User Manual UMAX030530
Version 3E
Firmware 3.xx
Axiomatic EA: 3.0.33.3 +

USER MANUAL

1 Analog Signal Input CAN Controller

P/N: AX030530

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

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

ACRONYMS

CAN	Controller Area Network
CANopen®	CAN-based higher layer protocol supported by CAN in Automation (CiA)
DM	Diagnostic message. Defined in J1939/73 standard
EA	The Axiomatic Electronic Assistant, P/Ns: AX070502 or AX070506K. The EA is a 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
ISO	International Organization for Standardization
LSB	Less Significant Byte
MAP	Memory Access Protocol. Defined in J1939/73 standard
OSI	Open System Interconnection
PC	Personal Computer
PGN	Parameter Group Number. Defined in J1939 standard
PID	Proportional–integral–derivative (regulator)
PWM	Pulse-width modulation
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 Input CAN Controller. The manual is intended to provide users with all necessary information for programming of custom solutions on the base of this controller.

The user should check whether the application firmware installed in the controller 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 V3.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 convert a physical signal from its universal input into one or several J1939 CAN signals transmitted on the CAN bus. The universal input accepts: voltage, current, resistance, frequency, PWM duty cycle, and discrete levels.

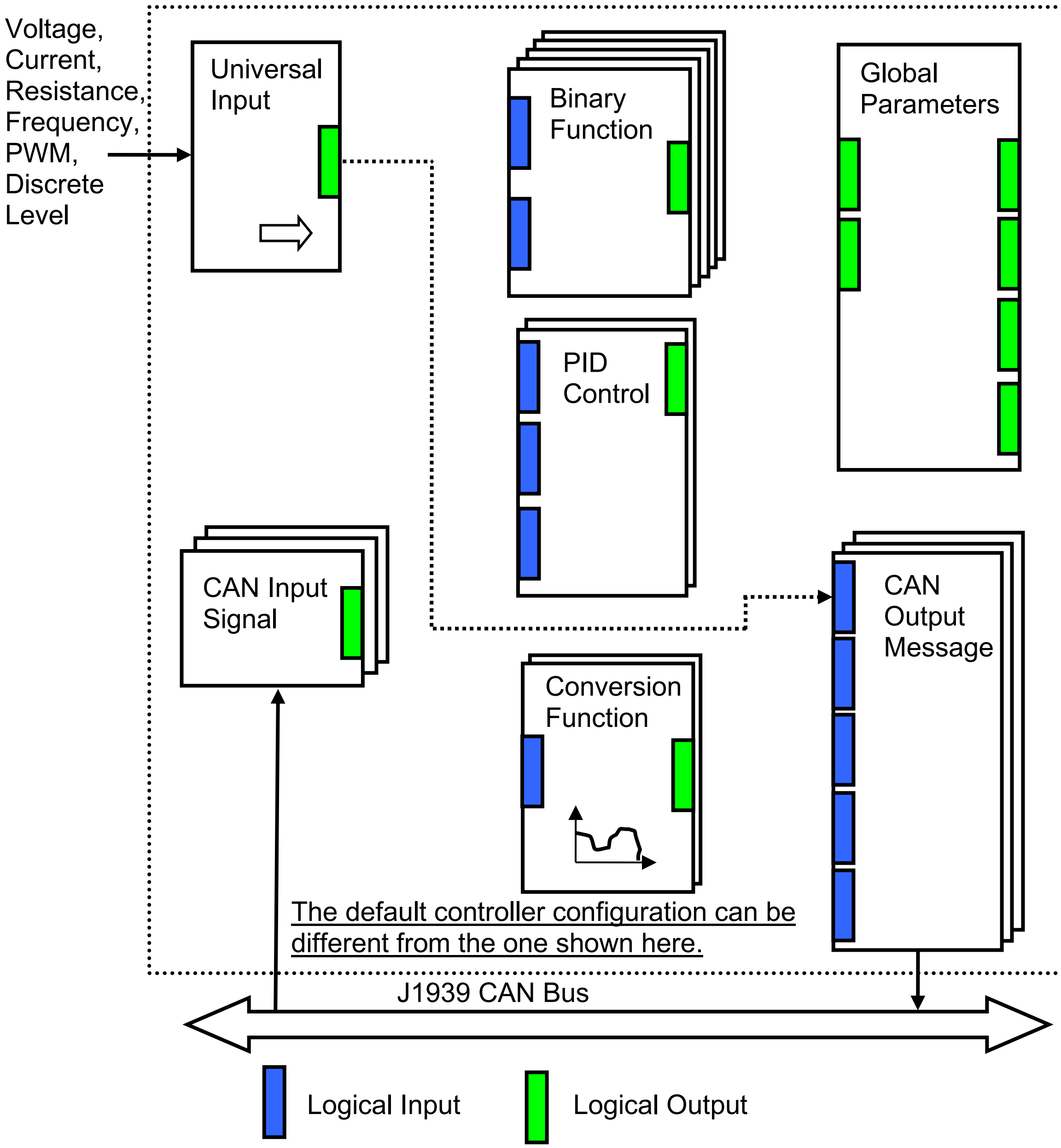
The 1 Analog Signal Input CAN Controller belongs to a family of Axiomatic user-customizable smart controllers. The programmable internal architecture provides users with an ultimate flexibility, allowing them to build their own custom controller with a required functionality from a set of predefined internal functional 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 control signals transmitted on the CAN bus, 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.

3 CONTROLLER ARCHITECTURE

From the software prospective, the controller consists of a set of internal functional 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 Universal Input functional block is connected to the logical input of the CAN Output Message functional block, providing a direct path for the input signal to the controller CAN output.

Figure 1.The Controller Internal Structure

Each functional 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 functional blocks. One type represents the controller hardware resources, for example the Universal Input functional block. The other type is purely logical – these functional blocks are included to program the user defined functionality of the controller. The number and functional diversity of these functional 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 functional 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 functional 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:

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

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 functional 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 functional 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 functional blocks, logical input signals can be inverted using the following inversion function:

$$Inv(Xn,I),\ I \in \{Yes,No\},$$

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

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.

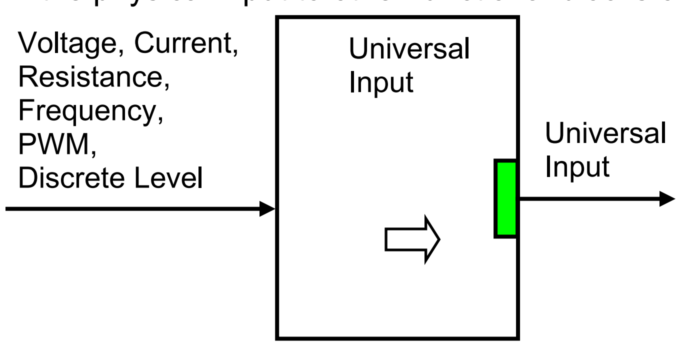
When the data source does not carry a valid data, the output signal value is always set to 0 and the inversion operation on the signal in suppressed. In this case, instead of the signal value, the logical signal carries a signal state code, associated with its signal state, see the table below:

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 functional blocks to report that a CAN input signal is absent on the bus, is out of range, etc. Other functional blocks usually use only the "Error" state to show an error condition.

3.1 Universal Input

The Universal Input functional block has one logical output providing a normalized input signal from the physical input to other functional blocks of the controller.



The functional block setpoints are presented in the following table:

Name	Default Value	Range	Units	Description
Input Parameter	Voltage	{Input Disabled, Voltage, Current, Resistance,	–	Type of the universal input electrical parameter to be

Name	Default Value	Range	Units	Description
		Discrete Voltage Level, Frequency, PWM Duty Cycle}		measured
Voltage Range	Auto-range	{Auto-range, 0…10V, 0…5V, 0…2.5V, 0…1V}	–	Signal range for Voltage measurements
Current Range	Auto-range	{Auto-range, 0…20mA, 4…20mA}	–	Signal range for Current measurements
Resistance Range	Auto Range	{ Auto Range, 0…150Ohm, 0…800Ohm, 0…2.5kOhm, 0…8kOhm, 0…25kOhm, 0…80kOhm, 0…250kOhm}	–	Signal range for Resistance measurements
Frequency Range	10Hz…1kHz	{10Hz…1kHz, 100Hz…10kHz}	–	Signal frequency range for Frequency and PWM Duty Cycle measurements
Pull-Up/Pull-Down Resistor	Disabled	{Disabled, 10kOhm Pull-Up, 10kOhm Pull-Down}	–	Connection of the pull-up/pull-down resistor for: Discrete Voltage Level, Frequency and PWM Duty Cycle measurements
Analog Input Filter	Both: 50Hz and 60Hz noise rejection	{Disabled, 50Hz noise rejection, 60Hz noise rejection, Both: 50Hz and 60Hz noise rejection}	–	Input filter for: Voltage, Current and Resistance measurements
Debounce Input Filter	1.78µs	{Disabled, 111ns, 1.78µs, 14.22µs}	–	Debounce input digital filter for Frequency and PWM Duty Cycle measurements
Digital Input Polarity	Active High	{Active High, Active Low}	–	Input polarity for Discrete Voltage Level and PWM Duty Cycle measurements
Vmax – Maximum Input Voltage	5.0	[0…10], but Vmax>Vmin	V	Normalization parameters for Voltage measurements
Vmin – Minimum Input Voltage	0.0	[0…10], but Vmin<Vmax	V	
Imax – Maximum Input Current	20.0	[0…20], but Imax>Imin	mA	Normalization parameters for Current measurements
Imin – Minimum Input Current	0.0	[0…20], but Imin<Imax	mA	
Rmax – Maximum Input Resistance	250.0	[0…250], but Rmax>Rmin	kOhm	Normalization parameters for Resistance measurements
Rmin – Minimum Input Resistance	0.0[1]	[0…250], but Rmin<Rmax	kOhm	
Fmax – Maximum Input Frequency	1000.0	[0…10000], but Fmax>Fmin	Hz	Normalization parameters for Frequency measurements
Fmin – Minimum Input Frequency	0.0[2]	[0…10000], but Fmin<Fmax	Hz	
Dmax – Maximum Duty Cycle	100.0	[0…100], but Dmax>Dmin	%	Normalization parameters for PWM Duty Cycle measurements
Dmin – Minimum Duty Cycle	0.0	[0…100], but Dmin<Dmax	%	

[1] Resistance bellow 20 Ohm is measured as 0 Ohm, when the Resistance Range is set to Auto Range or 0…150Ohm range.
[2] Frequencies bellow 9.5Hz for 10Hz…1kHz range (95Hz for 100Hz…10kHz range) are measured as 0 Hz.

Signal ranges should comply with the normalization parameters, unless the Auto-range is selected. Setting, for example, voltage range to 0…1V and Vmin=5V, Vmax=10V will result in the logical output being equal to 0.0 independently of the input voltage. In the Auto-range mode, the correct range is selected automatically based on the normalization parameters for voltage and current measurements or the actual resistance value for resistance measurements. For the previous example, with Vmin=5V and Vmax=10V, the voltage range will be automatically set to 0…10V.

Please note, that the Auto-range is not available for the Frequency and PWM Duty Cycle measurements. The user should choose a correct frequency range independently of the normalization parameters. If the frequency of the input signal is beyond the specified frequency range, the output signal of the functional block will be in the Error state.

3.1.1 Voltage Input

To acquire a voltage signal, the user should set: Input Parameter – to Voltage, Voltage Range – to the expected signal range or Auto-range, Vmin and Vmax – to the minimum and maximum voltage acquired by the functional block.

Usually, Vmin and Vmax are set to cover the entire signal range. For example, for Voltage Range equal to 0…5V: Vmin=0 [V] and Vmax=5 [V]. For some applications, however, they can be set inside the signal range. For example, if there is a +5V potentiometer input, setting Vmin=0.1[V] and Vmax=4.9 [V] will ensure that the minimum and maximum potentiometer positions will be clearly identified.

The voltage signal, as well as all other analog signals, is sampled every 1.1(1) ms. By default, it is filtered by a running average filter, which is adjusted using the Analog Input Filter setpoint. The parameters of the filter are provided below:

Analog Input Filter	Number of points	Averaging Period [ms]
Disabled	-	-
50Hz noise rejection	18	20
60Hz noise rejection	15	16.6(6)
Both: 50Hz and 60Hz noise rejection	90	100

3.1.2 Current Input

The current signal is acquired the same way as a voltage signal. The user should set: Input Parameter – to Current, Current Range – to the expected current signal range or Auto-range, Imin and Imax – to the minimum and maximum current that will be output as a logical signal by the functional block.

The user should also define the filter type using the Analog Input Filter setpoint.

Please, remember that the unit acquires current by measuring a voltage drop on an internal 124Ohm reference resistor. The value of this resistor should be within the acceptable range for the current source.

3.1.3 Resistance Input

The Universal Input functional block can be set to measure resistance by setting the Input Parameter setpoint to Resistance, Resistance Range – to Auto Range or a specific range, and Rmin, Rmax normalization parameters – to the required resistance range.

Analog input filter is also used for resistance measurements. It is recommended that the Analog Input Filter setpoint be set to the value rejecting both: 50Hz and 60Hz industrial noise.
When the Resistance Range setpoint is set to Auto Range, a special algorithm is used to dynamically switch between resistance ranges to ensure that the resistance value is measured with the best accuracy and resolution, see the table below:

Resistance Range	Resistance Value in the Auto Range Mode	Switching to the Lower Resistance Range	Switching to the Higher Resistance Range	Initial Resistance Value
0…150 Ohm	>150 Ohm	-	150 Ohm	< 150 Ohm
0…800 Ohm	100 Ohm …800 Ohm	100 Ohm	800 Ohm	150 Ohm …800 Ohm
0…2.5 kOhm	500 Ohm …2.5 kOhm	500 Ohm	2.5 kOhm	800 Ohm… 2.5 kOhm
0…8 kOhm	1.5 kOhm …8 kOhm	1.5 kOhm	8 kOhm	2.5 kOhm…8 kOhm
0…25 kOhm	5 kOhm …25 kOhm	5 kOhm	25 kOhm	8 kOhm… 25 kOhm
0…80 kOhm	15 kOhm…80 kOhm	15 kOhm	80 kOhm	25 kOhm…80 kOhm
0…250 kOhm	>50 kOhm	50 kOhm	-	> 80 kOhm

When the measurements start, the resistance range is first determined based on the initial resistance value. Then, the resistance range is adjusted according to the results of the subsequent measurements. The ranges overlap to avoid frequent switching in the vicinity of the range ends.

Switching from one resistance range to another takes some time. It should be taken into consideration, since the logical output update of the Universal Input functional block is suppressed during this time to avoid transients in the output data.

The switching time can be estimated as a sum of a resistance range switching delay, which is equal to 150ms, and an averaging period of the Analog Input Filter. For the default 50Hz and 60Hz noise rejection setting of the Analog Input Filter, the switching time is: 150ms+100ms=250ms.

When switching between resistance ranges is not desirable, the user can use one of the predefined resistance ranges, which includes Rmin and Rmax values.

3.1.4 Frequency and PWM Input

The user can set the Universal Input to measure frequency or PWM input signal using the Input Parameter setpoint. The user should define the frequency range of the input signal by the Frequency Range setpoint and set the Fmin, Fmax or Dmin, Dmax normalization parameters.

The polarity of the input signal is set by the Digital Input Polarity setpoint. The user can also apply a pull-up or pull-down resistor by the Pull-Up/Pull-Down Resistor setpoint and change the debounce input filter settings using the Debounce Input Filter setpoint to filter out parasitic spikes that can be present in the noisy input signal.

Be aware, that the debounce filter settings can affect accuracy of the frequency and PWM signal acquisition at high frequencies. For example, for the 10 kHz PWM signal, setting the Debounce Input Filter to 14.22µs will result in 14.22% additional error in the output data.

For the Frequency and PWM Duty Cycle input modes the Universal Input functional block will output an error code if the frequency of the input signal is beyond the selected frequency range. The signal value, in this case, will be 0.

Frequency Range	**Input Frequency**	**Error Code**
10Hz…1kHz	< 9.155 Hz	0
	≥ 1.2 kHz	1
100Hz…10kHz	< 91.55 Hz	0
	≥ 12 kHz	1

This error code can be acquired through the CAN bus when the logical output of the Universal Input is connected to the CAN Output Message functional block.

For the Duty Cycle measurements, a special algorithm will identify a loss of the PWM frequency carrier as 0% or 100% valid PWM signal depending on the Digital Input Polarity setpoint and the actual digital state of the input.

3.1.5 Discrete Voltage Level Input

The discrete voltage level input mode is the simplest mode of the Universal Input functional block. It is intended to input control signals mainly from switches and buttons.

To activate this mode the user should set the Input Parameter setpoint to the Discrete Voltage Level and define the polarity of the input signal by the Digital Input Polarity setpoint.

The user can also apply a pull-up or pull-down resistor by the Pull-Up/Pull-Down Resistor setpoint.

The debouncing time for the input signal in this mode is fixed and set to 100ms.

3.2 Conversion Function

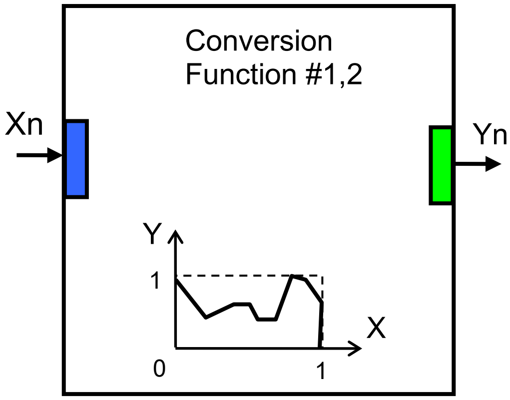
The Conversion Function functional block allows the user to perform a linearization of an input signal, apply a user-defined control profile, and to do a hotshot control, if necessary.

There are two Conversion Function blocks available in the current version of the controller. Each function block one logical input, one output and implements a function:

$$Yn = F(Xn),$$

where:

Xn – normalized input signal (can be inverted by the inversion function), Yn – normalized output signal.



The function F(x) is defined using a piecewise linear approximation in up to 11 points. Each point is presented by three parameters:

$$P_i = (State_i, Xn_i, Yn_i),\ i = 0\ldots 10,$$

where: P_i – i-th point of the function F,
$State_i$ – state of the i-th point. $State_i \in$ {Off, On},
Xn_i – normalized input value at the i-th point.
Yn_i – normalized output value at the i-th point.
If the $State_i$=Off, the point is not active and is not used in the function approximation.

The function values between active points (with $State_i$=On) are defined the following way:

$$Yn = A_j \cdot Xn + B_j\ ,\ j = 0\ldots N,\ N \leq 10,$$

$$A_j = (Yn_j - Yn_{(j+1)}) / (Xn_j - Xn_{(j+1)})\ ,$$

$$B_j = (Yn_{(j+1)} \cdot Xn_j - Yn_j \cdot Xn_{(j+1)}) / (Xn_j - Xn_{(j+1)})\ ,$$

$$Xn \in [Xn_j\ ;\ Xn_{(j+1)}[,\ State_j=On,\ State_{(j+1)}=On.$$

where: A_j, B_j – linear approximation coefficients between j and (j+1) active points.
N – number of active points.

The Conversion Function functional block is also capable to implement a hotshot control. For this purpose the user can specify two values for the last, 10-th, function point. The first value is a normalized output value at the 10-th point and the second one is the value that will be assigned to the output if the input remains Xn =1.0 for a hotshot time.

The Conversion Function functional block has the following set of setpoints:

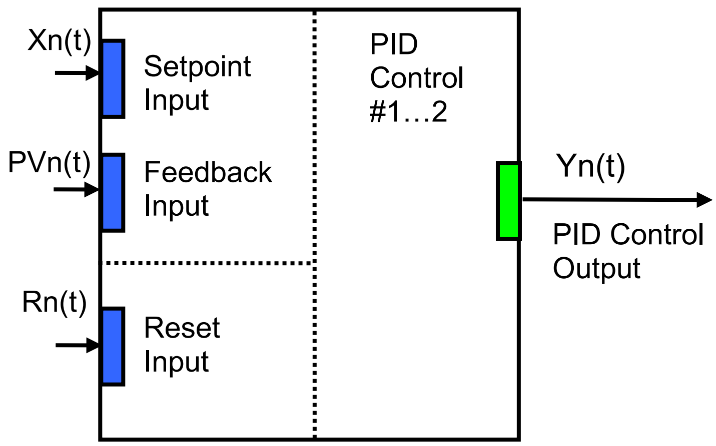
Name	Default Value	Range	Units	Description
Input Source	Not Connected	Any logical output of any functional block or "Not Connected"	–	Defines a source of the input signal Xn
Input Inversion	No	{Yes, No}	–	Specifies, whether the input signal Xn is inverted
Point 0 State	On	–	–	$State_0$. Read only parameter
Point 0 X	0	–	–	Xn_0. Read only parameter

Name	Default Value	Range	Units	Description
Point 0 Y	0	[0;1]	–	Yn_0
Point 1 State	Off	{Off, On}	–	$State_1$
Point 1 X	0.1	[Xn_0; Xn_2]	–	Xn_1
Point 1 Y	0	[0;1]	–	Yn_1
Point 2 State	Off	{Off, On}	–	$State_2$
Point 2 X	0.2	[Xn_1; Xn_3]	–	Xn_2
Point 2 Y	0	[0;1]	–	Yn_2
Point 3 State	Off	{Off, On}	–	$State_3$
Point 3 X	0.3	[Xn_2; Xn_4]	–	Xn_3
Point 3 Y	0	[0;1]	–	Yn_3
Point 4 State	Off	{Off, On}	–	$State_4$
Point 4 X	0.4	[Xn_3; Xn_5]	–	Xn_4
Point 4 Y	0	[0;1]	–	Yn_4
Point 5 State	Off	{Off, On}	–	$State_5$
Point 5 X	0.5	[Xn_4; Xn_6]	–	Xn_5
Point 5 Y	0	[0;1]	–	Yn_5
Point 6 State	Off	{Off, On}	–	$State_6$
Point 6 X	0.6	[Xn_5; Xn_7]	–	Xn_6
Point 6 Y	0	[0;1]	–	Yn_6
Point 7 State	Off	{Off, On}	–	$State_7$
Point 7 X	0.7	[Xn_6; Xn_8]	–	Xn_7
Point 7 Y	0	[0;1]	–	Yn_7
Point 8 State	Off	{Off, On}	–	$State_8$
Point 8 X	0.8	[Xn_7; Xn_9]	–	Xn_8
Point 8 Y	0	[0;1]	–	Yn_8
Point 9 State	Off	{Off, On}	–	$State_9$
Point 9 X	0.9	[Xn_8; Xn_{10}]	–	Xn_9
Point 9 Y	0	[0;1]	–	Yn_9
Point 10 State	On	–	–	$State_{10}$. Read only parameter
Point 10 X	1	–	–	Xn_{10}. Read only parameter
Point 10 Y	0	[0;1]	–	Yn_{10}
Hotshot Delay	0	0…10000	ms	Undefined if 0
Hotshot Y	0	[0;1]	–	Yn_{10}, if Xn=1.0 for Time>Hotshot Delay, and Hotshot Delay $\neq$ 0

3.3 PID Control

To provide the user with means to build generic closed loop PID regulators, two PID Control functional blocks were added to the controller.

A PID Control functional block has: setpoint and feedback inputs, manual control mode and a reset input to bring the regulator into its initial state. The user can also adjust the time resolution for fast or slow responding closed loop systems.



The normalized output of the PID Control functional block Yn(t), as a function of time, can be described by the following formula:

$$Yn(t)=Clip(Y(t)),$$

$$Y(t) = P \cdot [\ e(t) + 1/T_I \cdot \int e(t)dt - T_D \cdot dPVn(t)/dt\ ],$$

where:

Clip(Y(t))={Y(t), if $0 \le Y(t) \le 1$; 0, if $Y(t)<0$; 1, if $Y(t)>1$} – clipping function;
e(t) = Xn(t) – PVn(t) – error function, where
Xn(t) – normalized setpoint variable, set by the Setpoint Input,
PVn(t) – normalized process variable, set by the Feedback Input,
P – proportional gain,
T_I – integral time,
T_D – derivative time.

All PID Control logical inputs can be inverted.

To avoid saturation of the output due to the integral term of the PID regulator, an anti-windup algorithm is implemented. The integrator is stopped when the output saturates and the error function moves the output to further saturation:

- $Y(t)>1$ and $e(t)>0$ or
- $Y(t)<0$ and $e(t)<0$.

When the Reset Input is activated, the integral part of the PID regulator is reset to zero and the output of the PID Control functional block is brought to zero, too:

$$\int e(t)dt=0,\ Y(t) = 0, \quad \text{when } Rn(t) \ge 0.5,$$

where:

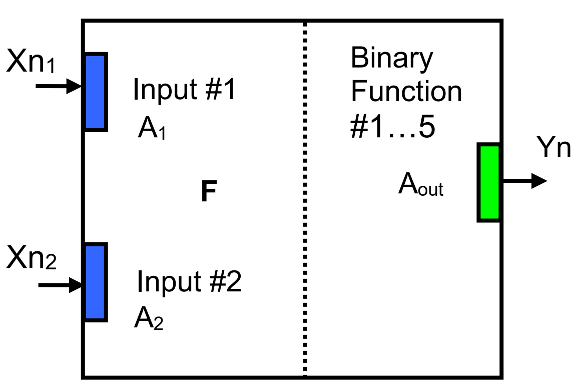
Rn(t) – normalized reset variable, set by the Reset Input.

Setpoints of the PID Control functional block are presented in the following table:

Name	Default Value	Range	Units	Description
Setpoint Input Source	Not Connected	Any logical output of any functional block or "Not Connected"	–	Source of a setpoint input signal
Setpoint Input Inversion	No	{Yes, No}	–	Specifies, whether to invert the setpoint signal
Feedback Input Source	Not Connected	Any logical output of any functional block or "Not Connected"	–	Source of a feedback input signal
Feedback Input Inversion	No	{Yes, No}	–	Specifies, whether to invert the feedback signal
Reset Input Source	Not Connected	Any logical output of any functional block or "Not Connected"	–	Source of a reset input signal. The signal brings the regulator into its initial state.
Reset Input Inversion	No	{Yes, No}	–	Specifies, whether to invert the reset signal
Manual Control	No	{Yes, No}	–	Put the PID control in a manual control mode. In this mode the PID regulator is off and the PID output is equal to the value of the "Manual Control Output" setpoint
Manual Control Output	0.5	[0;1]	–	Output of the PID control in the manual control mode
Proportional Gain	1.0	[0;100000]	–	Proportional PID parameter
Integral Time Constant	0.1	[0;100000]	s	Integral PID parameter
Derivative Time Constant	0.01	[0;100000]	s	Derivative PID parameter. Derivation from the process variable is used.
Time Resolution	0.001	[0.001;10]	s	Time interval between PID control cycles

3.4 Binary Function

There are five Binary Function functional blocks added to the controller to support advanced control algorithms. Each Binary Function functional 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 functional 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:

$Clip(Y)=\{Y, \text{ if } 0 \le Y \le 1; 0, \text{ if } Y<0; 1, \text{ if } Y>1\}$ – 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 functional block is not available and its signal value is equal to Yn=0.

The Binary Function functional block has the following set of setpoints:

Name	Default Value	Range	Units	Description
Input #1 Source	Not Connected	Any logical output of any functional 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 functional 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 $X_i \ge 0.5$ (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:

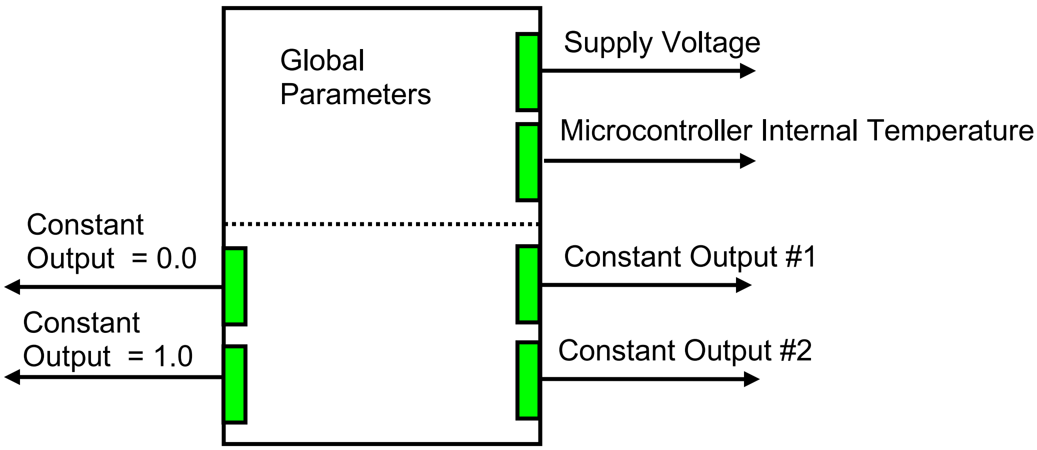
$F^{(\le)}[x,y] = \{1, \text{ if } x \le y+\delta;\ 0, \text{ if } x > y+\delta \}$,

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

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

3.5 Global Parameters

The Global Parameters functional 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 functional blocks as constant input sources. For example, they can be used to set up threshold values in Binary Function functional 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 functional 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.

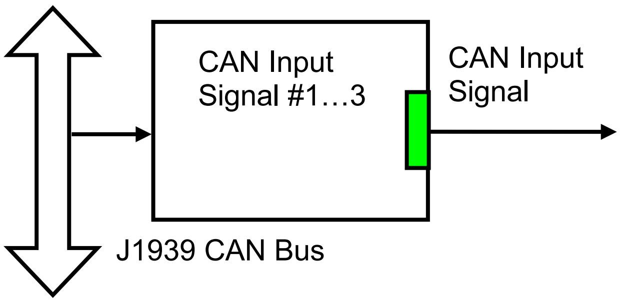
The setpoints for the Global Parameters functional block are presented in the following table:

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	100	–	V	Normalization parameters for the inclinometer supply voltage. Read only parameters.
Vsmin – Min Supply Voltage	0	–	V	
Tmax – Max Microcontroller	150	–	°C	Normalization parameters for the microcontroller embedded

Name	Default Value	Range	Units	Description
Temperature				temperature sensor. Read only parameters.
Tmin – Min Microcontroller Temperature	-50	–	°C	

3.6 CAN Input Signals

There are three CAN Input Signal functional blocks supported by the controller. Each functional 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 functional block then outputs the signal data to its logical output for processing by other functional blocks of the controller.



The CAN Input Signal functional 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 functional 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 functional 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

Name	Default Value	Range	Units	Description
				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.

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	0…250 (0…0xFA)	Valid Data	[0;1] - normalized	0

CAN Signal Type	CAN Signal Code	Logical Signal		
		State	Value	Signal State Code
Continuous			signal code	
	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…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.

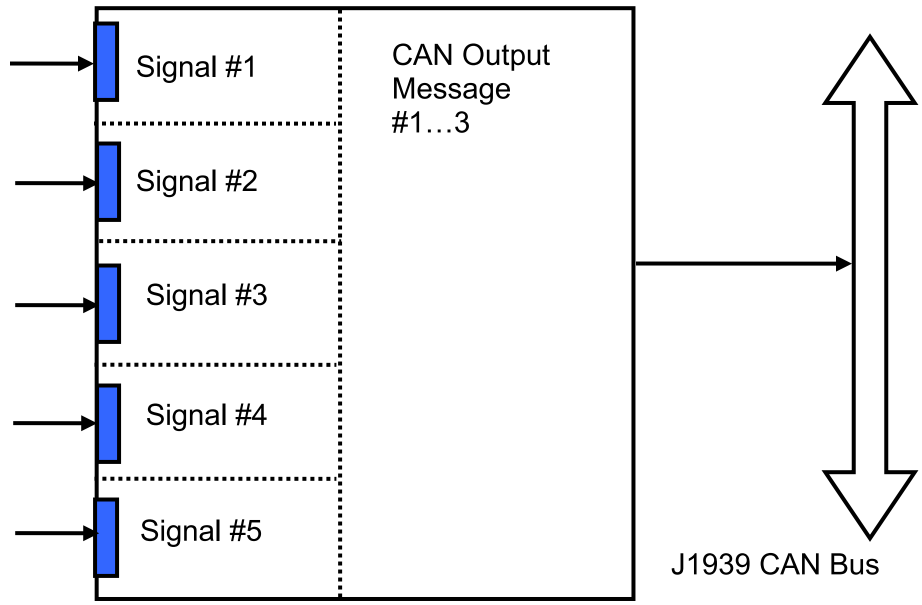
The J1939 standard does not specify how to encode the error codes and parameter specific indicators within the special indicator ranges. The control uses its own simple way of encoding, converting parameter specific and error indicators into absolute signal state codes. This allows to receive and transmit 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 functional block for reverse conversion of the logical signals into the CAN signal codes.

3.7 CAN Output Message

There are three CAN Output Message functional 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 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 functional 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	100	[0;10000]		CAN output message transmission rate. If 0 – transmission is upon request
Signal #1 Type	1-Byte Continuous	{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	Universal Input #1	Any logical output of any functional 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

Name	Default Value	Range	Units	Description
				Byte" for discrete output signals[1]
Signal #1 Resolution	0.02	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	5.0	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 functional 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	–	Type of the CAN output signal #3

Name	Default Value	Range	Units	Description
		Continuous, 4-Byte Continuous}		
Signal #3 Source	Not Connected	Any logical output of any functional 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 functional 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

Name	Default Value	Range	Units	Description
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, 4-Byte Continuous}	–	Type of the CAN output signal #5
Signal #5 Source	Not Connected	Any logical output of any functional 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 functional 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 (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	4211081216… 4261412863

CAN Signal Type	Logical Signal			CAN Signal Code
	State	Value	Signal State Code	
			(0…0xFFFFFFFF)	(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) 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 the Axiomatic 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 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	21 (Twenty second 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 Unit Serial Number	No[1]

[1]Programmed through the RS232 service interface in production

The user can change the controller ECU instance using the Axiomatic 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 156, 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 the Axiomatic EA, if necessary.

4.2 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.

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 Axiomatic EA programmable setpoints which control 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	156	[0…253]		ECU Address
Slew Rate	Slow	{Slow, Fast}	–	Slew rate control of the CAN transceiver

5 SETPOINT PROGRAMMING

The controller setpoints can be viewed and programmed using the standard J1939 memory access protocol through the CAN bus. Axiomatic provides its own PC-based Electronic Assistant (EA) software, together with a USB-CAN converter, to accommodate this task. Please refer to the Axiomatic EA User Manual for a detailed description of the Axiomatic EA's functionality, and for network connection troubleshooting.

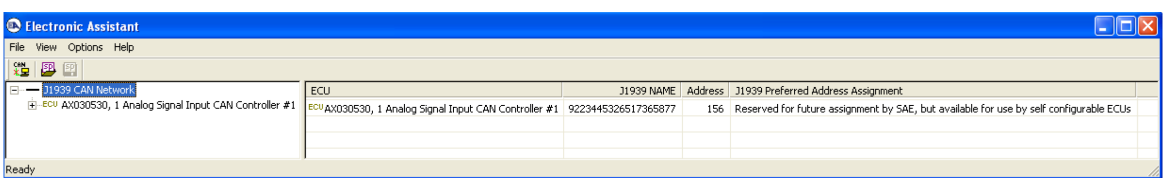
5.1 Axiomatic EA Software

Axiomatic provides its own PC-based Electronic Assistant (EA) software, together with a USB-CAN converter, as a KIT, P/Ns: AX070502 or AX070506K, to communicate with a wide range of Axiomatic products, including this converter. Please also refer to the Axiomatic EA user manual UMAX07050X for a description of the Axiomatic EA and for network connection troubleshooting.

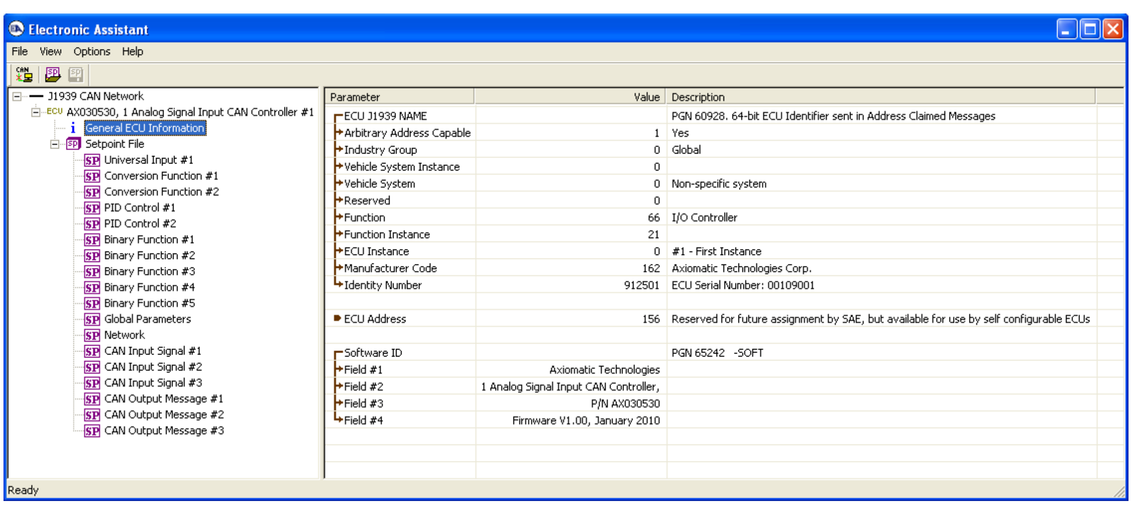
The user should use Axiomatic EA software version 3.0.33.3 or higher, which supports this converter firmware. The most recent Axiomatic EA software version can be downloaded from the Axiomatic web site, at: www.axiomatic.com.

Before connecting to the converter, the user should first check whether the baud rate in the Axiomatic EA is set to the default 250kBit/s (displayed in the bottom-right corner of the Axiomatic EA screen in EA versions starting from V4.3.41.0).

Upon connection, the Axiomatic 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:



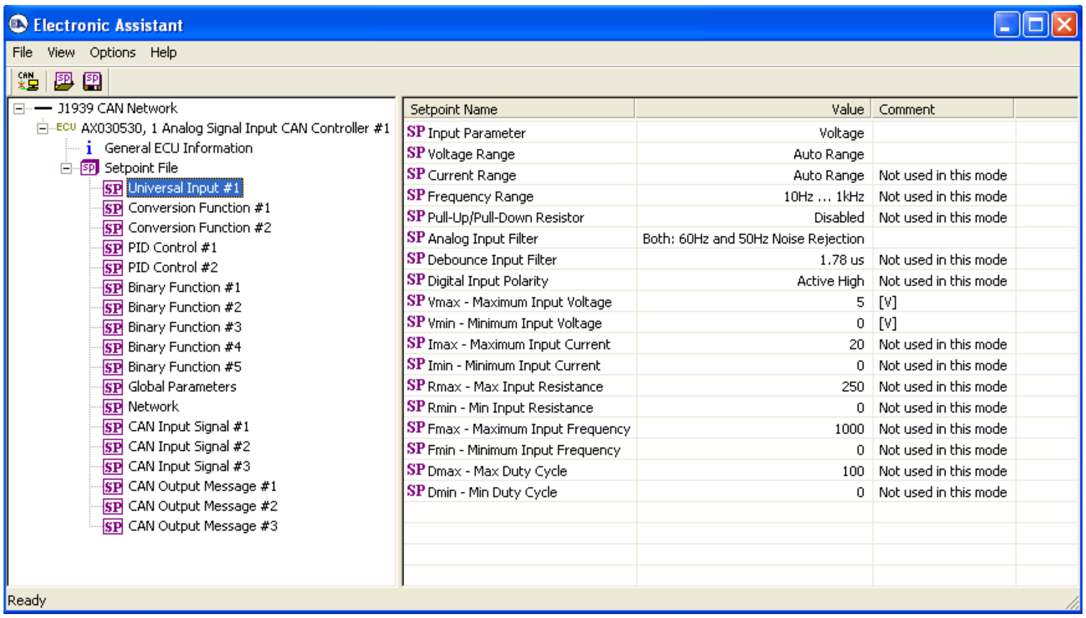
The user then can open the General ECU Information folder in the left pane to check the ECU information including the controller firmware version.



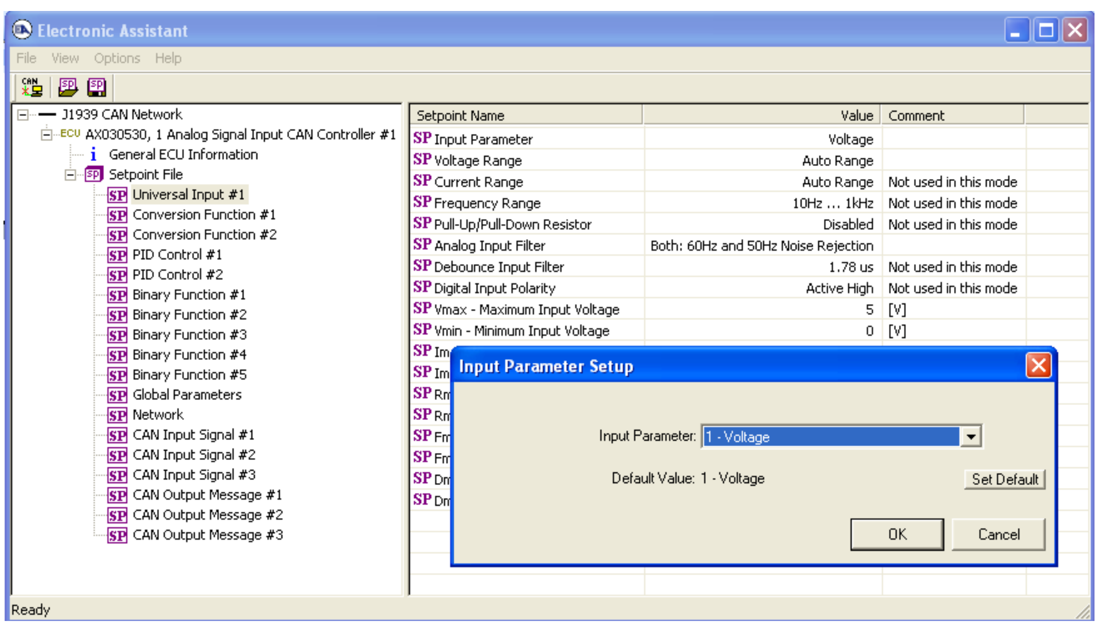
The user should check whether this version is supported by the manual. Otherwise, a different user manual is required to program the controller.

5.2 Controller Functional Blocks in the Axiomatic EA

Each functional block of the controller is presented by its own folder in the Setpoint File root folder. The individual setpoints of the functional blocks can be accessed through these folders:



The user can view and, when necessary, change these setpoints by double-clicking on the appropriate setpoint name activating the setpoint editing dialog box:



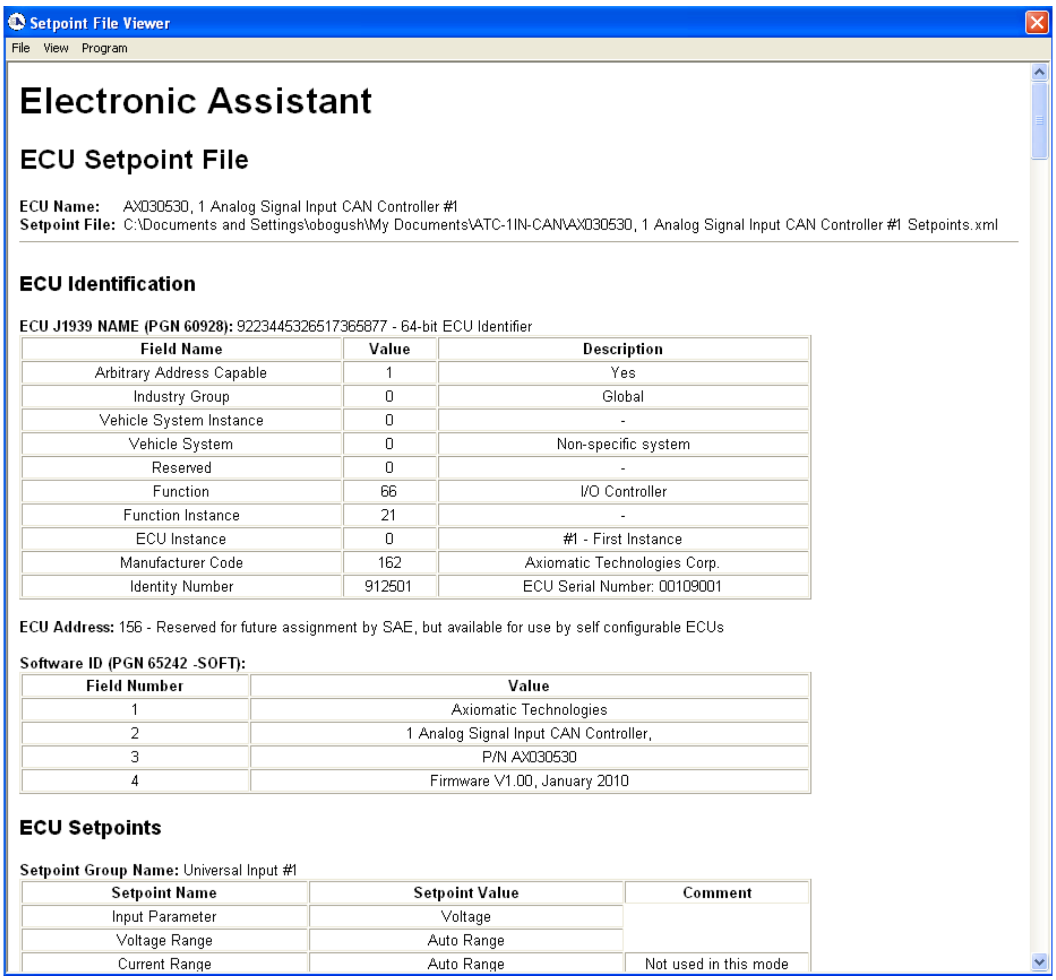
The controller will perform an internal reset of all functional blocks after each change of the setpoints. If the new setpoint affects the network identification, the controller will reclaim its network address with a new network identification message, see J1939 Name and Address.

All controller functional blocks are described in the appropriate 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 Axiomatic EA can store all controller setpoints in one setpoint file and then flash them into the controller in one operation.

The setpoint file can be created and stored on disk using the command *Save Setpoint File* from the Axiomatic EA menu or toolbar. Users can then 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: AX030530, 1 Analog Signal Input CAN Controller #1
Setpoint File: C:\Documents and Settings\obogush\My Documents\ATC-1IN-CAN\AX030530, 1 Analog Signal Input CAN Controller #1 Setpoints.xml

ECU Identification

ECU J1939 NAME (PGN 60928): 9223445326517365877 - 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	21	-
ECU Instance	0	#1 - First Instance
Manufacturer Code	162	Axiomatic Technologies Corp.
Identity Number	912501	ECU Serial Number: 00109001

ECU Address: 156 - 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	1 Analog Signal Input CAN Controller,
3	P/N AX030530
4	Firmware V1.00, January 2010

ECU Setpoints

Setpoint Group Name: Universal Input #1

Setpoint Name	Setpoint Value	Comment
Input Parameter	Voltage	
Voltage Range	Auto Range	
Current Range	Auto Range	Not used in this mode

To ensure correctness of the flashing operation, a setpoint file should be transferred between controllers with the same major firmware version using the recommended Axiomatic EA version. Otherwise, a manual inspection of all setpoints is recommended after the setpoint flashing operation.

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 functional blocks during the setpoint flashing operation.

5.4 Default Setpoints

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

The default setpoint values form a default controller configuration. In this configuration, the Universal Input is set to the input voltage mode with auto-range and normalization parameters: Vmin=0V and Vmax=5V. The output of the Universal Input is connected to the Signal #1 Source input of the CAN Output Message #1 functional block (Figure 2). The transmission of the CAN output message, defined by this functional block, is disabled by default, but can be easily enabled through the *Transmission Enable* setpoint. In this case, the unit will be transmitting the input voltage every 100ms in the first byte of PGN 65281.

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

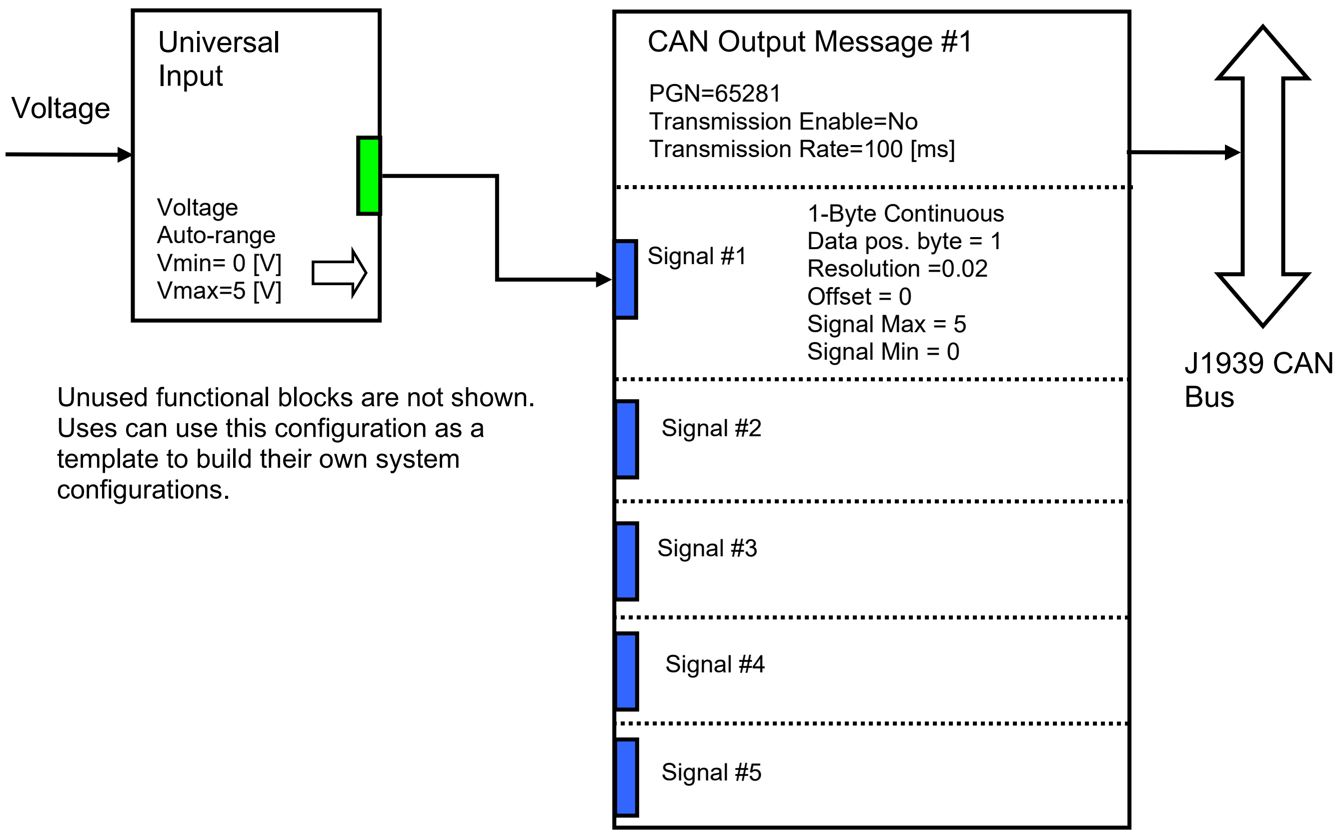


Figure 2. The Block Diagram of the Default Controller Configuration

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 bellow, as an example.

5.5.1 User Requirements

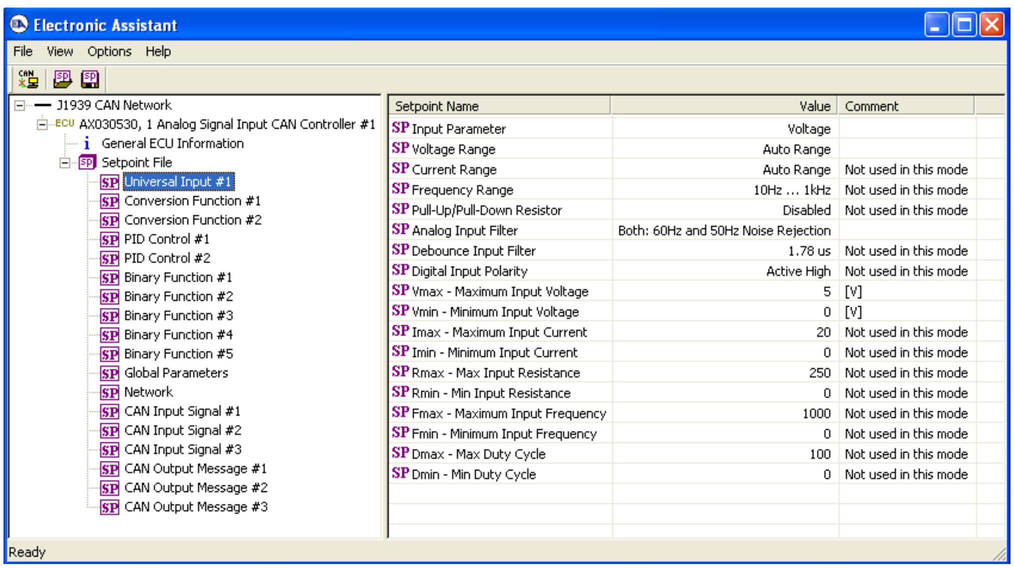
Let us assume that the controller should be programmed to output an alarm signal on the CAN bus if the input voltage from a sensor, connected to the universal input, exceeds 3V or is below 0.5V. The sensor output voltage is in a standard range from 0 to 5V.

For consistency, let the alarm signal occupy first 2 bits of the second byte in PGN 65281 and let this signal be sent every 250 ms.

5.5.2 Programming Steps

First, create a block diagram of the required controller configuration using the controller functional blocks (Figure 3).

Then, configure the controller Universal Input #1 functional block:



Set the *Input Parameter* setpoint to "Voltage", the *Vmin – Minimum Input Voltage* to 0 V, and the *Vmax – Maximum Input Voltage* to 5V. The universal input is now accepting a voltage signal from the sensor and converting it to a logical signal.

Now, configure the Binary Functions #1…3 to convert the universal input logical signal into the required alarm signal.

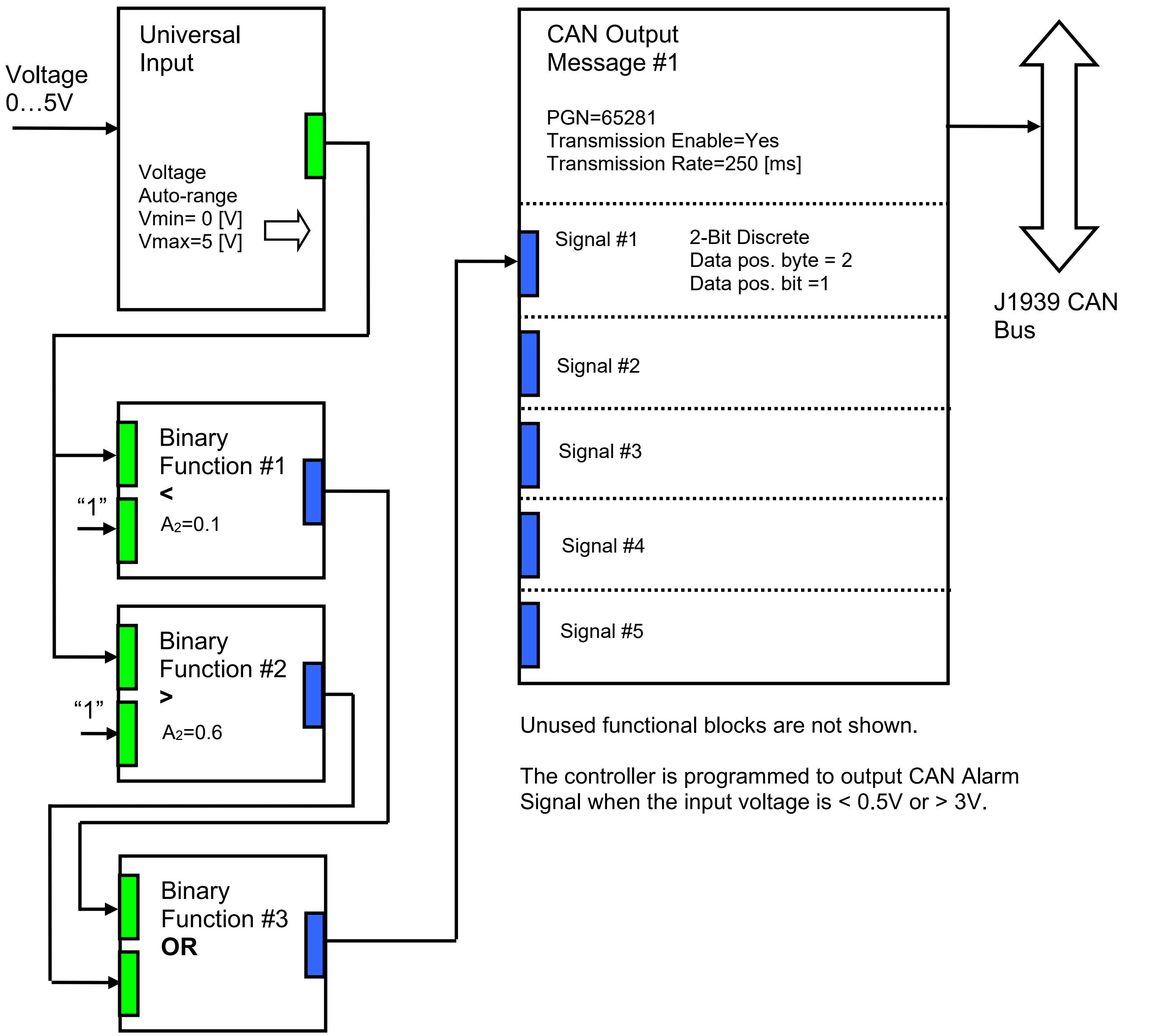
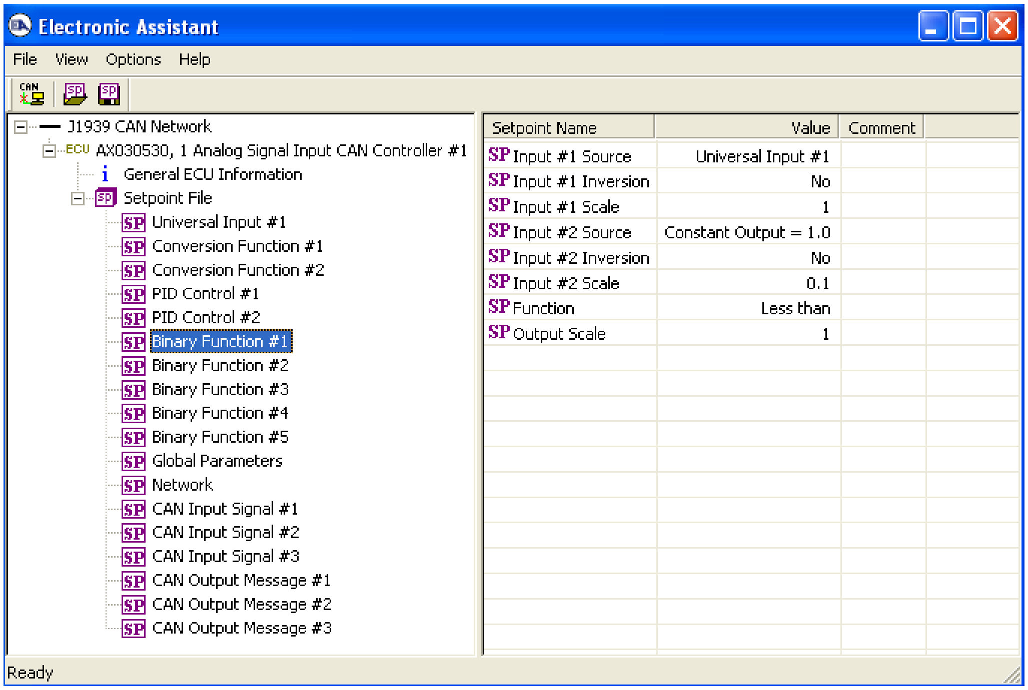
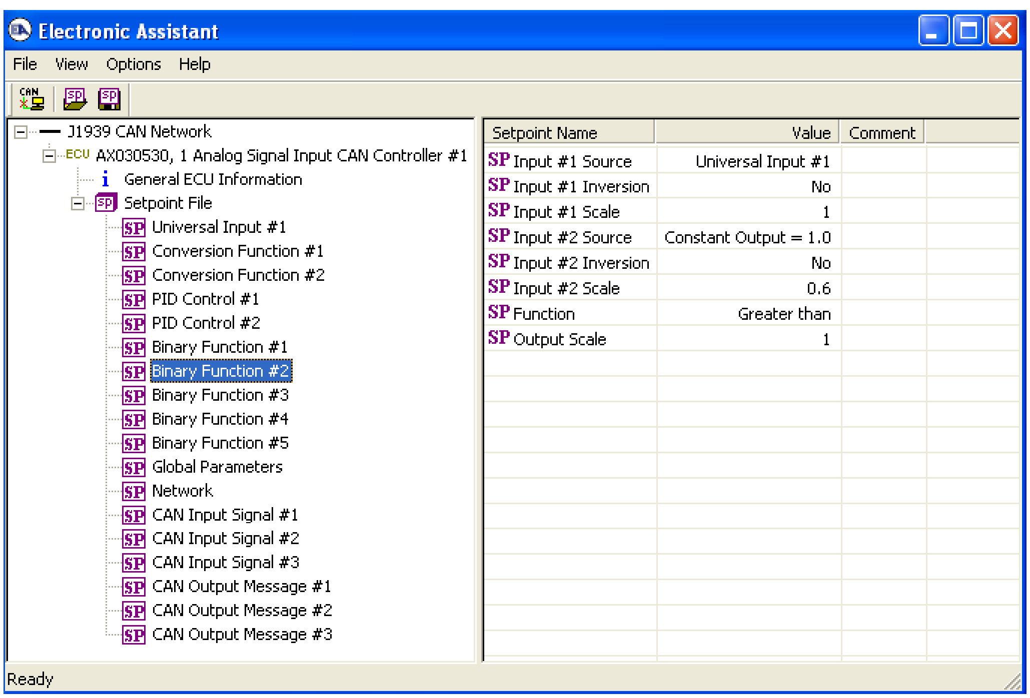


Figure 3. The Block Diagram of the Controller Configuration for the Setpoint Programming Example

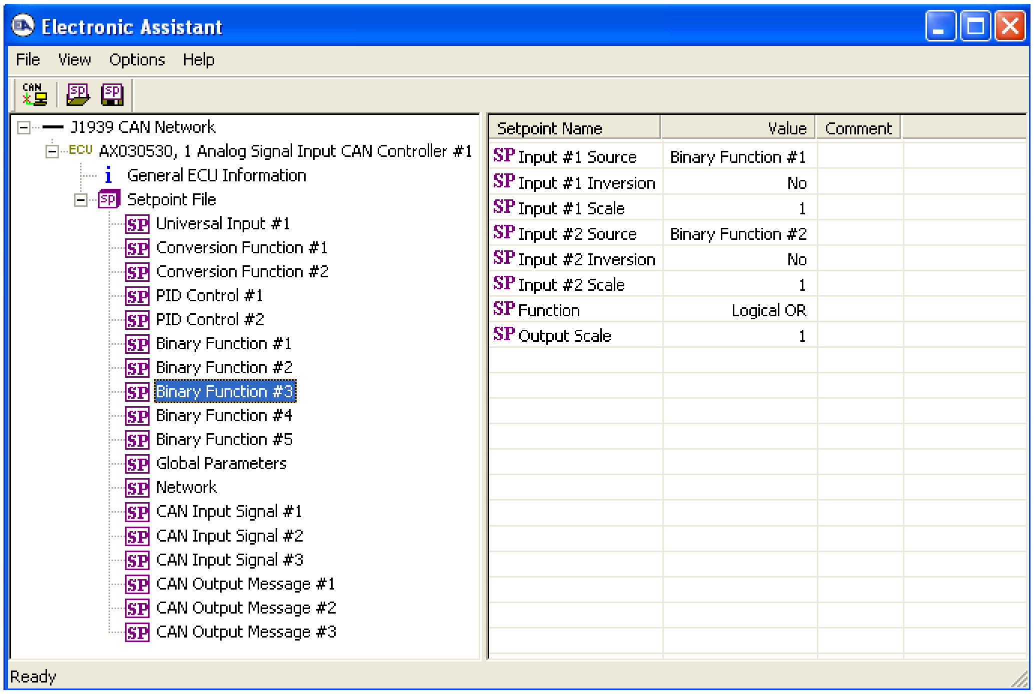
In the Binary Function #1 setpoint group connect *Input #1 Source* to the “Universal Input #1” logical output and *Input #2 Source* to the “Constant Output=1.0”. Set the *Input #2 Scale* to: {(0.5[V] + 0[V])/(5[V]-0[V]) = 0.1} and *Function* to “Less than”. The output of this functional block will be 1 if the input voltage is less than 0.5V.



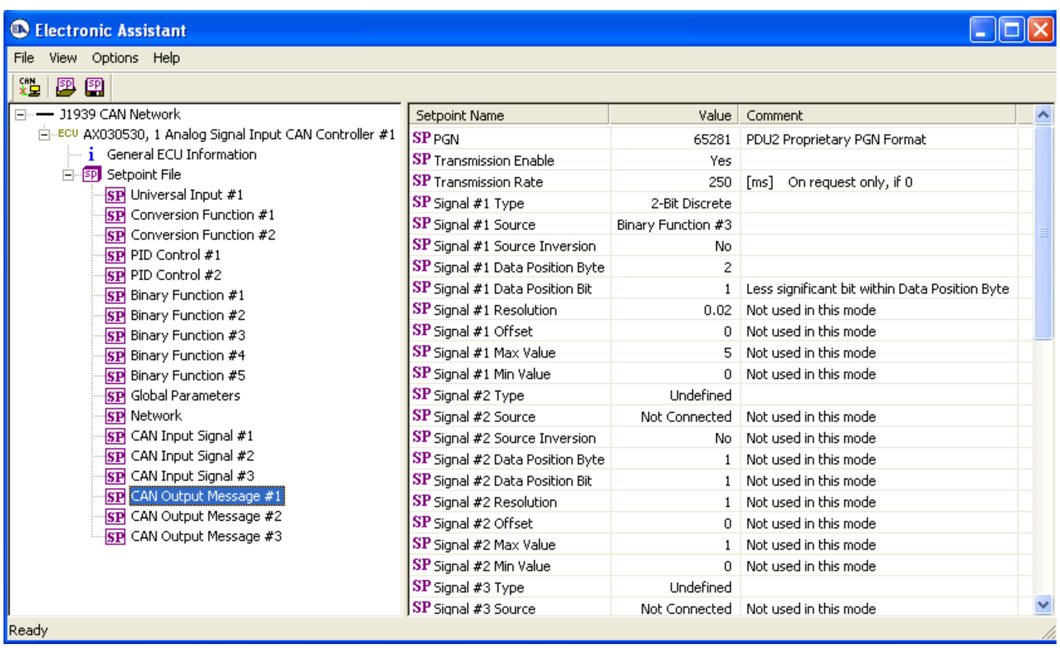
In a similar way, connect the *Input #1 Source* to the "Universal Input #1" logical output and *Input #2 Source* to the "Constant Output=1.0" in the Binary Function #2. Set the *Input #2 Scale* to: {(3[V] + 0[V])/(5[V]-0[V]) = 0.6} and *Function* to "Greater than". The output of this functional block will be 1 if the input voltage is greater than 3V.



Now, to form the alarm signal, combine outputs of the Binary Function #1 and #2. In the Binary Function #3 connect *Input #1 Source* to the "Binary Function #1" and *Input #2 Source* to the "Binary Function #2". Set the *Function* setpoint to the "Logical OR".



As a final step, configure the CAN Output Message #1 functional block. Set *PGN* to 65281, *Transmission Enable* to "Yes" and *Transmission Rate* to 250 ms. Then, configure the Signal Output #1. Set *Signal #1 Type* to "2-Bit Discrete", *Signal #1 Source* to "Binary Function #3" and finally set the position of the discrete signal in the CAN data frame: *Signal #1 Data Position Byte* = 2 and *Signal #1 Data Position Bit* = 1. Keep all other signal outputs in the default disable state.



The controller setpoints are now programmed to perform the user defined functionality. The user can save the controller setpoint configuration into a setpoint file for the future reference or for programming the other controllers with the same functionality.

6 FIRMWARE FLASHING

The controller does not support in-application flashing of the new firmware. It is assumed that in case the firmware upgrade is required, the unit is returned to the manufacturer for re-flashing.

In some special cases, however, the firmware can be reprogrammed through an internal service port in the field by a qualified technician. The flashing instructions, together with a firmware file, RS232 converter and a cable harness, can be obtained from Axiomatic on request.

7 TECHNICAL SPECIFICATIONS

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

Input Specifications

Power Supply Input - Nominal	12V or 24VDC or 48VDC nominal (9…60 VDC power supply range)
Protection	Transient and reverse polarity protection is provided.
Universal Signal Input	1 universal signal input (user selectable) (Voltage, Current, Resistive, Digital, Frequency or PWM) Refer to Table 1.0.
Ground Connection	1 Analog GND connection is provided.

Table 1.0 – **Input – User Selectable Options**	
Analog Input Functions	Voltage Input, Current Input or Resistive Input
Voltage Input	0-1V (Impedance 1 MOhm) 0-2.5V (Impedance 1 MOhm) 0-5V (Impedance 204 KOhm) 0-10V (Impedance 136 KOhm)
Current Input	0-20 mA (Impedance 124 Ohm) 4-20 mA (Impedance 124 Ohm)
Resistive Input	Range: 20Ω to 250 kΩ (Auto Range) User-selectable ranges: 0…150 Ω 0…800 Ω 0…2.5 kΩ 0…8 kΩ 0…25 kΩ 0…80 kΩ 0…250 kΩ
Digital Input Functions	Discrete Input, PWM Input, Frequency Input
Digital Input Level	5V CMOS compatible
PWM Input	0 to 100% 10 Hz to 1kHz 100 Hz to 10 kHz
Frequency Input	10 Hz to 1kHz 100 Hz to 10 kHz
Digital Input	Active High, Active Low
Input Impedance	1 MOhm high impedance, 10KOhm pull down, 10KOhm pull up to +5V
Input Accuracy	$\leq$ 1%
Input Resolution	12-bit

Output Specifications

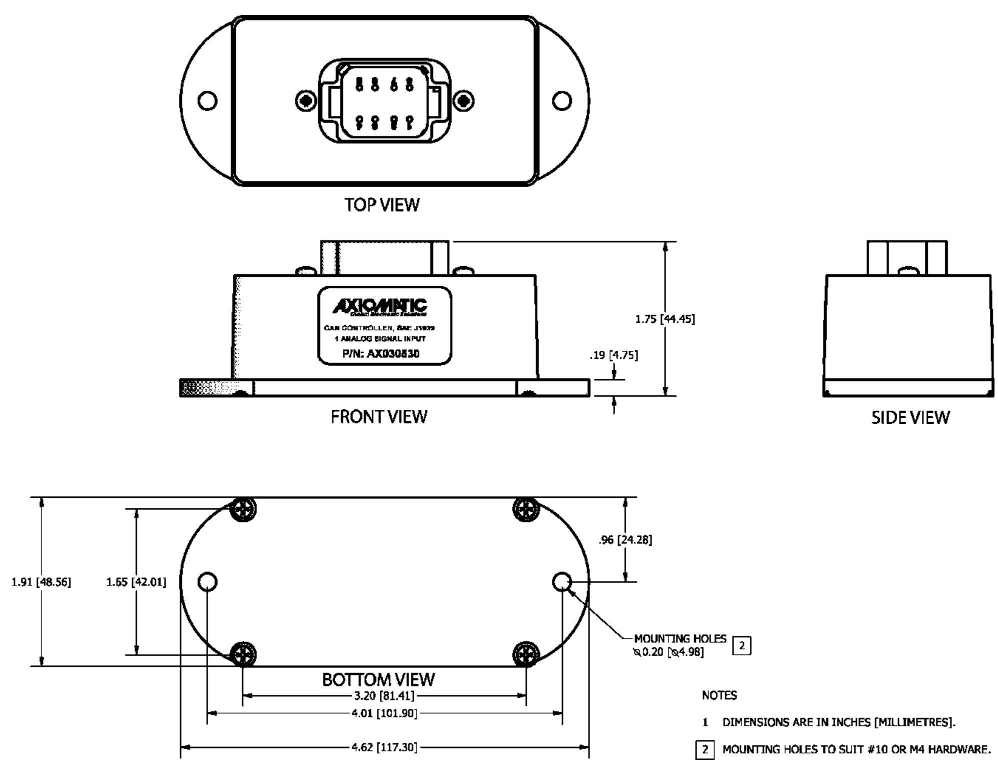
Output	CAN Messages, SAE J1939 {CANopen® available on request} The Axiomatic Electronic Assistant (EA) is used to set up CAN signal acquisition and processing algorithms.
CAN	The controller can send a single frame application specific CAN message to the network continuously or on request. Using the Axiomatic EA, the user can configure this feature.

General Specifications

Microprocessor	32-bit, 128 KByte flash program memory
Control Logic	Standard embedded software is provided. Refer to Figure 1.0. (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 functional 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. Refer to the User Manual for details.
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 Axiomatic 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 Axiomatic 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 Axiomatic Electronic Assistant for *Windows* operating systems comes with a royalty-free license for use on multiple computers. It requires an Axiomatic USB-CAN converter to link the device's CAN port to a *Windows*-based PC. **P/N: AX070502**, an 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: Axiomatic Electronic Assistant software; Axiomatic EA & USB-CAN User Manual UMAX07050X; USB-CAN drivers & documentation; CAN Assistant (Scope and Visual) software & documentation; and the SDK Software Development Kit.
Typical Current Draw	25 mA @ 12V 14 mA @ 24V 8.5 mA @ 48V Conditions: Resistance Input, 0…150Ω, CAN output transmission every 100ms.
Weight	0.65 lbs. (0.29 kg)
Operating Conditions	-40 to 85 °C (-40 to 185 °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

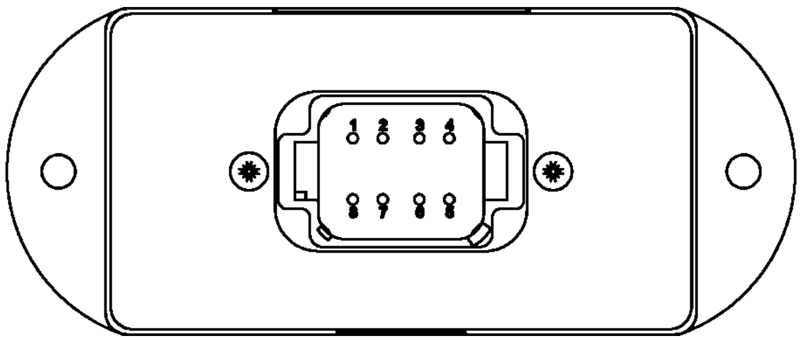


<table>
<tr><td>Mounting</td><td>Mounting holes – The controller accepts 2 #10 or M4 screws.

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).</td></tr>
<tr><td>Network Termination</td><td>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.</td></tr>
<tr><td>Electrical Connections</td><td>



8-pin plug (equivalent TE Deutsch P/N: DT15-8PA)

Mating plug KIT: Axiomatic P/N AX070112
(Equivalent to the TE Deutsch P/Ns: DT016-8SA 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.
<table>
<tr><th>PIN #</th><th>FUNCTION</th></tr>
<tr><td>1</td><td>Power +</td></tr>
<tr><td>8</td><td>Power -</td></tr>
<tr><td>2</td><td>CAN Shield</td></tr>
<tr><td>7</td><td>NOT USED</td></tr>
<tr><td>3</td><td>CAN HI</td></tr>
<tr><td>6</td><td>AGND</td></tr>
<tr><td>4</td><td>CAN L</td></tr>
<tr><td>5</td><td>ANALOG SIGNAL INPUT</td></tr>
</table>
</td></tr>
</table>

8 REVISION HISTORY

User Manual Version	Firmware version	Axiomatic Electronic Assistant (EA) version	Date	Author	Modifications
3E	-	-	August 1, 2023	Kiril Mojsov	• Performed Legacy Updates
3D	3.xx	3.0.33.3 or higher	April 11, 2014	Olek Bogush	• Corrected CAN Input Signals sub-section. • Clarified the Axiomatic EA Software sub-section. • Fixed sub-section numbering in Controller Architecture section. • Changed Revision History section format.
				A. Wilkins	Changed Programming Manual, Revision C, into a User Manual by adding Technical Specifications section.

Programming Manual

Firmware	Manual Revision	Date	Author	Changes
1.xx	A	Jan 5, 2010	Olek Bogush	Initial release.
2.xx	A	June 18, 2010	Olek Bogush	• In the PID Control functional block, the anti-windup algorithm was changed from simply clamping the integral part to stopping the integration process when the output of the functional block saturates and the control error contributes to its further saturation. • Added the number of the functional blocks available in the controller inside the graphical presentation of the functional blocks. • The inversion function formula Inv(...) was removed from all logical inputs of all functional blocks for simplicity. It was mentioned instead that the inputs can be inverted. • Updated Fig. 1…3. Added Conversion Functions on Fig. 1 (which were omitted). • Corrected Inverted Signal Value in a table describing signal inversion. • Changed some functional block drawings. • Clarified description of the Universal Input Function. • Clarified descriptions of the Conversion Function. • Clarified descriptions of the Binary Function. • Clarified descriptions of the PID Function. • J1939 standard document revisions were updated in the Network Support section. • Added hyperlinks to functional block names. • Updated the list of recommended Axiomatic EA versions for different firmware versions.
3.xx	A	August 12, 2010	Olek Bogush	• In the Universal Input functional block added Resistance Range setpoint. Changed Resistance Input description (added resistance ranges, range switching time, etc) and corrected description of other input modes. • Changed Revision History presentation to include all previously released manuals.
	B	Sept. 2, 2010	Olek Bogush	• The internal service port description and usage was clarified in Firmware Flashing section.

	C	Oct 14, 2010	Olek Bogush	• Explained discrete logical inputs in the Controller Architecture section. • Clarified the Reset Input of the PID Control functional block. • Showed input variables on the PID Control functional block diagram.

Global Electronic Solutions

OUR PRODUCTS

AC/DC Power Supplies

Actuator Controls/Interfaces

Automotive Ethernet Interfaces

Battery Chargers

CAN Controls, Routers, Repeaters

CAN/WiFi, CAN/Bluetooth, Routers

Current/Voltage/PWM Converters

DC/DC Power Converters

Engine Temperature Scanners

Ethernet/CAN Converters, Gateways, Switches

Fan Drive Controllers

Gateways, CAN/Modbus, RS-232

Gyroscopes, Inclinometers

Hydraulic Valve Controllers

Inclinometers, Triaxial

I/O Controls

LVDT Signal Converters

Machine Controls

Modbus, RS-422, RS-485 Controls

Motor Controls, Inverters

Power Supplies, DC/DC, AC/DC

PWM Signal Converters/Isolators

Resolver Signal Conditioners

Service Tools

Signal Conditioners, Converters

Strain Gauge CAN Controls

Surge Suppressors

OUR COMPANY

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

QUALITY DESIGN AND MANUFACTURING

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

WARRANTY, APPLICATION APPROVALS/LIMITATIONS

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

COMPLIANCE

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

SAFE USE

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

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

SERVICE

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

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

DISPOSAL

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

CONTACTS

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

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