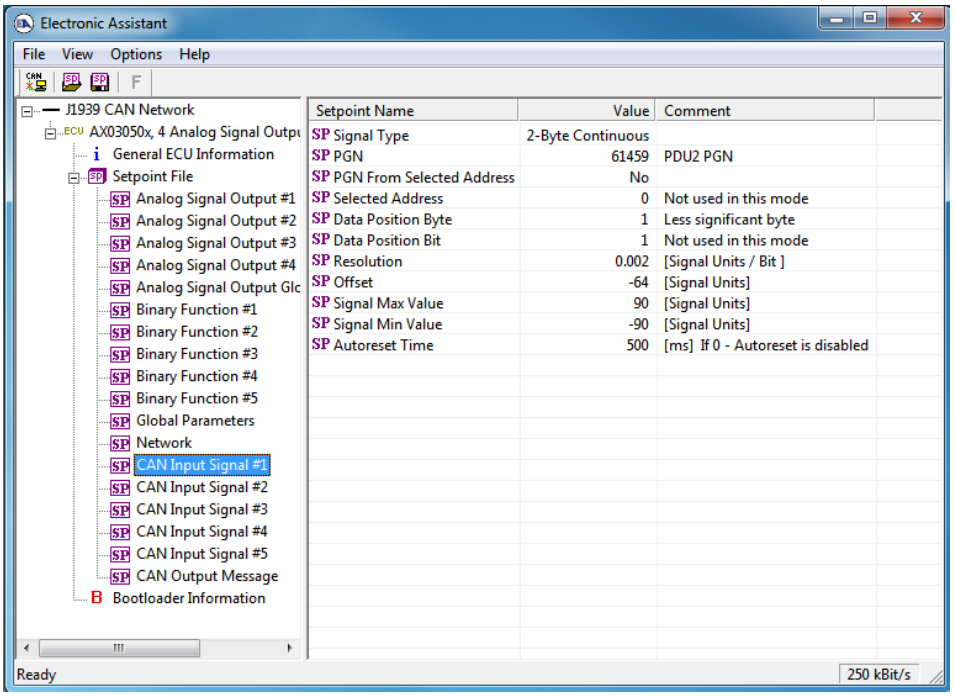
Now, configure the CAN Input Signal #1 and CAN Input Signal #2 function blocks to accept pitch and roll signals from the CAN bus. According to the J1939/71 standard, these signals are transmitted in PGN 61459 (Slope Sensor Information) as SPN 3318 (Pitch Angle) and SPN 3319 (Roll Angle), see the following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

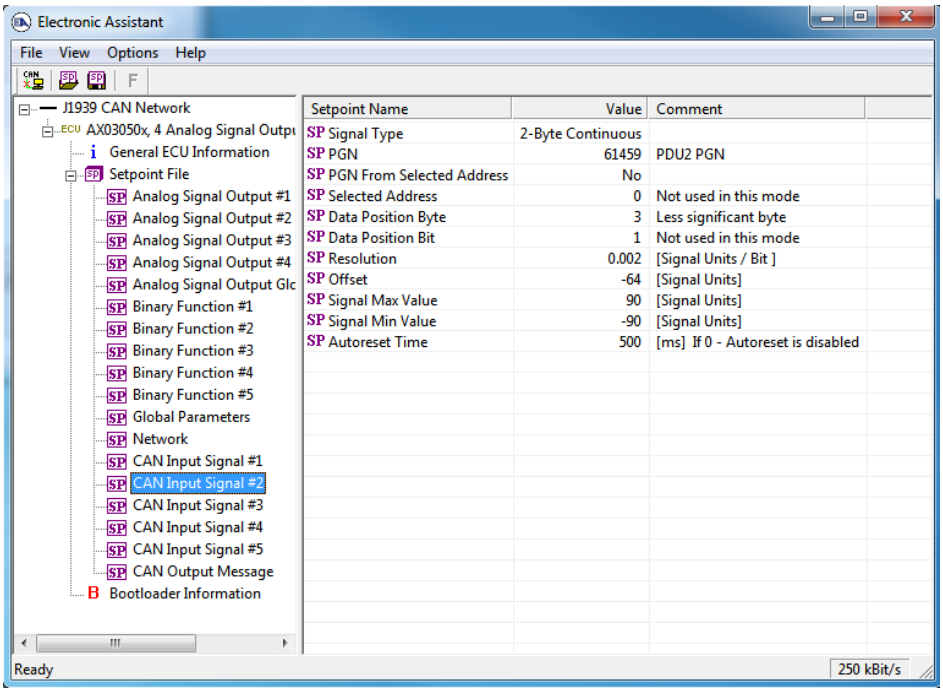
First, in the CAN Input Signal #1 function block, set *Signal Type* setpoint to “2-Byte Continuous” to meet the data length parameter of the pitch angle signal. Then, set the *PGN* setpoint to 61459 (Slope Sensor Information), the *Data Position Byte* – to 1, the *Resolution* – to 0.002 [Signal Units/Bit]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

Keep the default value “No” for the *PGN From Selected Address* setpoint unless you have several slope sensors on the network and the PGN filtering is required. Also, keep *Autoreset Time* setpoint 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.

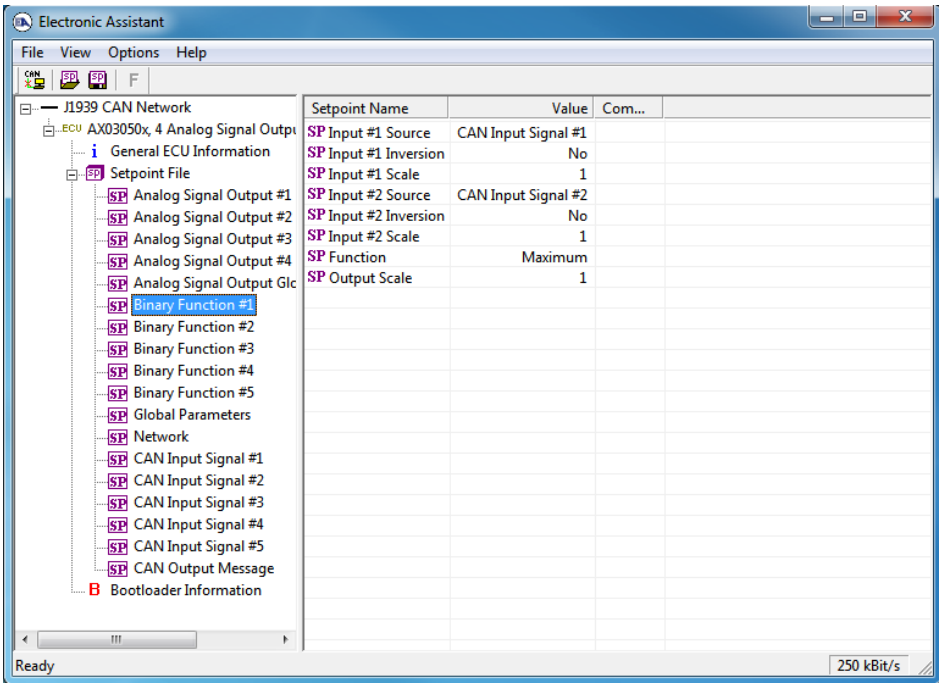


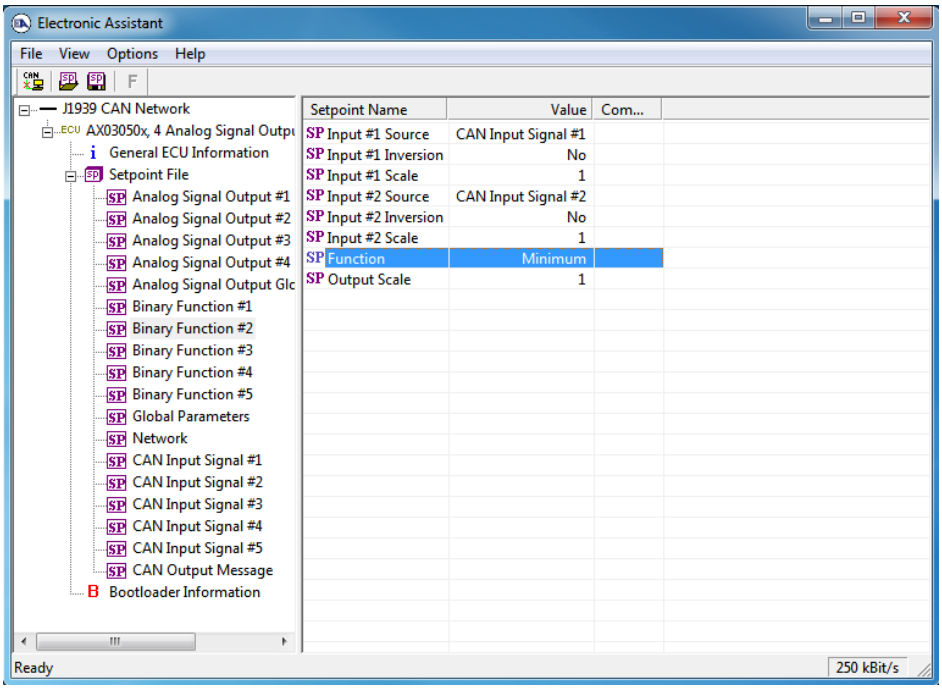
Similarly configure the CAN Input Signal #2 function block for receiving the roll angle signal.



Now, when the pitch and roll analog output signals are set up, configure the emergency signal output.

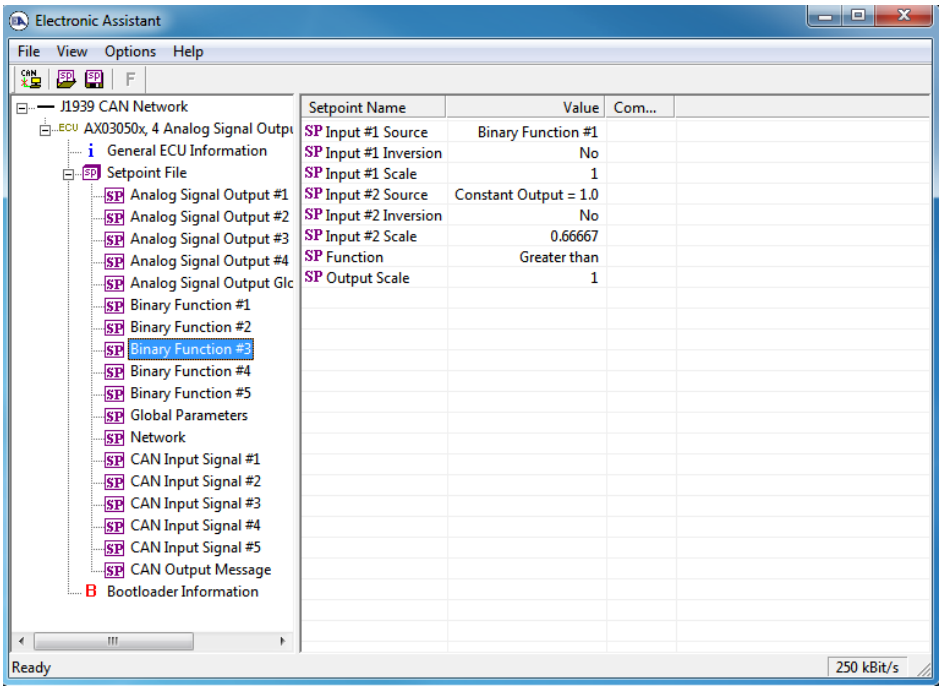
First, configure the logics of the signal using Binary Function blocks. Use Binary Function #1 function block to receive the minimum value and Binary Function #2 function block – maximum value of the pitch and roll angles. The function blocks will be set the following way:

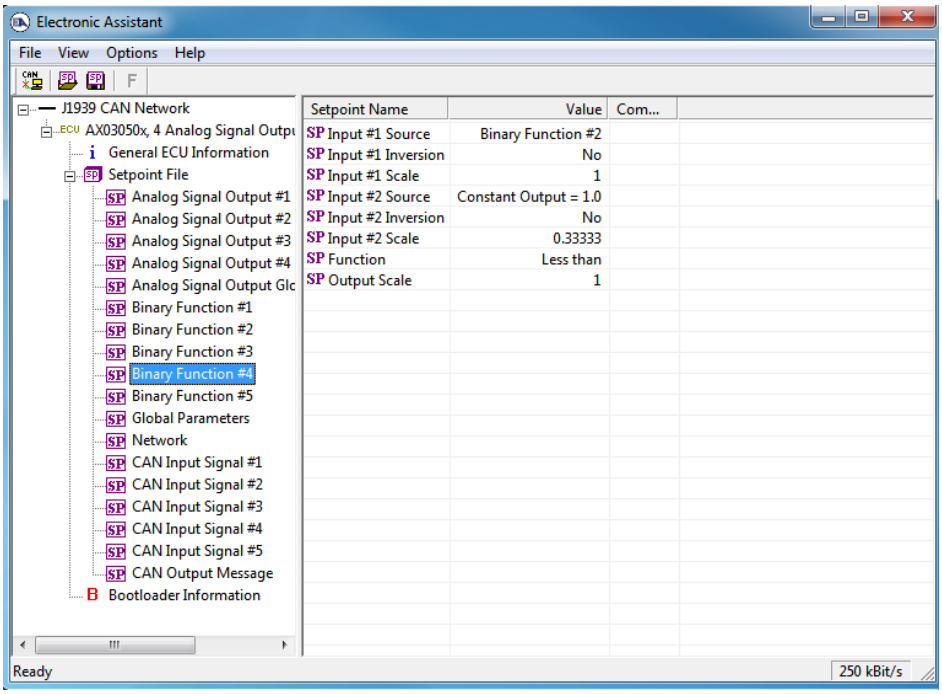




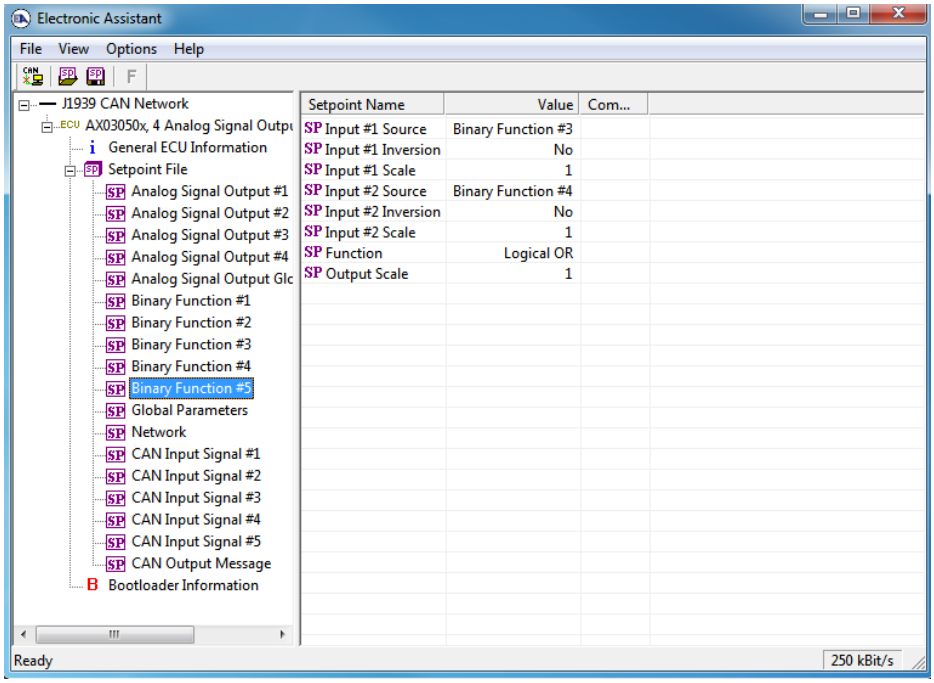
Then, configure Binary Function #3 and #4 function blocks to detect conditions when the minimum and maximum angles go out of range. Use Constant Output = 1.0 logical output together with the input scale setpoints to create appropriate thresholds.

These thresholds will be: {0.66667, 0.33333}, which are: 0.5 ± 30º/180º, for the -30º…30º angle range and -90º…90º full range scale.

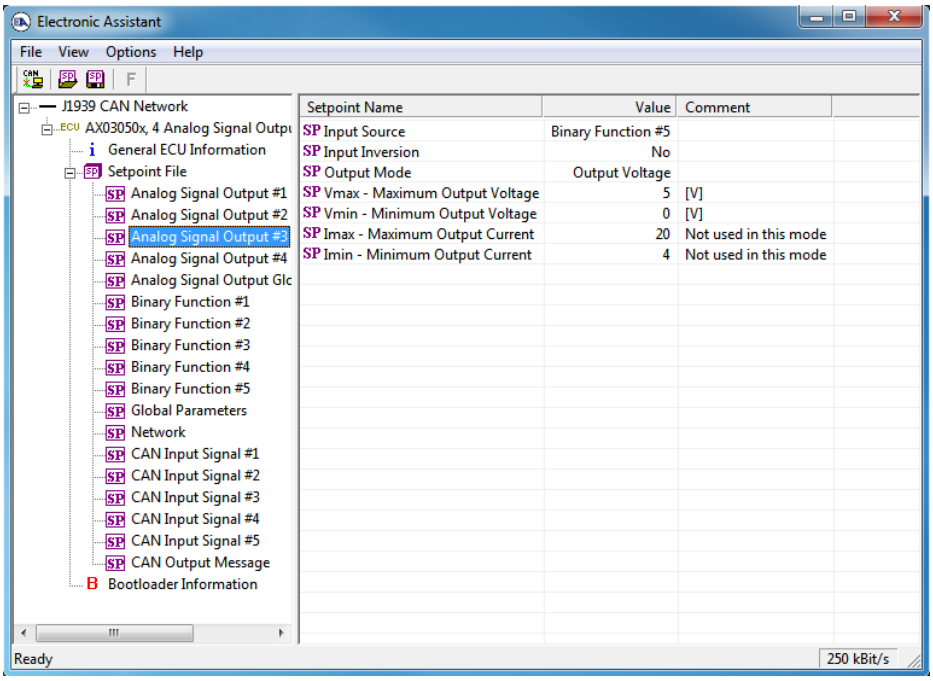




Now use Binary Function #5 function block to combine minimum and maximum “out of range” conditions into one signal using the logical OR function.



As the last step, in the Analog Signal Output #3 function block, set the *Output Mode* setpoint to “Output Voltage”, the *Vmax – Maximum Output Voltage* to 0V, the *Vmin – Minimum Output Voltage* to 5V, and the *Input Source* setpoint – to the Binary Function #5.

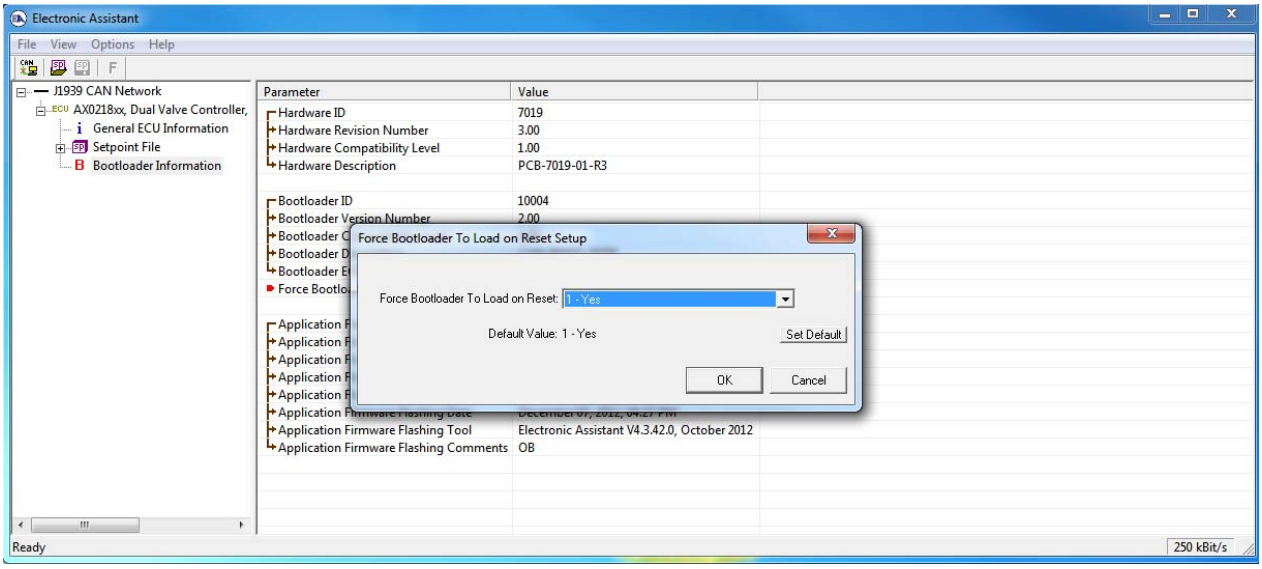


The controller setpoints are now programmed. The user can save the controller setpoint configuration into a setpoint file for the future reference or for programming of the other controllers.

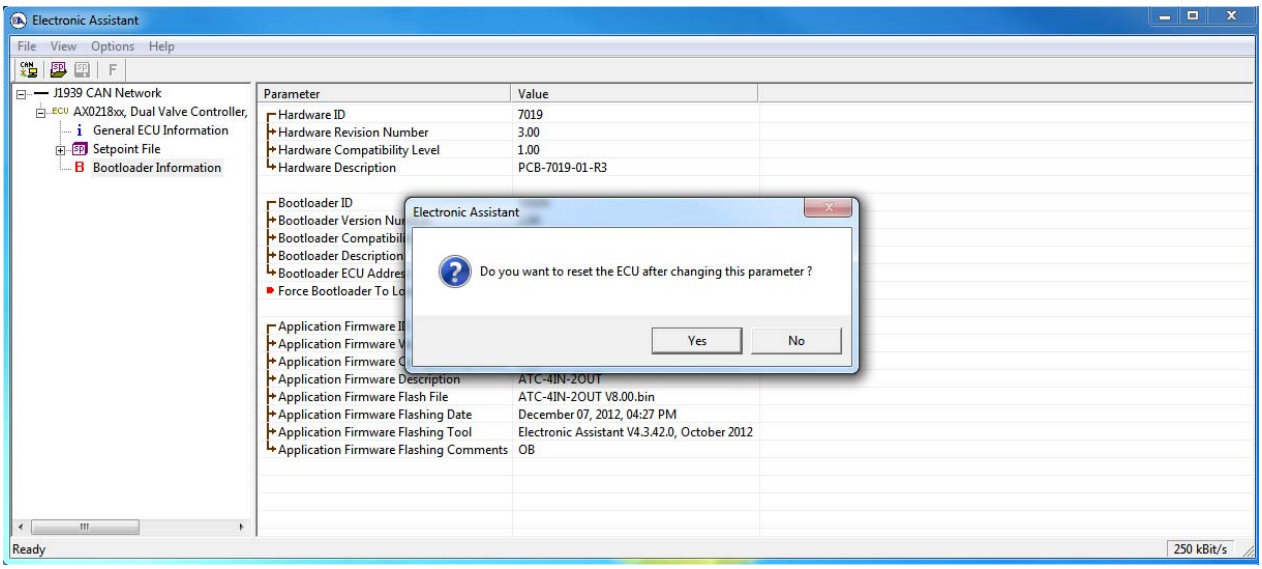
6 FLASHING NEW FIRMWARE

The controller application firmware can be updated in the field starting from V2.00. The Axiomatic EA 4.4.42.0 or later will be necessary to perform this operation. The user should contact Axiomatic to obtain a flash file with the new firmware before starting the flashing operation. Changing baud rate of the unit by flashing a new firmware supporting a different baud rate is not allowed.

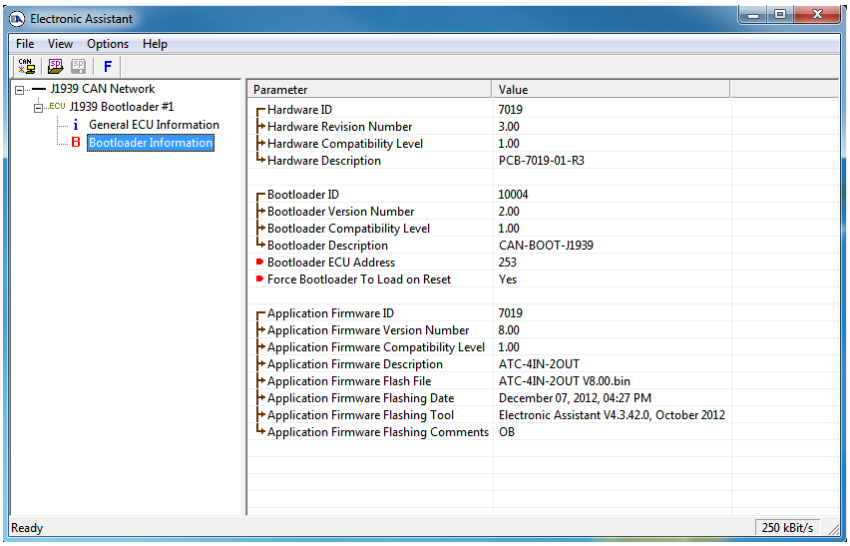
To flash the new firmware, the user should activate the embedded bootloader. To do so, start the EA and in the *Bootloader Information* group pane click on the *Force Bootloader to Load on Reset* parameter. The following dialog will appear:



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*.

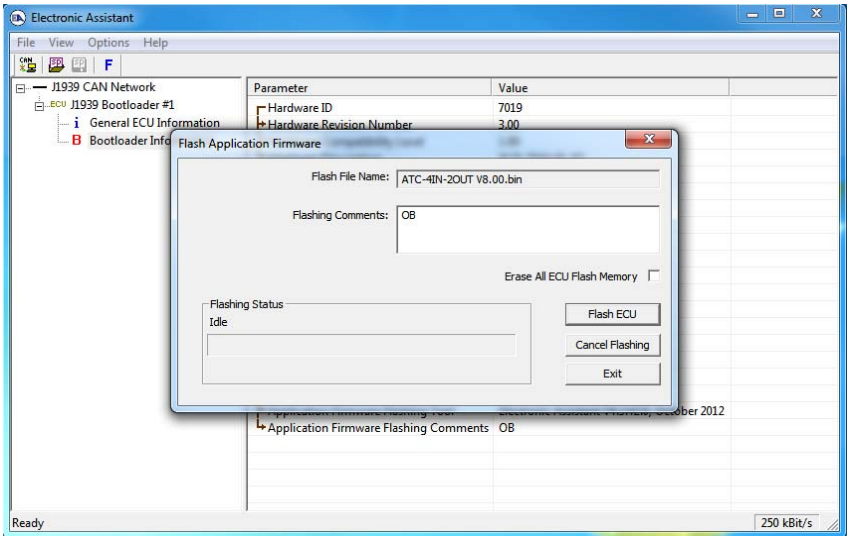


After automatic reset, instead of *AX03050x, 4 Analog Signal Output CAN Controller*, the user will see *J1939 Bootloader* ECU in the *J1939 CAN Network* top level group in the EA. This means that the bootloader is activated and ready to accept the new firmware. All the bootloader specific information: 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 pane.



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 controller firmware and confirm selection by pressing the *Open* button. The *Flash Application Firmware* dialog window will appear[1].

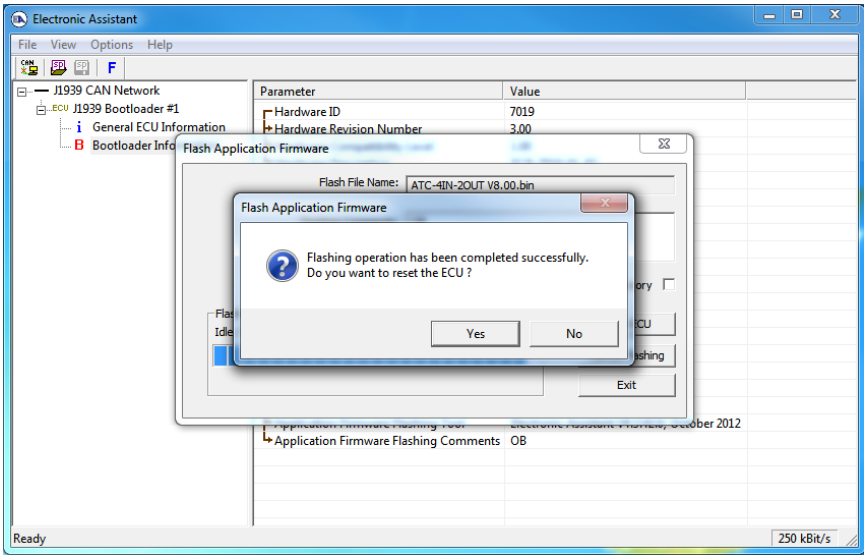


[1] In this example, instead of the new firmware the old firmware is being simply reflashed.

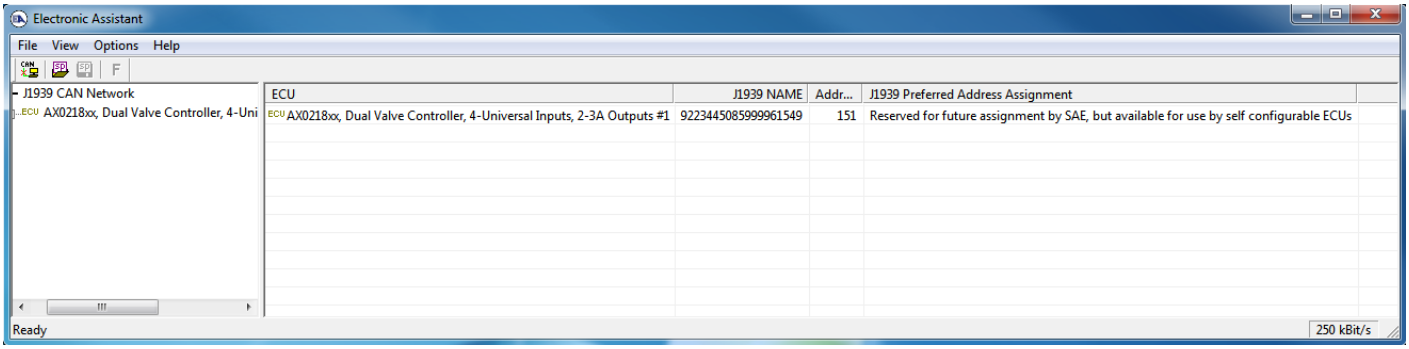
Now the user can add any comments to the flashing operation in the *Flashing Comments* field. They will be stored in the *Bootloader Information* group after flashing.

The user can also check the *Erase All ECU Flash Memory* flag to erase all setpoint values set by the old firmware and force the controller to load default setpoints 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.
Isolation	A transformer power supply provides galvanic isolation between the power supply input and the analog signal outputs.
Input	CAN Messages, SAE J1939, For baud rate see the table: AX030500 | 250 kBit/s | J1939/11, J1939/15. Most common AX030502 | 500 kBit/s | J1939/14. New standard AX030503 | 1Mbit/s | Non-standard {CANopen® (model AX030501) 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. 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. For baud rate see the table: AX030500 | 250 kBit/s | J1939/11, J1939/15. Most common AX030502 | 500 kBit/s | J1939/14. New standard AX030503 | 1Mbit/s | Non-standard
Analog Outputs	Up to 4 analog signal outputs are selectable by the user. Refer to Table 1.0.
Ground Connection	3 Analog GND connections are 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 Outputs	Up to 4 analog signal outputs are available. Using the Electronic Assistant®, the user selects: • the output mode (voltage or current); • and 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; 4-20 mA; and +/-20mA. The outputs can be globally enabled or disabled.
Output Accuracy	0.5%
Output Resolution	0.015% (>12 bit)

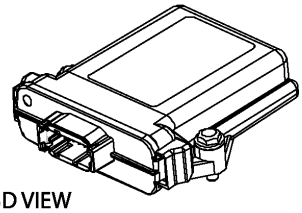
General Specifications

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.)
Monitoring and Troubleshooting	The controller can also transmit a CAN application message carrying signals internally generated by the controller. This feature can be used for monitoring and debugging purposes.
CAN	1 CAN port (SAE J1939). For baud rate see the table: AX030500 | 250 kBit/s | J1939/11, J1939/15. Most common AX030502 | 500 kBit/s | J1939/14. New standard AX030503 | 1Mbit/s | Non-standard (Model AX030501 is CANopen®)
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®. 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 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. These are downloadable from www.axiomatic.com. Go to the Log-in tab and email sales@axiomatic.com for the password.
Quiescent Current Draw	< 340 mA @ 12V and full load < 160 mA @ 24V and full load < 90 mA @ 48V and full load
Settling Time	≤ 5 mSec. (0…95% PWM) ≈ 7 mSec. (0…99% PWM)
Weight	0.50 lb. (0.23 kg)
Operating Conditions	-40 to 85 °C (-40 to 185 °F)
Storage Temperature	-55 to 125 °C (-67 to 257°F)
Vibration and Shock Compliance	MIL-STD-202G, Test 204D, 214A and 213B 7.68 Grms (Random) 10 g peak (Sine) 50 g (Shock)
Protection	IP67, PCB is conformal coated and protected by the housing.
Packaging and Dimensions	High Temperature Nylon housing - Deutsch IPD PCB Enclosure (EEC-325X4B) 4.62 x 5.24 x 1.43 inches 117.42 x 133.09 x 36.36 mm (W x L x H excluding mating plugs) Refer to the housing dimensions drawing in Installation Instructions.

8 INSTALLATION INSTRUCTIONS

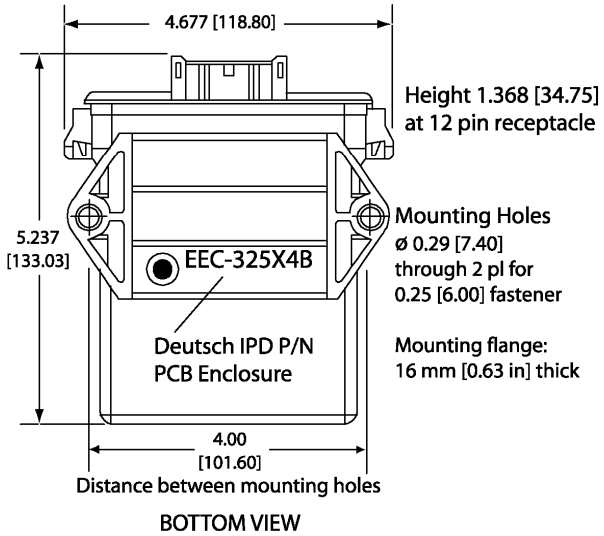
Operating Conditions	-40 to 85 °C (-40 to 185 °F)
Storage Temperature	-55 to 125 °C (-67 to 257°F)
Protection	IP67, PCB is conformal coated and protected by the housing.
Packaging and Dimensions	High Temperature Nylon housing - Deutsch IPD PCB Enclosure (EEC-325X4B) 4.62 x 5.24 x 1.43 inches 117.42 x 133.09 x 36.36 mm (W x L x H excluding mating plugs) Refer to housing dimensions drawing.
Mounting	Mounting holes sized for ¼ inch or M6 bolts. The bolt length will be determined by the end-user's mounting plate thickness. The mounting flange of the controller is 0.63 inches (16 mm) thick. If the module is mounted without an enclosure, it should be mounted vertically with connectors facing left and right to reduce likelihood of moisture entry. The CAN wiring is considered intrinsically safe. The power wires are not considered intrinsically safe and so in hazardous locations, they need to be located in conduit or conduit trays at all times. The module must be mounted in an enclosure in hazardous locations for this purpose. No wire or cable harness should exceed 30 meters in length. The power input wiring should be limited to 10 meters. All field wiring should be suitable for the operating temperature range. Install the unit with appropriate space available for servicing and for adequate wire harness access (6 inches or 15 cm) and strain relief (12 inches or 30 cm).
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.



3D VIEW
12 Pin Receptacle

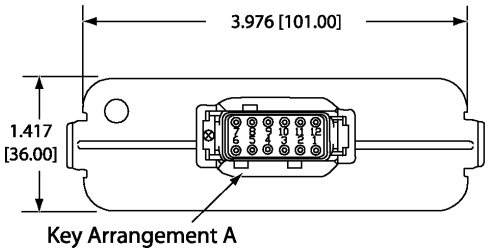
HOUSING DIMENSIONS

Housing Material: High Temperature Nylon (Black)
Protection Rating: IP67



BOTTOM VIEW

FRONT VIEW (NOT TO SCALE)
12 PIN RECEPTACLE



Mating Plug Assemblies:
12 pin receptacle - DTM06-12SA
with wedgelock WM12S and contacts
24 pin receptacle - DTM06-12SA and DTM06-12SB
with wedgelocks WM12S and contacts
Contact factory for contact specification.

Dimensions: inches [mm]
excluding mating plug(s)

<table>
<tr>
<td>Electrical Connections</td>
<td>Deutsch DTM series 12 pin receptacle (P/N: DTM13-12PA-R008)

20 AWG wire is recommended for use with contacts 0462-201-20141.

<table>
<tr><th>PIN #</th><th>FUNCTION</th></tr>
<tr><td>1</td><td>Analog GND</td></tr>
<tr><td>12</td><td>Analog GND</td></tr>
<tr><td>2</td><td>Output 1+</td></tr>
<tr><td>11</td><td>CAN_L</td></tr>
<tr><td>3</td><td>Output 2+</td></tr>
<tr><td>10</td><td>CAN_H</td></tr>
<tr><td>4</td><td>Output 3+</td></tr>
<tr><td>9</td><td>CAN_Shield</td></tr>
<tr><td>5</td><td>Output 4+</td></tr>
<tr><td>8</td><td>Power -</td></tr>
<tr><td>6</td><td>Analog GND</td></tr>
<tr><td>7</td><td>Power +</td></tr>
</table>
</td>
<td>Mating plug KIT:
Available from Axiomatic as p/n: PL-DTM06-12SA.
It is comprised of the following Deutsch IPD parts: plug (DTM06-12SA); wedgelock (WM12S); and 12 contacts (0462-201-20141) as well as 6 sealing plugs (0413-204-2005). If not all of the outputs are required for the application, use the sealing plugs to fill the mating connector pins.

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.</td>
</tr>
</table>

9 VERSION HISTORY

User Manual Version	Firmware version	Electronic Assistant™ (EA) version	Date	Author	Modifications
1A	1.xx	3.0.27.0 or higher	Aug 5, 2009	Olek Bogush	Initial release – draft version.
1B	1.xx	3.0.27.0 or higher	Oct 19, 2009	Olek Bogush	In the Binary Function function block the range of scale values was changed from [-1;1] to “Any value” for EA version 3.0.29.0 or later.
1C	1.xx	3.0.27.0 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.
1D	1.xx	3.0.27.0 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 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.
--	1.xx	3.0.27.0 or higher	--	A. Wilkins	• Added Technical specifications, installation instructions
1E	1.xx	3.0.27.0 or higher	Oct 14, 2010	Olek Bogush	• Explained discrete logical inputs in the Controller Architecture section. • Corrected Introduction section.
-	1.xx	3.0.27.0 or higher	Sept 29, 2011	Amanda Wilkins	• Added response time to Tech. Specs
2	2.xx	EA 4.4.42.0 or higher	Dec 12, 2012	Olek Bogush	• Added support for controllers with different baud rate. Changed the front page of the manual, Technical Specification section, and applied appropriate changes throughout the manual. • Added support for the J1939

					bootloader. Renamed Firmware Flashing to Flashing New Firmware section and rewrote it. • Controller Description section was rewritten. It contains now Hardware Block Diagram and Software Organization subsections. • In the Network Support section changed description of how the Identity Number is calculated in the V2.xx of the firmware. Modified the slew rate control description to address multiple baud rates. • Changed “functional blocks” to “function blocks” throughout the text. • Renamed Revision History section to Version History and changed the look of the revision history table. • Moved Installation Instructions section after Technical Specifications section. • Modified Setpoint Programming section. Electronic Assistance Software subsection was rewritten.
--	--	--	July 3, 2015	Amanda Wilkins	• Added vibration compliance information.

OUR PRODUCTS

Battery Chargers

CAN bus Controls

Current Converters

DC/DC Power Converters

DC Voltage/Current Signal Converters

Engine Temperature Scanners

Fan Drive Controllers

CAN Gateways

Hydraulic Valve Controllers

I/O Controls

LVDT Simulators

Machine Control Systems

Motor Controls

PID Controls

Position Sensors, Angle Measurement Inclinometers

Power Supplies

PWM Signal Converters/Isolators

Resolver Signal Conditioners

Service Tools

Signal Conditioners

Strain Gauge CAN Controls

Surge Suppressors

OUR COMPANY

Axiomatic provides electronic machine controls, components, and systems to the off-highway, commercial vehicle, electric vehicle, military, power generator set, material handling and industrial OEM markets.

We provide efficient, innovative solutions that focus on adding value for our customers.

We emphasize service and partnership with our customers, suppliers, and employees to build long term relationships and mutual trust.

QUALITY DESIGN AND MANUFACTURING

Axiomatic is an ISO 9001:2008 registered facility.

SERVICE

All products to be returned to Axiomatic require a Return Materials Authorization Number (RMA#).

Please provide the following information when requesting an RMA number:

- Serial number, part number
- Axiomatic invoice number and date
- Hours of operation, description of problem
- Wiring set up diagram, application
- Other comments as needed

When preparing the return shipping paperwork, please note the following. The commercial invoice for customs (and packing slip) should state the harmonized international HS (tariff code), valuation and return goods terminology, as shown in italics below. The value of the units on the commercial invoice should be identical to their purchase price.

Goods Made In Canada (or Finland)
Returned Goods for Warranty Evaluation, HS: 9813.00
Valuation Identical Goods
Axiomatic RMA#

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 as described on www.axiomatic.com/service.html.

CONTACTS

Axiomatic Technologies Corporation
5915 Wallace Street
Mississauga, ON
CANADA L4Z 1Z8
TEL: +1 905 602 9270
FAX: +1 905 602 9279
www.axiomatic.com

Axiomatic Technologies Oy
Höytämöntie 6
33880 Lempäälä
FINLAND
TEL: +358 3 3595 600
FAX: +358 3 3595 660
www.axiomatic.fi