User Manual UMAX140610
Version 1

USER MANUAL

LIN – J1939 CAN Protocol Converter with PWM Output

P/N: AX140610

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
DM	Diagnostic message. Defined in J1939/73 standard
EA	The Axiomatic Electronic Assistant. The Axiomatic EA is a PC application software primarily designed to view and program Axiomatic control configuration parameters (setpoints) through CAN bus using J1939 Memory Access Protocol
ECU	Electronic control unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMI	Electromagnetic Interference
LED	Light-emitting diode
LIN	Local Interconnect Network. Automotive network maintained by the LIN Consortium
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 the Society of automobile Engineers (SAE)
USB	Universal Serial Bus
UTP	Unshielded twisted pair

Table of Contents

1 INTRODUCTION

The following user manual describes architecture, functionality, configuration parameters and flashing instructions for LIN – J1939 CAN Protocol Converter with PWM Output. It also contains technical specifications and installation instructions to help users build a custom solution on the base of this converter.

The user should check whether the application firmware installed in the converter is covered by this user manual. It can be done through CAN bus using Axiomatic Electronic Assistant (EA) software. The user manual is valid for application firmware with the same major version number as the user manual. For example, this user manual is valid for any converter application firmware V1.xx. Updates specific to the user manual are done by adding letters: A, B, …, Z to the user manual version number.

The converter supports LIN and SAE J1939 CAN interfaces. It is assumed, that the user is familiar with LIN Specification Package and J1939 group of standards. The terminology from these standards is widely used in this manual.

2 CONVERTER DESCRIPTION

The converter is designed to translate application signals between LIN 2.2 and J1939 CAN networks, at the same time, providing a PWM output to interface to an LED, Sound Annunciator or other devise.

The converter accepts power supply voltages from 9 to 36 VDC. It can run in LIN master or slave mode at different baud rate from 2.4…20 kbit/s. The J1939 CAN network can operate at standard 250 kbit/s and 500 kbit/s baud rates and non-standard 667 kbit/s and 1Mbit/s baud rates. The required baud rate is detected automatically upon connection to the CAN network. The converter can be configured through a set of configuration parameters to fit the user specific application requirements.

2.1 Hardware Block Diagram

The converter contains one LIN port, one CAN port and a protected power supply. An embedded 32-bit microcontroller provides necessary processing power to the converter.

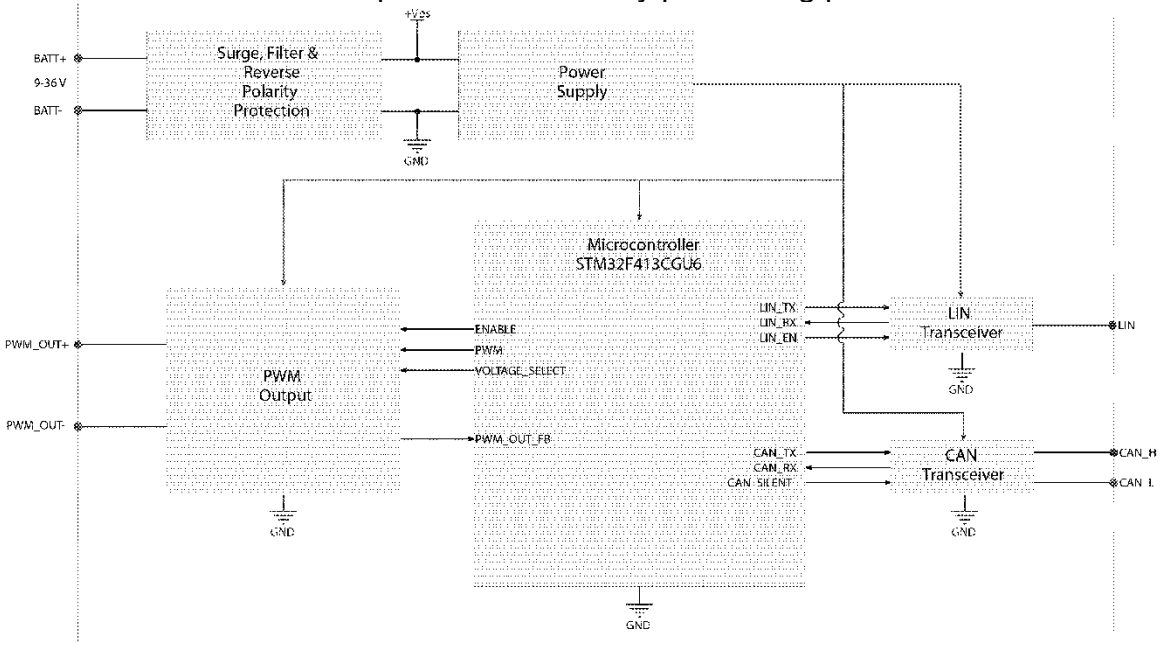


Figure 1. The Converter Hardware Block Diagram

The converter has a wide range of protections features including a transient and reverse polarity protection, see *Technical Specifications* section.

2.2 Software Organization

The LIN – J1939 CAN Protocol Converter with PWM Output belongs to a family of Axiomatic smart controllers with configurable internal architecture. This architecture allows building a converting algorithm based on a set of predefined internal configurable function blocks without the need of custom software.

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

The converter application firmware can be updated the same way using the Axiomatic EA in the field. For more information on this process, see the *Flashing New Firmware* section.

2.3 LIN Interface

The LIN interface is compliant with the LIN Specification Package, Revision 2.2A, December 31, 2010. The following parts of this standard specification package were implemented:

Table 1. LIN Standard Implementation

ISO/OSI Network Model Layer	LIN Specification Package Document
Physical	Physical Layer Specification.
Data Link	Protocol Specification.
Network	Not Implemented.
Transport	N/A in LIN.
Session	N/A in LIN.
Presentation	N/A in LIN.
Application	Application Program Interface Specification. Master and Slave nodes are supported. The user can configure 75 signals, 25 unconditional frames, 1 event triggered and 1 sporadic frame. One main schedule table and one collision schedule table are also available for master nodes.

2.4 CAN Interface

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

Table 2. CAN Standard Implementation

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. J1939/14 - Physical Layer, 500 Kbps. Rev. OCT 2011. J1939/16 – Automatic Baud Rate Detection Process. Rev. NOV 2018.
Data Link	J1939/21 – Data Link Layer. Rev. DEC 2006
	The converter supports Transport Protocol for Commanded Address messages (PGN 65240), ECU identification messages -ECUID (PGN 64965), and software identification messages -SOFT (PGN 65242). It also supports responses on PGN Requests (PGN 59904). Please note that the Proprietary A PGN (PGN 61184) is taken by Axiomatic Simple Proprietary Protocol and is not available for the user.
Network	J1939, Appendix B – Address and Identity Assignments. Rev. FEB 2010. J1939/81 – Network Management. Rev. MAR2017.
	The converter 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 converter supports: Address Claimed Messages (PGN 60928), Requests for Address Claimed Messages (PGN 59904) and Commanded Address Messages (PGN 65240).

ISO/OSI Network Model Layer	J1939 Standard
Transport	N/A in J1939.
Session	N/A in J1939.
Presentation	N/A in J1939.
Application	J1939/71 – Vehicle Application Layer. Rev. APR 2014 with J1939DA – Digital Annex. Rev. OCT 2014.
	The converter can receive and transmit application specific PGNs. All application specific PGNs are user programmable.
	J1939/73 – Application Layer – Diagnostics. Rev. FEB 2010
	Memory access protocol (MAP) supports: DM14, DM15, DM16 messages used by the Axiomatic EA to program configuration parameters.

2.4.1 CAN Baud Rate

The converter can operate at J1939 standard 250 kbit/s and 500 kbit/s baud rates. It can also run at 667 kbit/s and 1Mbit/s – the maximum baud rate supported by the CAN hardware.

The baud rate selection is performed automatically upon connection to the CAN network using passive and active automatic baud rate detection process described in J1939/16. Once detected, the baud rate is stored in non-volatile memory and used on the next converter power-up.

The baud rate detection can be disabled for permanently installed units to maintain the desired baud rate on the CAN network.

2.4.2 J1939 Name and Address

Before sending and receiving any application data, the converter claims its network address with a unique J1939 Name. The Name fields are presented in the table below:

Table 3. J1939 Name Fields

Field Name	Field Length	Field Value	Configurable
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	25 (Network Interconnect ECU)	No
Function Instance	5 bit	25 (AX140610, LIN – J1939 CAN, Axiomatic with PWM Output proprietary)	No
ECU Instance	3 bit	0 (First Instance)	Yes
Manufacturer Code	11 bit	162 (Axiomatic Technologies Corp.)	No
Identity Number	21 bit	Calculated on the base of the microcontroller unique ID	No

The user can change the converter *ECU Instance* using the Axiomatic EA to accommodate multiple converters on the same CAN network.

The converter takes its network *ECU Address* from a pool of addresses assigned to self-configurable ECUs. The default address can be changed during an arbitration process or upon receiving a commanded address message. The new address value will be stored in a non-volatile memory and used next time for claiming the network address. The *ECU Address* can also be changed in the Axiomatic EA.

2.4.3 Network Bus Terminating Resistors

The converter does not have an embedded 120 Ohm CAN bus terminating resistor.

Terminating resistors should be installed externally on both ends of the CAN twisted pair cable according to the J1939/11(15) standards to avoid communication errors.

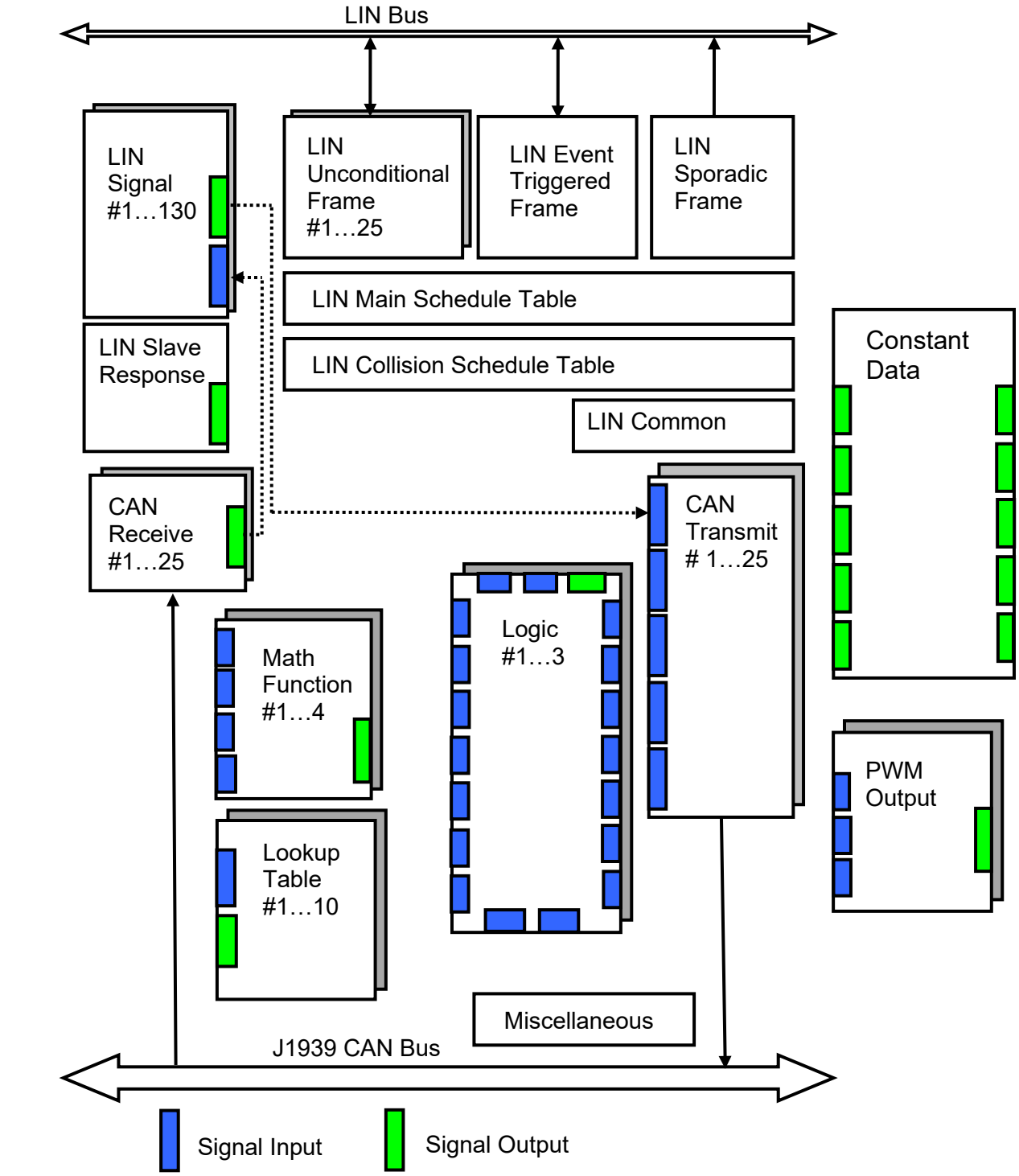
Even if the length of the CAN network is short and the signal reflection from both ends of the cable can be ignored, at least one 120 Ohm resistor is required for the majority of CAN transceivers to operate properly.

2.5 Default Settings

The converter does not have any preprogrammed functionality. See *Configuration Parameters* section for an example of an application-specific converter configuration.

3 CONVERTER LOGICAL STRUCTURE

The converter is internally organized as a set of function blocks, which can be individually configured and arbitrarily connected together to achieve the required system functionality, see Figure 2.



The actual connections between signal inputs and outputs are defined by the configuration parameters.

Figure 2. The Converter Logical Block Diagram

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

LIN interface is presented by the *LIN Signal* function blocks and function blocks controlling sending and receiving of the LIN signals. Each *LIN Signal* function block has one signal input and one signal output. When the *LIN Signal* functional block transmits data on the LIN bus, it reads data from the signal input. When it receives data from the LIN bus, the data is written to the signal output.

J1939 CAN interface is presented by the *CAN Receive, CAN Transmit* and *Miscellaneous* function blocks. The *CAN Receive* functional blocks are used to receive CAN signals transmitted on the CAN bus. They have one signal output, which is updated once the signal is received. The *CAN Transmit* function blocks are used to transmit CAN signals on the CAN bus. Each CAN message can hold up to five individual CAN output signals, which receive data from five signal inputs.

The converter has a PWM output, it can be configured by the *PWM Output* block. It can take three input signals for 3 different uses: Command, Enable or Override. It will be discussed in more detail later. For data processing, when required, the unit can use *Lookup Table, Programmable Logic or Math Function* blocks. They take two input signals and combine them together in one signal output using different functions. They will be described in more detail in further sections.

The converter also has a *Constant Data* function block containing five discrete constant output signals and five continuous constant output signals.

3.1 Function Block Signals

The converter function blocks can communicate with each other through signal inputs and outputs. Each signal input can be connected to any signal output using an appropriate configuration parameter. There is no limitation on the number of signal inputs connected to a signal output.

When a signal input is connected to a signal output, data from the signal output of one function block is available on the signal input of another function block.

Depending on the signal type, the function block signal output can be either *Discrete* or *Continuous*. The function block signal input, receiving the output signal, can be: *Undefined*, *Discrete* or *Continuous.* The *Undefined* input type is reserved for a disconnected input, while: *Discrete* and *Continuous* input types present inputs receiving discrete and continuous signals, respectively.

3.1.1 Undefined Signal

The *Undefined* signal type is used to present a no-signal condition in signal data or to specify that the signal input is not connected (not used).

3.1.2 Discrete Signal

The *Discrete* signal type is used to present a discrete signal that has a finite number of states in signal data or to specify that the signal input or output is communicating this type of signals.

The discrete signals are stored in four-byte unsigned integer variables that can present any state value in the 0…0xFFFFFFFF range.

3.1.3 Continuous Signal

The *Continuous* signal type presents continuous signals, usually physical parameters, in signal data or as a signal input or output type.

The continuous signals are stored in floating point variables. They are not normalized and present data in the appropriate physical units. The user can do simple scaling of the continuous signal data by changing *Scale (Resolution)* and *Offset* configuration parameters in the appropriate function blocks.

3.1.4 Signal Type Conversion

Discrete and *Continuous* signals are automatically converted into each other when a signal input of one signal type is connected to a signal output of a different signal type.

3.1.4.1 Discrete to Continuous Conversion

A *Discrete* signal is converted into a positive *Continuous* signal of the same value.

3.1.4.2 Continuous to Discrete Conversion

A positive *Continuous* signal is converted into the same value *Discrete* signal. A fractional part of the *Continuous* signal is truncated. If the *Continuous* signal value is above the maximum *Discrete* signal value, the resulted *Discrete* signal value will saturate at the maximum *Discrete* signal value: 0xFFFFFFFF.

All negative *Continuous* signals are converted into zero value *Discrete* signals.

3.1.4.3 Undefined Signal Conversion

An *Undefined* signal is not converted into a specific discrete or continuous signal value. It presents a no-signal condition on both: *Discrete* and *Continuous* signal inputs and outputs. The value of an undefined signal is not defined unless a default signal value configuration parameter is used in a function block. In this case, the configuration parameter value is used as a signal value when the signal is not defined.

3.2 PWM Output

Table 4. PWM Output Function Block Configuration Parameters

Name	Default Value	Range	Description
Output Type	1, Digital PWM	Drop List	See Table 5.
Output at Minimum Command	0.00	Depends on Output Type	Digital PWM: Digital Mixed PWM and Frequency: [0…100] %DC Digital Frequency: [0.1…20000] Hz Digital ON/OFF: [0…1]
Output at Maximum Command	100.00	Depends on Output Type	
Ramp Up (Min to Max)	0ms	0 to 10,000	Unit in milliseconds
Ramp Down (Max to Min)	0ms	0 to 10,000	Unit in milliseconds
Fixed Frequency/Duty Cycle	500Hz	Depends on Output Type	Default values depend on output type.
Digital Control Response	0, Normal Logic	Drop List	Only configurable when output type is set to Digital ON/OFF. See Table 6.
Digital Blink Rate	500ms	100 to 5,000	Only configurable when control response is set to Blink Logic.
Digital Type VPS Range	0, 0V to 6V	Drop List	0 = 0V to 6V 1 = 0V to 13V
Control Source	1, CAN Receive Messages	Drop List	See Table 8.
Control Number	1, CAN Receive Messages #1	Per Source	See Table 8.
Control Response	1, Off Below Control Minimum	Drop List	See Table 9.
Enable Source	0, Control Not Used	Drop List	See Table 8.
Enable Number	N/A	Per Source	See Table 8.
Enable Response	N/A	Drop List	See Table 10.
Override Source	0, Control Not Used	Drop List	See Table 8.
Override Number	N/A	Per Source	See Table 8.
Override Response	N/A	Drop List	See Table 11.
Output at Override Command	1V	Depends on Output Type	Same as minimum and maximum command
Fault Response	1, Output Fault Mode Value	Drop List	See Table 12.
Output at Fault Command	0V	Depends on Output Type	Same as minimum and maximum command
Fault Detection is Enabled	True	False or True	See Section 3.2.4 and 3.9
Event Generates a DTC in DM1	False	False or True	See Section 3.2.4 and 3.9
Event Cleared Only by DM11	False	False or True	See Section 3.2.4 and 3.9

Name	Default Value	Range	Description
Lamp Set by Event in DM1	0, Protect	Drop List	See Section 3.2.4 and 3.9
SPN for Event used in DTC	520448 ($7F100)	1 to 524287	See Section 3.2.4 and 3.9
FMI for Event used in DTC	0, Data Above Normal-Most Severe	Drop List	See Section 3.2.4 and 3.9
Delay Before Sending DM1	100 ms	0 to 60,000 ms	See Section 3.2.4 and 3.9

The converter has a PWM output which can be configured by the corresponding function block. The **"Output Type"** setpoint determines what kind of signal the output produces. Changing this parameter will update other parameters in the group to match the selected type. For this reason, it should be the first parameter to be changed. The supported output types by the controller are listed in the table below. By default, the PWM output is configured as '*1, Digital PWM*' type.

Table 5. Output Type Options

Value	Meaning
0	*Output Not Used*
1	***Digital PWM***
2	*Digital Frequency*
4	*Digital Mixed PWM and Frequency*
5	*Digital ON/OFF*

The control signal of the output will have associated with it a minimum and maximum values. Besides type '*Digital ON/OFF'*, all the other output types are always responding in a linear fashion to changes in the control source per the calculation in the figure below.

$$y = mx + a$$

$$m = \frac{Y\max - Y\min}{X\max - X\min}$$

$$a = Y\min - m * X\min$$

Figure 3. Linear Slope Calculations

Where X and Y are defined as:

Xmin = Control Input Minimum Ymin = **"Output At Minimum Command"**
Xmax = Control Input Maximum Ymax = **"Output At Maximum Command"**

In all cases, while the X-axis has the constraint that Xmin < Xmax, there is no such limitation on the Y-axis. This allows for a negative slope so that as the control input signal increases, the target output value decreases. Or it allows output to follow control signal inversely.

3.2.1 Digital PWM/Digital Frequency

Simply setting the **"Output at Minimum Command"** and **"Output at Maximum Command"** to corresponding value in each range will drive the output to different range options. The unit of measurement for PWM output variables is percentage [%] and Hertz [Hz] for the frequency output. Pulse Width Modulated output use a fixed frequency determined by the value in the **"Fixed Frequency/Duty Cycle"** setpoint and frequency output use a fixed duty cycle as selected by this setpoint.

3.2.2 Digital Mixed PWM and Frequency

With the *'Mixed PWM and Frequency'* output type, output control signal controls duty cycle of the output. With this output type duty cycle is considered as main variable, thus associated setpoints are interpreted in percentages, Frequency control signal is selected with **"Frequency Control Source"** and **"Frequency Control Number"** setpoints. Associated **"Mixed Output Frequency Min"** and **"Mixed Output Frequency Max"** setpoints determine minimum and maximum values of the frequency control signal.

3.2.3 Digital ON/OFF

If a non-digital control is selected for this type, the command state will be OFF at or below the minimum input, ON at or above the maximum input, and it will not change in between those points. In other words, the input has its built-in hysteresis, as shown in the figure below. This relationship is true for any function block that has a non-digital input mapped to a digital control.

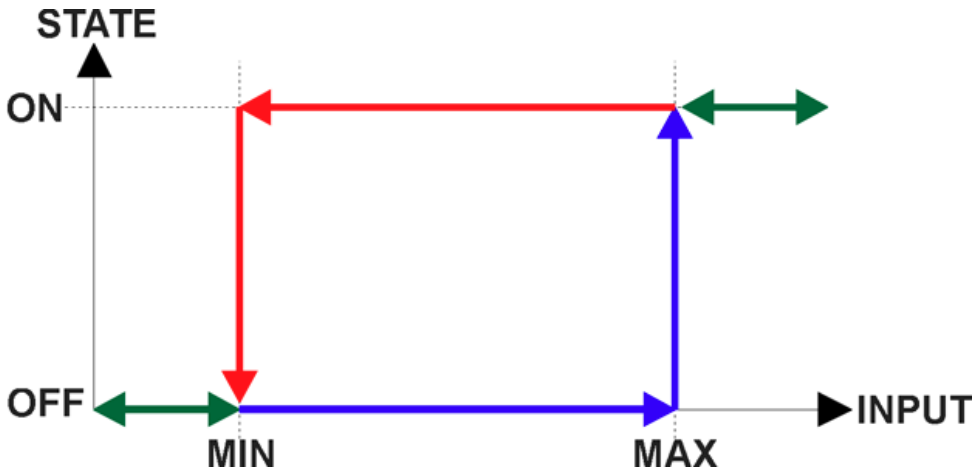


Figure 4. Analog to Digital Input

Only when a *'Digital ON/OFF'* type has been selected will the **"Digital Control Response"** setpoint be enabled as shown in the table below.

Table 6. Digital Control Response Options

Value	Meaning
0	***Normal Logic***
1	*Inverse Logic*
2	*Latched Logic*
3	*Blinking Logic*

When the output is being driven by the Control Input, the state is logically set to OFF when the Control Input is zero and is set to ON whenever a non-zero value is written. By default, '*Normal Logic'* is used. The resulting Drive State will depend on the "**Digital Control Response**" as the table below.

Table 7. Digital ON/OFF Logic

Value	Meaning	Logic State	Drive State
0	***Normal Logic***	OFF	OFF
		ON	ON
1	*Inverse Logic*	OFF	ON
		ON	OFF
2	*Latched Logic*	ON to OFF	No change
		OFF to ON	State change (i.e., OFF to ON or ON to OFF)
3	*Blinking Logic*	OFF	OFF
		ON	Toggling OFF and ON at the rate defined in setpoint "**Digital Blink Rate**" (in ms)

3.2.4 Common Parameters

In order to prevent abrupt changes at the output due to sudden changes in the command input, the user can choose to use the independent up or down ramps to smooth out the response. The **"Ramp Up (Min to Max)"** and **"Ramp Down (Max to Min)"** parameters are in milliseconds, and the step size of the output change will be determined by taking the absolute value of the output range and dividing it by the ramp time. However, these setpoints are set to zero by default since in most signal conversion applications, fast response times are desired. Another setpoint **"Digital Type VPS range"** determines if the signal will toggle between 0V and *+6V* or *+13V*.

By default, the **"Control Source"** is setup to be *'CAN Receive Message'.* In other words, the output will response in a linear fashion to the corresponding CAN received command data. The **"Control Source"** together with **"Control Number"** parameter determine which signal is used to drive the output. For example, setting **"Control Source"** to *'CAN Receive Message'* and **"Control Number"** to '*1'* will connect signal measured from CAN Receive 1 to the output in question. The options for **"Control Sources"** and available **"Control Number"** are listed in the table below.

Table 8. Control Source Options

Value	Meaning	Control Number Range
0	*Control Not Used*	[1]
1	***CAN Receive Message***	**[1…10]**
2	*LIN Signal*	[1…130]
3	*LIN Slave Node Response Error*	[1]
4	*Constant Discrete Data*	[1…5]
5	*Constant Continuous Data*	[1…5]
6	*Lookup Table Block*	[1…10]
7	*Math Function Block*	[1…4]
8	*Programmable Logic Block*	[1…3]
9	*PWM Output Command*	[1]
10	*PWM Output Feedback*	[1]
11	*Power Supply Measured*	[1]
12	*Temperature Measured*	[1]

In general, when the control input is within its range, output will respond in a linear fashion to changes in the control source as the calculation in Figure 3 and when the control input is out of range, output will respond with the minimum/maximum value in the output range. However, in some cases it may be desired that the minimum offset not to be applied when the value is outside of the range, i.e. when using a joystick profile with a deadband. For this reason, setpoint "**Control Response**" has the options shown in the table below.

Table 9. Control Response Options

Value	Meaning
0	Single Output Profile
1	**Off Below Control Minimum**
2	Off Above Control Maximum

In addition to the Control input, the function block also supports an enable input which can be setup as either an enable or disable signal.

When an Enable input is used, the output will be shutoff as per the **“Enable Response”** in the table below. If the response is selected as a disable signal (3 or 4), when the enable input is ON, the output will be shut off.

Table 10. Enable Response Options

Value	Meaning
0	*Enable When On, Else Shutoff*
1	*Enable When On, Else Rampoff*
2	*Enable When On, Else Keep Last Value*
3	***Enable When Off, Else Shutoff***
4	*Enable When Off, Else Rampoff*
5	*Enable When Off, Else Keep Last Value*

The Override option allows the user to choose whether or not to drive the output with the override input being engaged/disengaged, depending on the logic selected in **“Override Response.”** The options for **“Override Response”** listed in the table below. When override is active, the output will be driven to the value in “**Output at Override Command**” regardless of the value of the Control input.

Table 11. Override Response Options

Value	Meaning
0	***Override When ON***
1	*Override When OFF*

The options for both **“Enable Source”** and **“Override Source”** are same as sources listed in Table 8.

When an input to the output block goes into an error condition, setpoint “**Fault Response**” determines how the output will respond as the table below. By default, the output will be driven to the value defined in setpoint “**Output at Fault Command**” which is set to 0 by default.

Table 12. Fault Response Options

Value	Meaning
0	*Maintain Last State*
1	***Output Fault Mode Value***

Fault conditions are checked for first, and only if they are not present are the control signal then evaluated. If Enable, Override and Control inputs are all used, the Enable logic is evaluated first, then the Override, and lastly the Control. The logic flow chart for evaluating the output drive is shown in the figure below.

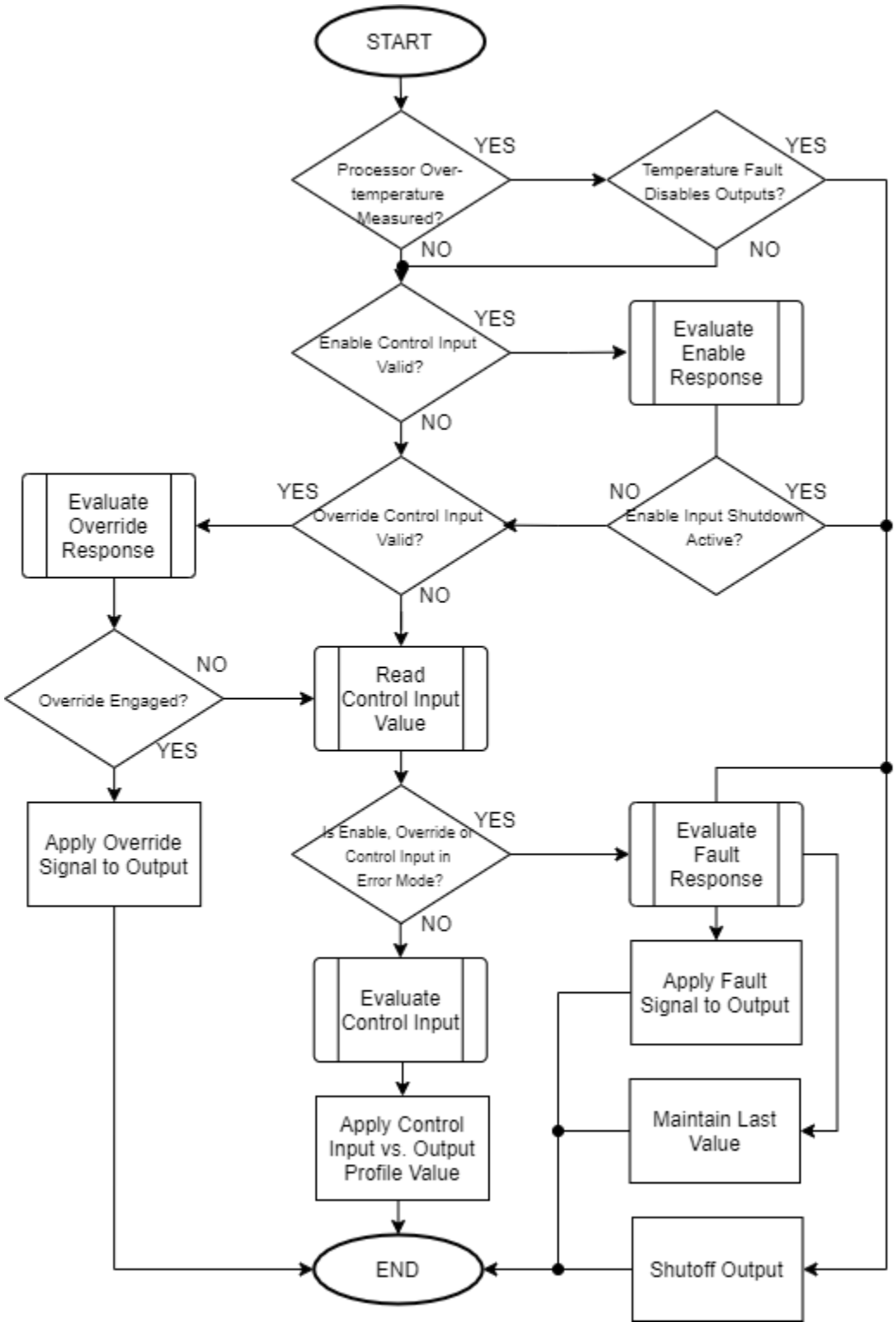


Figure 5. Output Logic Flowchart

The output is inherently protected against a short/open circuit. In the case of a dead short, or an open circuit, the hardware will automatically disable the output drive, regardless of what the processor is commanding for the output. The processor will be flagged by the hardware that the output circuitry has been shutoff due to a short or an open at the output, but it will not know which type it is.

For this reason, both types of errors would be identified on the CAN network with the same Failure Mode Indicator (FMI). Unless commanded OFF by some other logic, when an error disables the output, the processor will still attempt to drive the output to the desired value. When the error is removed, the output automatically recovers and resumes normal operation.

See Section 3.9 for the complete description of the diagnostic functions available on this coverter.

3.3 LIN Interface

The converter LIN interface is defined by *LIN Signal*, *LIN Unconditional Frame*, *Event Triggered Frame*, *Sporadic Frame*, *Main Schedule Table, Collision Schedule Table* and *LIN Common* function blocks.

LIN signals are sent and received through *LIN Signal* function blocks. Other function blocks are used to define the LIN network communication.

The function blocks will be presented in the order they appear in the Axiomatic EA.

3.3.1 LIN Common

LIN Common function block does not have any signal inputs and outputs.

Figure 6. LIN Common Function Block

It defines the high-level LIN bus configuration parameters, see the following table.

Table 13. LIN Common Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Node Type	Undefined	{Undefined, Master, Slave}	–	LIN node type. The “Undefined” value is used to switch off the node, for example during debugging.
Baud Rate	19200	[2400…20000]	bit/s	LIN bus baud rate. The minimum baud rate value is limited by the converter transceiver hardware.
Tick Time	10	[1…10000]	ms	Tick time, if Node Type is “Master”.

3.3.2 LIN Signal

LIN Signal function blocks are used to specify input and output signals on the LIN bus. There are *130 LIN Signal* function blocks available to the user. Each *LIN Signal* function block has one signal input and one signal output for interfacing with other function blocks.

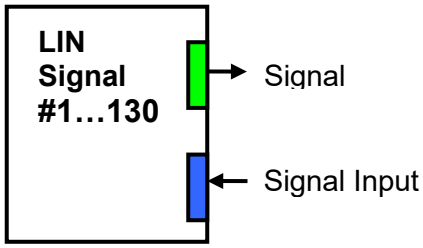


Figure 7. LIN Signal Function Block

Configuration parameters of the *LIN Signal* function block are presented below:

Table 14. LIN Signal Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
LIN Signal Type	Undefined	{Undefined, Scalar, Byte Array}	–	Type of the LIN signal.
Input Signal Source	Not Connected	Any signal output of any function block or “Not Connected”	–	Input signal source if LIN signal is output (data is sent on the LIN bus).
Output Signal Autoreset Time	1000	[1…10000]	ms	Autoreset time if LIN signal is input (data is received from the LIN bus). If 0 – Autoreset is disabled.
Size	1	[1…64]	bit	Signal size.
Encoding Type	Undefined	{Undefined, Logical Value, Physical Value, BCD Value, ASCII Value}	–	Type of the LIN signal content.
Min Value	0	[0…0xFFFFFFFF]	–	Min value of the signal code when the Encoding Type is “Logical Value” or “Physical Value”.
Max Value	1	[0…0xFFFFFFFF]	–	Max value of the signal code when the Encoding Type is “Logical Value” or “Physical Value”.
Scale	1	Any value	Signal Units/bit	Signal scale when the Encoding Type is “Physical Value”.
Offset	0	Any value	Signal Units	Signal offset when the Encoding Type is “Physical Value”.
Init Value Scalar	0	[0…0xFFFF]	–	Initial signal value when LIN Signal Type is Scalar.

Name	Default Value	Range	Units	Description
Init Value Byte Array [0]	0	[0…0xFF]	–	Initial signal value of the 1-st byte when LIN Signal Type is “Byte Array”.
Init Value Byte Array [1]	0	[0…0xFF]	–	Initial signal value of the 2-nd byte when LIN Signal Type is “Byte Array”.
…	…	…	…	…
Init Value Byte Array [7]	0	[0…0xFF]	–	Initial signal value of the 8-th byte when LIN Signal Type is “Byte Array”.

Encoding Type configuration parameter defines the function block signal input and output type the following way:

Table 15. LIN Signal Encoding Type

Encoding Type	Function Block Signal Type
Undefined	Undefined
Logical Value	Discrete
Physical Value	Continuous
BCD Value	Discrete
ASCII Value	Discrete

3.3.2.1 Receiving LIN Signals

When the *LIN Signal* function block receives signals from the LIN bus, they are converted to the function block output signal the following way:

- Logical signals are received only when they are within the [*MinValue; MaxValue*] range. No conversion is performed;
- BCD signals are received unconditionally. No conversion is performed;
- ASCII signals are masked with 0xFF value. This way, only the least significant byte of the input signal is received and passed to the output;
- Physical signals are received only when they are within the [*MinValue; MaxValue*] range. They are then converted to the output signal value using *Scale* and *Offset* configuration parameters:

$$LINOutputSignal = LINSignalCode * Scale + Offset, \quad if \qquad (1)$$

$$LINSignalCode \in [MinValue;\ MaxValue].$$

If the output signal is not updated within the *Output Signal Autoreset Time*, the output signal will be reset to undefined. The auto-reset is disabled when the *Output Signal Autoreset Time* configuration parameter is equal to 0.

The initial value of the *LIN Signal* function block output signal is undefined.

3.3.2.2 Transmitting LIN Signals

When the *LIN Signal* function block transmits signals on the LIN bus, the signal data are acquired from the function block signal input and processed, before transmission, in a way similar to the incoming LIN signals:

- Logical signals are transmitted only when they are in the [*MinValue; MaxValue*] range;
- BCD signals are transmitted without any conversion;
- ASCII signals are masked with 0xFF value and then transmitted;
- The physical signals are converted to the LIN signal code using *Scale* and *Offset* configuration parameters and then saturated to the *MinValue* or *MaxValue* if the code goes out of the [*MinValue; MaxValue*] range.
- Undefined signals are presented by their initial signal values.

3.3.3 LIN Slave Response

There is a special *LIN Slave Response* function block with one output signal.

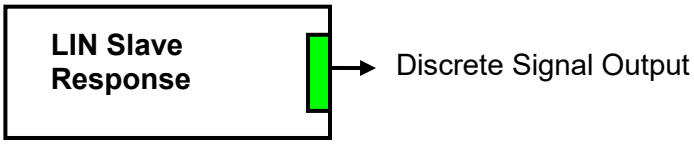


Figure 8. LIN Slave Response Function Block

This function block does not have any configuration parameters. It provides a discrete logical output signal reflecting the state of the LIN slave node according to the LIN standard requirements.

In the master mode, this function block is not used, and its output signal is undefined.

3.3.4 LIN Unconditional Frame

There are *25 LIN Unconditional Frame* function blocks available to the user. Each function block represents one LIN frame that can be sent or received on the LIN bus.

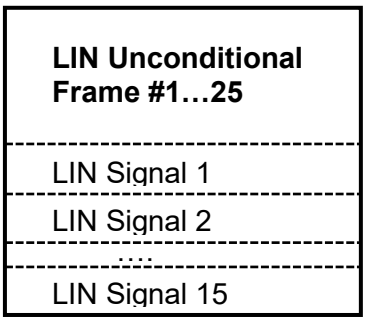


Figure 9. LIN Unconditional Frame Function Block

There are no signal inputs and outputs in this function block; all data is transmitted or received through the associated *LIN Signal* function blocks. There can be up to 15 LIN signals associated with each LIN frame.

Configuration parameters of the *LIN Unconditional Frame* function block are presented below:

Table 16. LIN Unconditional Frame Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
LIN Frame Kind	Undefined	{Undefined, Publish, Subscribe}	–	Defines whether the frame is transmitted or received.
Frame ID	0	0…0x3F	–	Frame ID.

Name	Default Value	Range	Units	Description
Associated with Event Triggered Frame	False	{False, True}	–	Defines whether this Frame is used in the Event Triggered Frame.
Size	1	[1…8]	byte	Frame Size.
Checksum Type	Classic	{Classic, Enhanced}	–	Type of the frame checksum.
Signal #1 Number	0	[0…130]	–	Number of the 1-st *LIN Signal* function block.
Signal #1 Offset	0	[0…63]	bit	Offset of the LIN signal defined by the 1-st *LIN Signal* function block.
Signal #2 Number	0	[0…130]	–	Number of the 2-nd *LIN Signal* function block.
Signal #2 Offset	0	[0…63]	bit	Offset of the LIN signal defined by the 2-nd *LIN Signal* function block.
…	…	…	…	…
Signal #15 Number	0	[0…130]	–	Number of the 15-th *LIN Signal* function block.
Signal #15 Offset	0	[0…63]	bit	Offset of the LIN signal defined by the 15-th *LIN Signal* function block.

The *LIN Signal* function blocks are numbered from 1 to 15. When the *Signal #1…15 Number* is equal to 0, the *LIN Signal* function block is not defined and the associated *Signal #1…15 Offset* configuration parameter is not used.

3.3.5 LIN Event Triggered Frame

There is one *LIN Event Triggered Frame* function block available to the user. It can contain up to 5 associated LIN frames. There are no signal inputs and outputs in this function block.

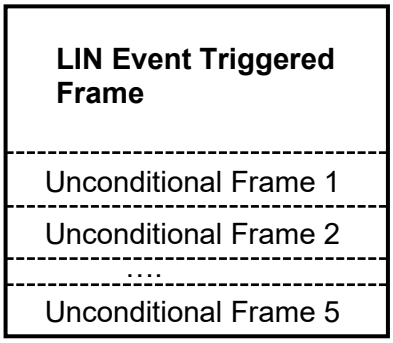


Figure 10. LIN Event Triggered Frame Function Block

Configuration parameters of the *LIN Event Triggered Frame* function block are presented below:

Table 17. LIN Event Triggered Frame Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
LIN Frame Kind	Undefined	{Undefined, Publish, Subscribe}	–	Defines whether the frame is transmitted or received.
Frame ID	0	0…0x3F	–	Frame ID.
Size	1	[1…8]	byte	Frame Size.
Checksum Type	Classic	{Classic, Enhanced}	–	Type of the frame checksum.
Collision Resolving Schedule Table Number	0	[0…1]	–	Number of the *Collision Resolving Schedule Table* function block used in case of a collision.
Unconditional Frame #1 Number	0	[0…25]	–	Number of the 1-st *LIN Unconditional Frame* function block.
Unconditional Frame #2 Number	0	[0…25]	–	Number of the 2-nd *LIN Unconditional Frame* function block.
…	…	…	…	…
Unconditional Frame #5 Number	0	[0…25]	–	Number of the 5-th *LIN Unconditional Frame* function block.

Collision Resolving Schedule Table function blocks are numbered starting from 0. When the *Collision Resolving Schedule Table Number* is equal to 0, the function block is not defined. The same rule applies to the *LIN Unconditional Frame* function blocks.

When the event triggered frame is defined, all associated unconditional frames should have the same frame size and checksum type as defined in the *LIN Event Triggered Frame* function block. They should also have *Associated with Event Triggered Frame* configuration parameter set to *Yes*.

3.3.6 LIN Sporadic Frame

There is one *LIN Sporadic Frame* function block available to the user. It can contain up to 5 associated LIN frames. There are no signal inputs and outputs in this function block.

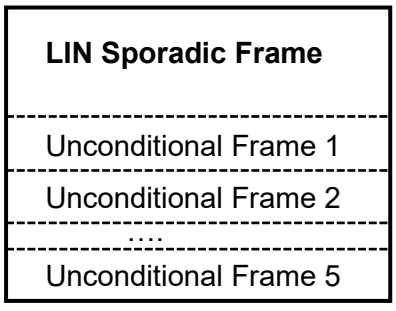


Figure 11. LIN Sporadic Frame Function Block

This block is used only by master nodes, see: *Node Type* configuration parameter in the *LIN Common* function block.

Configuration parameters of the *LIN Sporadic Frame* function block are presented below:

Table 18. LIN Sporadic Frame Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
LIN Frame Kind	Undefined	{Undefined, Publish, Subscribe}	–	Defines whether the frame is transmitted or received.
Unconditional Frame #1 Number	0	[0…25]	–	Number of the 1-st *LIN Unconditional Frame* function block.
Unconditional Frame #2 Number	0	[0…25]	–	Number of the 2-nd *LIN Unconditional Frame* function block.
…	…	…	…	…
Unconditional Frame #5 Number	0	[0…25]	–	Number of the 5-th *LIN Unconditional Frame* function block.

Sending priorities in the sporadic frame are defined in the descending order: unconditional frame #1 has the maximum priority and unconditional frame #5 – minimum. When *Unconditional Frame #1…5 Number* is equal to 0, the frame is undefined.

3.3.7 Main Schedule Table

There is one *Main Schedule Table* function block available to the user. It is used by the master node by default and can contain up to 25 schedule entries. There are no signal inputs and outputs in this function block.

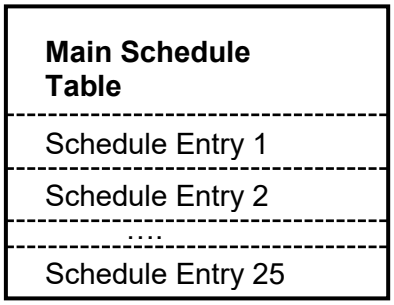


Figure 12. Main Schedule Table Function Block

Configuration parameters of the *Main Schedule Table* function block are presented below:

Table 19. Main Schedule Table Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Entry #1 Frame Type	Undefined	{Undefined, Unconditional, Event Triggered, Sporadic}	–	1-st schedule entry frame type.
Entry #1 Frame Number	0	Depends on the frame type	–	1-st schedule entry frame number.
Entry #1 Delay	0	[0…10000]	ms	1-st schedule entry delay.

Name	Default Value	Range	Units	Description
Entry #2 Frame Type	Undefined	{Undefined, Unconditional, Event Triggered, Sporadic}	–	2-nd schedule entry frame type.
Entry #2 Frame Number	0	Depends on the frame type	–	2-nd schedule entry frame number.
Entry #2 Delay	0	[0…10000]	ms	2-nd schedule entry delay.
…	…	…	…	…
Entry #25 Frame Type	Undefined	{Undefined, Unconditional, Event Triggered, Sporadic}	–	25-th schedule entry frame type.
Entry #25 Frame Number	0	Depends on the frame type	–	25-th schedule entry frame number.
Entry #25 Delay	0	[0…10000]	ms	25-th schedule entry delay.

When the frame type is undefined, the schedule entry is skipped. When the frame number is equal to 0, the schedule entry is empty. When the schedule table is used, the first schedule entry should be defined.

3.3.8 Collision Schedule Table

There is one *Collision Schedule Table* function block available to the user. It is used by the master node to resolve collisions in the event triggered frames. *Collision Schedule Table* function block has the same structure as the *Main Schedule Table*. The only difference is in the number of the schedule table entries.

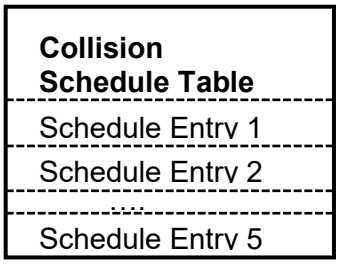


Figure 13. Collision Schedule Table Function Block

The collision schedule table can contain only up to 5 schedule entries. The schedule entries follow the same rules as defined for the *Main Schedule Table* with the exception that only unconditional frames are allowed in the collision schedule table.

Configuration parameters of the *Collision Schedule Table* function block are presented below:

Table 20. Collision Schedule Table Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Entry #1 Frame Type	Undefined	{Undefined, Unconditional}	–	1-st schedule entry frame type.

Name	Default Value	Range	Units	Description
Entry #1 Frame Number	0	[0…25]	–	1-st schedule entry frame number.
Entry #1 Delay	0	[0…10000]	ms	1-st schedule entry delay.
Entry #2 Frame Type	Undefined	{Undefined, Unconditional}	–	2-nd schedule entry frame type.
Entry #2 Frame Number	0	[0…25]	–	2-nd schedule entry frame number.
Entry #2 Delay	0	[0…10000]	ms	2-nd schedule entry delay.
…	…	…	…	…
Entry #5 Frame Type	Undefined	{Undefined, Unconditional}	–	5-th schedule entry frame type.
Entry #5 Frame Number	0	[0…25]	–	5-th schedule entry frame number.
Entry #5 Delay	0	[0…10000]	ms	5-th schedule entry delay.

3.4 Lookup Table Function Block

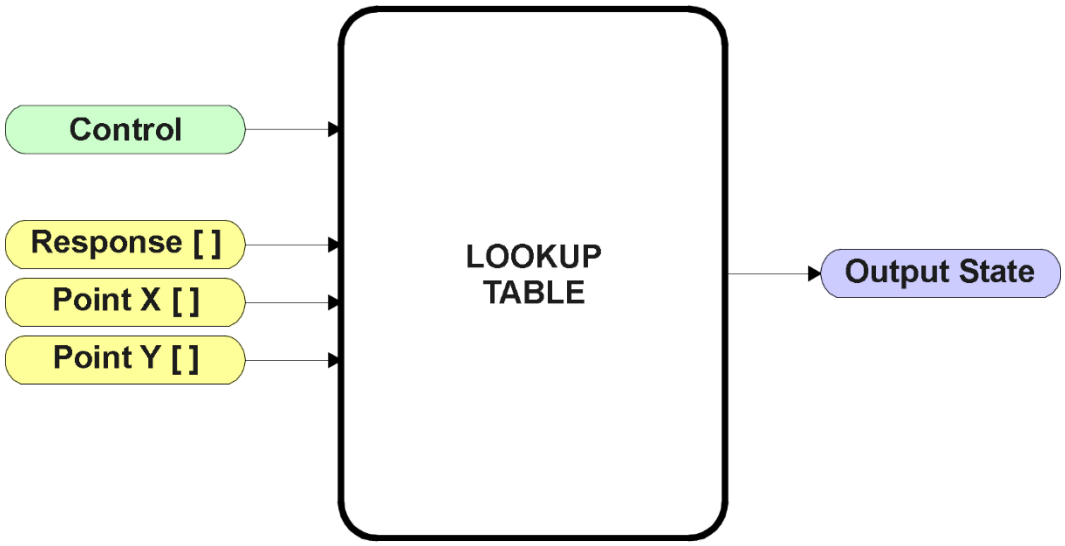


Figure 14. Lookup Table Function Block

Lookup Tables are used to give an output response of up to 10 slopes per input. The array size of the Response [ ], Point X [ ] and Point Y [ ] setpoints shown in the block diagram above is therefore 11.

Note: If more than 10 slopes are required, a Programmable Logic Block can be used to combine up to three tables to get 30 slopes, as is described in Section 3.5.

There are two key setpoints that will affect this function block. The first is the "**X-Axis Source**" and **"X-Axis Number"** which together define the Control Source for the function block. When it is changed, the points are automatically updated with new defaults based on the X-Axis source

selected. However, there are 2 more setpoints "**X Decimal Digits**" and "**Y Decimal Digits**" that might affect the point values. These 2 setpoints will change the resolution of the X Values or Y Values, respectively, with the decimal digits value user chooses. Keep in mind that any change on the decimal digits will make effect to the current X values or Y values.

Initialize the Control Source of a Lookup Table BEFORE changing the table values, as the new settings WILL get erased when the control is updated.

The second setpoint that will affect the function block (i.e. reset to defaults), is the "**X-Axis Type**". By default, the tables have a *'Data Response'* output. Alternatively, it can be selected as a *'Time Response',* which is described later in Section 3.4.5.

3.4.1 X-Axis, Input Data Response

In the case where the **X-Axis Type**" = *'Data Response',* the points on the X-Axis represents the data of the control source.

For example, if the control source is a CAN Receive message, setup as a 0-5V type, with an operating range of 0.5V to 4.5V, the X-Axis will be setup to have a default "**Point 1 – X Value**" of 0.5V, and setpoint "**Point 10 – X Value**" will be set to 4.5V. The "**Point 0 – X Value**" will be set to the default value of 0.0V.

For most *'Data Responses'*, the default value at point (0,0) is [0,0].

However, should the minimum input be less than zero, for example a CAN message that is reflecting temperature in the range of -40ºC to 210ºC, then the "**Point 0 – X Value**" will be set to the minimum instead, in this case -40ºC.

The constraint on the X-Axis data is that the next index value is greater than or equal to the one below it, as shown in the equation below. Therefore, when adjusting the X-Axis data, it is recommended that X_{10} is changed first, then lower indexes in descending order.

MinInputRange <= X_0 <= X_1 <= X_2<= X_3<= X_4<= X_5<= X_6<= X_7<= X_8<= X_9<= X_{10}<= MaxInputRange

As stated earlier, MinInputRange and MaxInputRange will be determined by the X-Axis Source that has been selected.

If some of the data points are *'Ignored'* as described in Section 3.4.4, they will not be used in the X-Axis calculation shown above. For example, if points X_4 and higher are ignored, the formula becomes MinInputRange <= X_0 <= X_1 <= X_2<= X_3<= MaxInputRange instead.

3.4.2 Y-Axis, Lookup Table Output

The Y-Axis has no constraints on the data that it represents. This means that inverse, increasing/decreasing or other responses can be easily established.

For example, should the X-Axis of a table be a resistive value (as read from another controller), the output of the table could be temperature from an NTC sensor in the range Y_0=125ºC to Y_{10}= -20ºC. If this table is used as the control source for another function block

(i.e. transmitted over CAN), then Xmin would be -20 and Xmax would be 125 when used the linear formula.

In all cases, the controller looks at the **entire range** of the data in the Y-Axis setpoints and selects the lowest value as the MinOutRange and the highest value as the MaxOutRange. They are passed directly to other function blocks as the limits on the Lookup Table output. (i.e used as Xmin and Xmax values in linear calculations.)

However, if some of the data points are *'Ignored'* as described in Section 3.4.4, they will not be used in the Y-Axis range determination. Only the Y-Axis values shown on the Axiomatic EA will be considered when establishing the limits of the table when it is used to drive another function block, such as an PWM Output.

3.4.3 Default Configuration, Data Response

By default, all Lookup Tables in the ECU are disabled (**"X-Axis Source"** equals *'Control Source Not Used'.*) If they were to use the default settings for CAN Receive messages instead as the X-Axis and output Duty cycle (percentages), they could be used to control the PWM Output. If a non-linear response for one or more of the outputs is required, the user can easily use the table(s) to create the desired response profiles.

Recall, any controlled function block which uses the Lookup Table as an input source (not only the PWM Output) will also apply a linearization to the data. **Therefore, for a 1:1 control response, ensure that the minimum and maximum values of the output (Ymin and Ymax in Figure 3) correspond to the minimum and maximum values of the table's Y-Axis (Xmin and Xmax in Figure 3).**

To control "PWM Output" by "CAN Received Message 1" modified by "Lookup Table 1", it is recommended to do so in the following order:

a) Change PWM Output **"Output at Minimum Command"** and **"Output at Maximum Command"** to the desired limits.
b) Configure the desired Control Source (i.e. CAN Receive Message) and set the appropriate limits.
c) Change the Lookup Table 1 **"X-Axis Source"** setpoints. (If applicable)

At this point, the X-Axis limits will match the control source, and the Y-Axis limits and the Y-Axis limits would correspond to the PWM Output range, as a percentage.

d) Update the X and Y setpoints for the application

Note: Order (b) to (d) holds true for all configuration done using any Lookup Table function block.

All tables (1 to 10) are disabled by default (no control source selected). However, should an **"X-Axis Source"** be selected, the Y-Axis defaults will be in the range of 0 to 100% as described in the "Y-Axis, Lookup Table Output" section above. X-Axis minimum and maximum defaults will be set as described in the "X-Axis, Data Response" section above.

By default, the X and Y axes data is setup for an equal value between each point from the minimum to maximum in each case.

For example, with a 0.5 to 4.5V input (X-Axis) driving a 0 to 1500mA output (Y-Axis), the default points would be setup as per figure (a) below. However, a 100Ω to 54kΩ input (X-Axis) representing 120ºC to -30ºC (Y-Axis) would be setup as per figure (b) below. In each case, the user would have to adjust the table for the desired response.

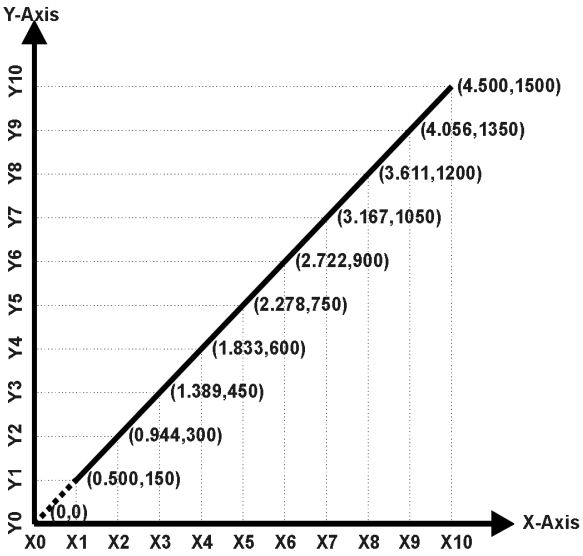


Figure A - 0.5 to 4.5V Input, 0 to 1500mA Output

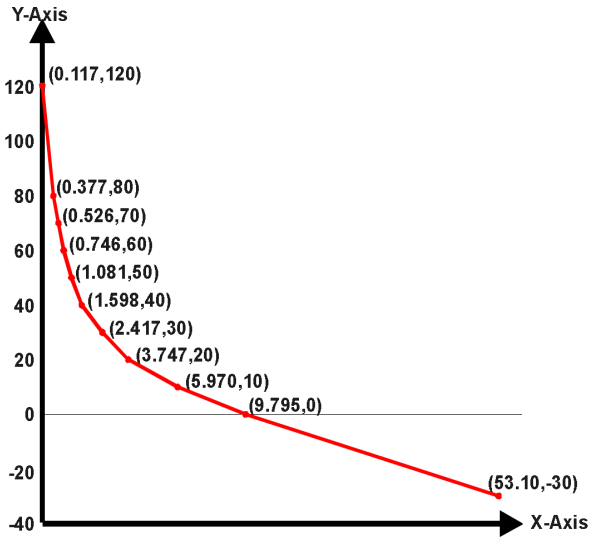


Figure B - 0.1 to 54kOhm Input, 120 to -30ºC Output

Figure 15. Lookup Table Initialization Examples

3.4.4 Point To Point Response

By default, the X and Y axes are setup for a linear response from point (0,0) to (10,10), where the output will use linearization between each point, as shown in Figure 3. To get the linearization, each "**Point N – Response**", where N = 1 to 10, is setup for a *'Ramp To'* output response.

Alternatively, the user could select a *'Jump To'* response for "**Point N – Response**", where N = 1 to 10. In this case, any input value between X_{N-1} to X_N will result in an output from the Lookup Table function block of Y_N.

An example of a CAN message (0 to 100) used to control a default table (0 to 100) but with a *'Jump To'* response instead of the default *'Ramp To'* is shown in the figure below.

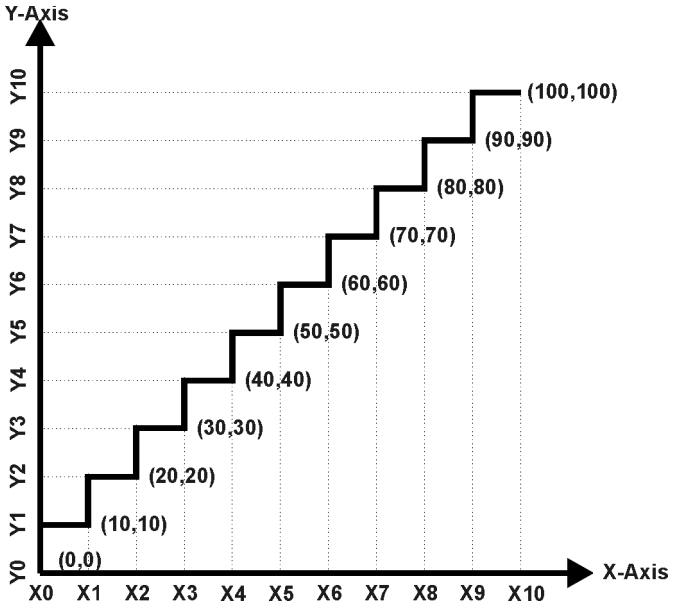


Figure 16. Lookup Table "Jump To" Data Response

Lastly, any point except (0,0) can be selected for an *'Ignore'* response. If "**Point N – Response**" is set to ignore, then all points from (X_N, Y_N) to (X_{10}, Y_{10}) will also be ignored. For all data greater than X_{N-1}, the output from the Lookup Table function block will be Y_{N-1}.

A combination of *'Ramp To'*, *'Jump To'* and *'Ignore'* responses can be used to create an application specific output profile. An example of where the same input (i.e. a CAN Message) is used as the X-Axis for two tables, but where the output profiles 'mirror' each other for a deadband joystick response is shown in the figure below. The example shows a dual slope output response for each side of the deadband, but additional slopes can be easily added as needed.

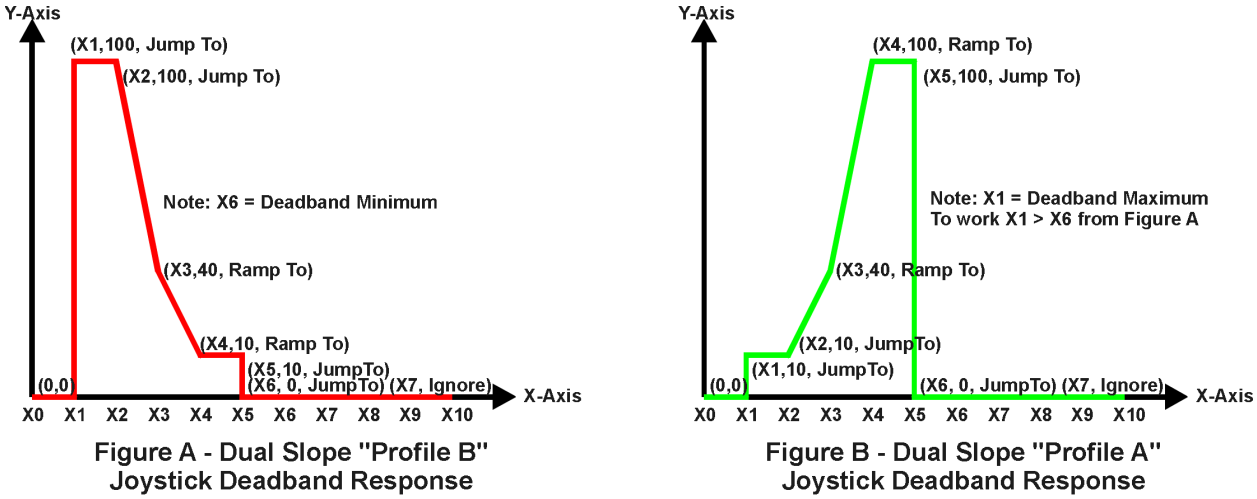


Figure 17. Lookup Table Examples to Setup for Joystick Deadband Response

3.4.5 X-Axis, Time Response

As mentioned in Section 3.4, a Lookup Table can also be used to get a custom output response where the "**X-Axis Type**" is a *'Time Response.'* When this is selected, the X-Axis now represents time, in units of milliseconds, while the Y-Axis still represents the output of the function block.

In this case, the **"X-Axis Source"** is treated as a digital input. If the signal is actually an analog input, it is interpreted like a digital input per Figure 4. When the control input is ON, the output will be changed over a period of time based on the profile in the Lookup Table. Once the profile has finished (i.e. index 10, or *'Ignored'* response), the output will remain at the last output at the end of the profile until the control input turns OFF.

When the control input is OFF, the output is always at zero. When the input comes ON, the profile ALWAYS starts at position (X_0, Y_0) which is 0 output for 0ms.

When using the Lookup Table to drive an output based on **time**, it is mandatory that setpoints **"Ramp Up (min to max)"** and **"Ramp Down (max to min)"** in the PWM Output function block be set to **zero**. Otherwise, the output result will not match the profile as expected. Recall, also, that the Y-Axis range of the table should be set to match the Analog Output 1 range in order to get a 1:1 response of table output versus drive output.

An application where this feature would be useful is filling a clutch when a transmission is engaged. An example of some fill profiles is shown in the figure below.

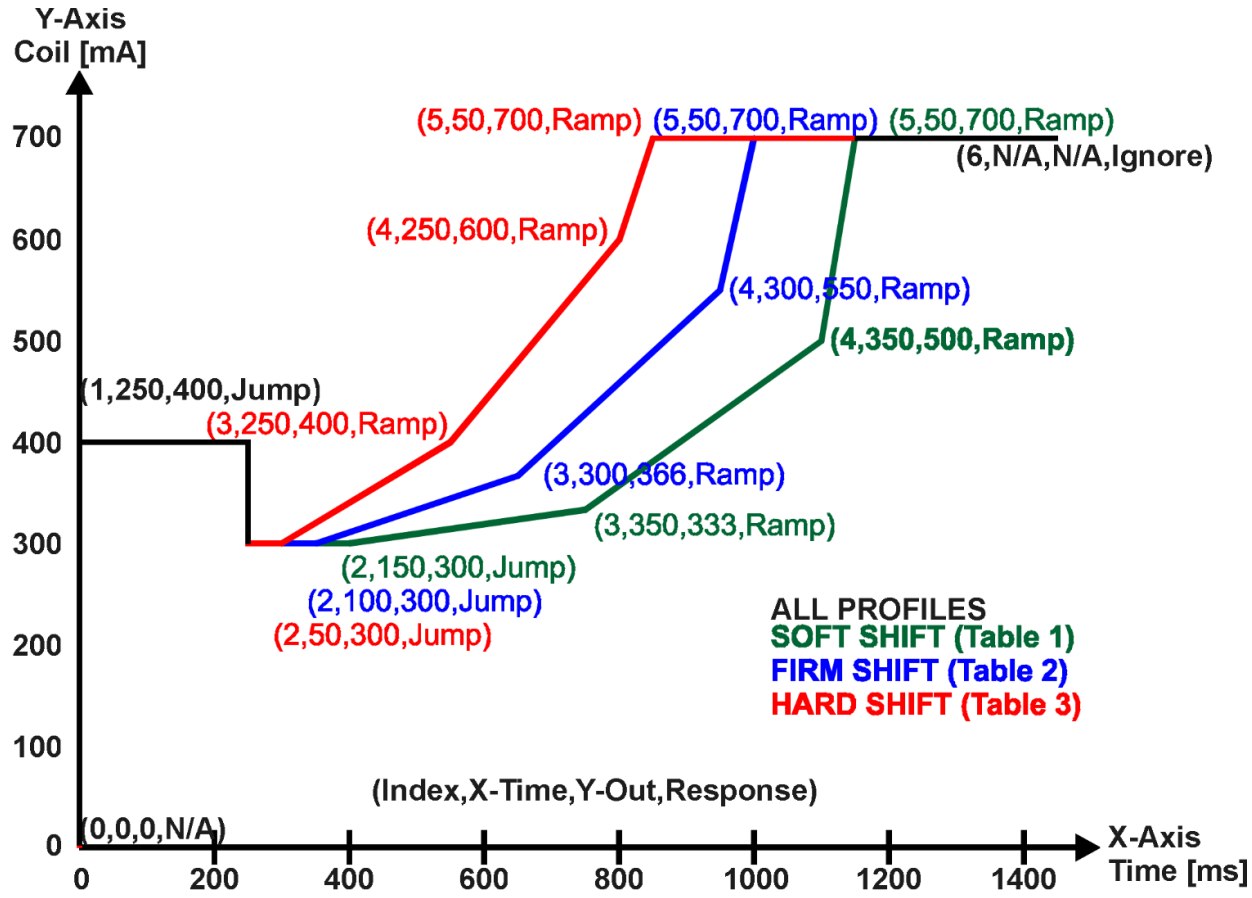


Figure 18. Lookup Table Time Response Clutch Fill Profiles

In a time response, the interval time between each point on the X-axis can be set anywhere from 1ms to 24 hours. [86,400,000 ms]

One final note about the Lookup Tables is that if a digital input is selected as the control source for the X-Axis, only a 0 (Off) or 1 (On) will be measured. Ensure that the data range for the X-Axis on the table is updated appropriately in this condition.

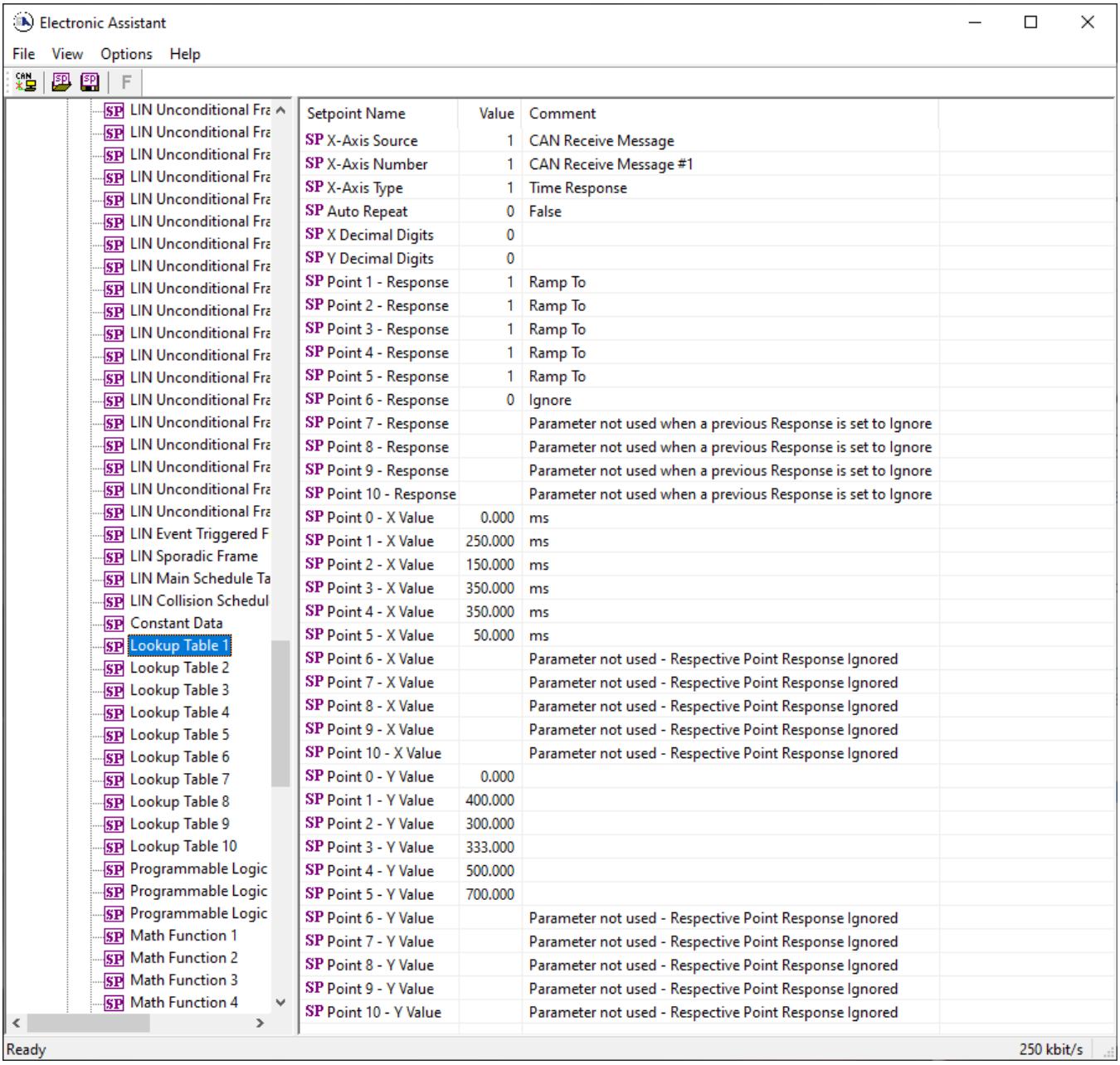


Figure 19. Lookup Table "Soft Shift" Axiomatic EA Configuration

3.5 Programmable Logic Function Block

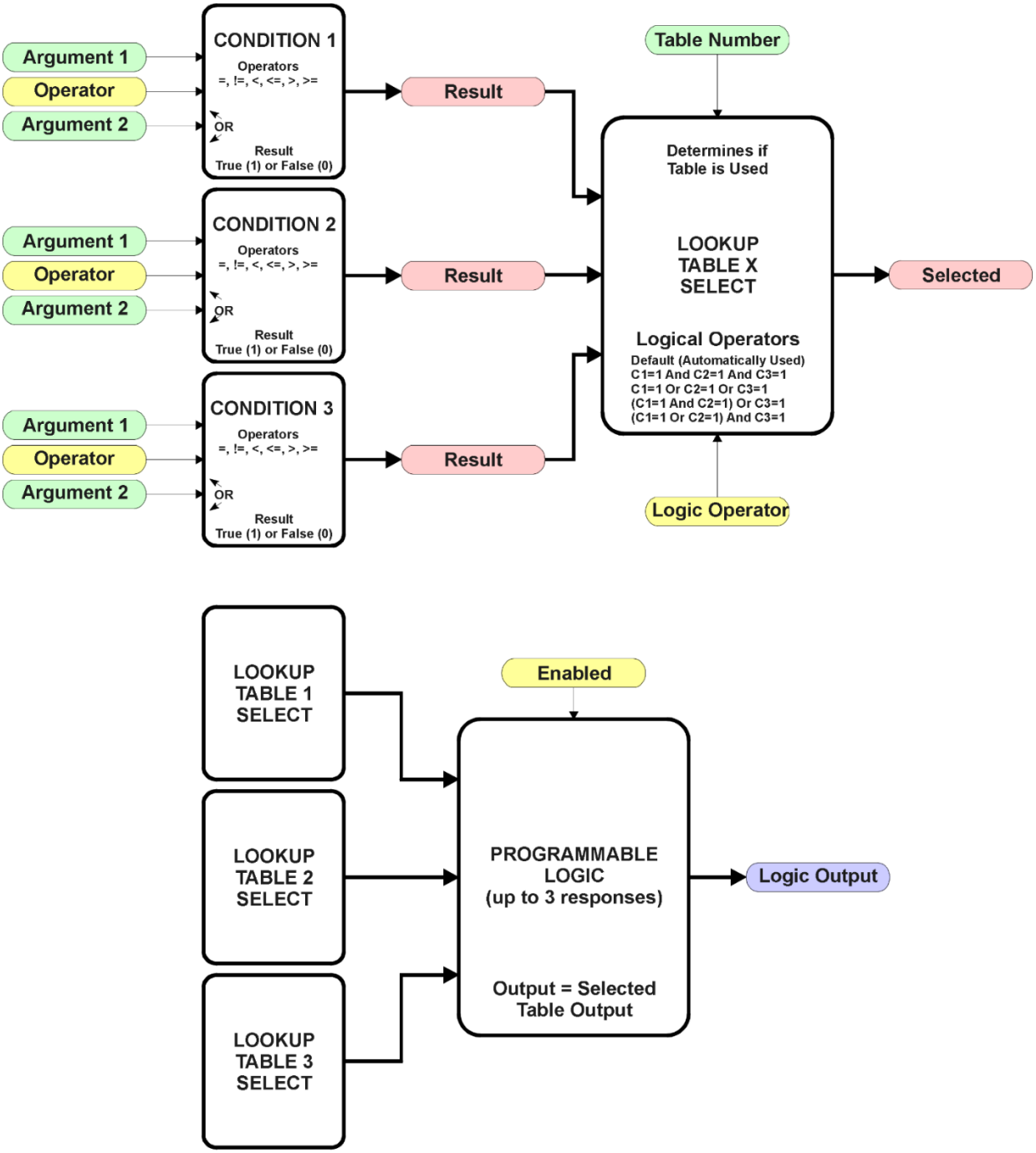


Figure 20. Programmable Logic Function Block

This function block is obviously the most complicated of them all, but very powerful. The Programmable Logic can be linked to up to three tables, any one of which would be selected only under given conditions. Any three tables (of the available 10) can be associated with the logic, and which ones are used is fully configurable.

Should the conditions be such that a particular table (1, 2 or 3) has been selected as described in Section 3.5.2, then the output from the selected table, at any given time, will be passed directly to the Logic Output.

Therefore, up to three different responses to the same input, or three different responses to different inputs, can become the input to another function block, such as PWM Output. To do this, the "**Control Source**" for the reactive block would be selected to be the *'Programmable Logic Function Block.'*

In order to enable any one of Programmable Logic blocks, the "**Programmable Logic Block Enabled**" setpoint must be set to *'True'*. They are all disabled by default.

Logic is evaluated in the order shown in the figure below. Only if a lower number table has not been selected will the conditions for the next table be looked at. **The default table is always selected as soon as it is evaluated. It is therefore required that the default table always be the highest number in any configuration.**

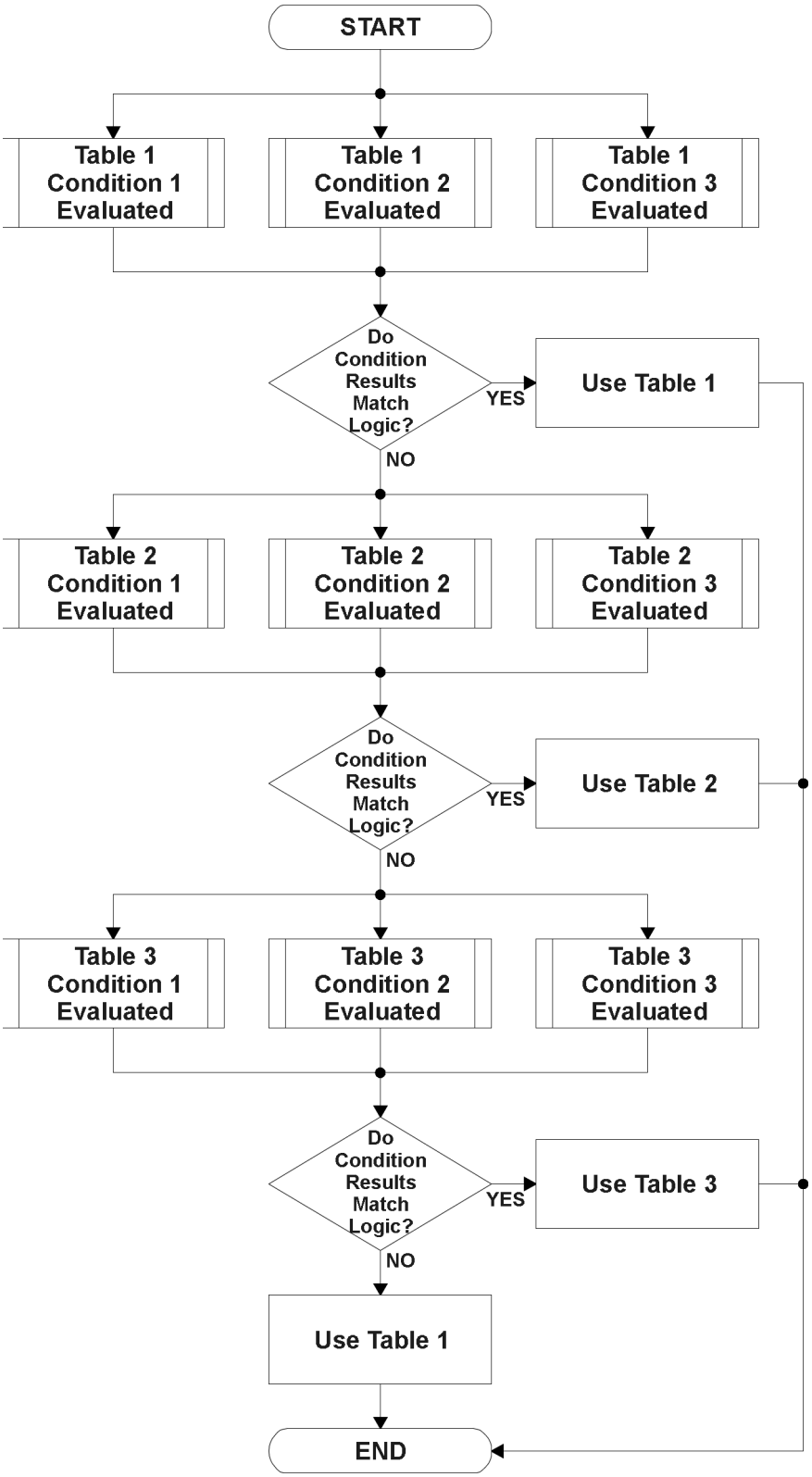


Figure 21. Programmable Logic Flowchart

3.5.1 Conditions Evaluation

The first step in determining which table will be selected as the active table is to first evaluate the conditions associated with a given table. Each table has associated with it up to three conditions that can be evaluated.

Argument Z is always a logical output from another function block. As always, the source is a combination of the functional block type and number, setpoints **"Table X - Condition Y, Argument 1 Source"** and **"Table X, Condition Y, Argument 1 Number"**, where both X = 1 to 3 and Y = 1 to 3.

The condition is evaluated based on the **"Table X, Condition Y Operator"** selected by the user. It is always *'=, Equal'* by default. The only way to change this is to have two valid arguments selected for any given condition. Options for the operator are listed in the table below.

Table 21. Condition Operator Options

Value	Meaning
0	*=, Equal*
1	*!=, Not Equal*
2	*>, Greater Than*
3	*>=, Greater Than or Equal*
4	*<, Less Than*
5	*<=, Less Than or Equal*

For example, a condition for a transmission control shift selection, as shown in Figure 18 in the previous section, could be that the Engine RPM received on CAN message 3 be less than a certain value to select a Soft Fill profile. In this case, **"…Argument 1 Source"** would be set to *'CAN Receive Message 3'*, **"…Argument 2 Source"** to *'Constant Continuous Data 1*, and the **"…Operator"** to *'<, Less Than.'*

By default, both arguments are set to *'Control Source Not Used'* which disables the condition, and automatically results in a value of N/A as the result. Although Figure 20 shows only True or False as a result of a condition evaluation, the reality is that there could be four possible results, as described in the table below.

Table 22. Condition Evaluation Results

Value	Meaning	Reason
0	False	(Argument 1) Operator (Argument 2) = False
1	True	(Argument 1) Operator (Argument 2) = True
2	Error	Argument 1 or 2 output was reported as being in an error state
3	Not Applicable	Argument 1 or 2 is not available (i.e. set to *'Control Source Not Used'*)

3.5.2 Table Selection

In order to determine if a particular table will be selected, logical operations are performed on the results of the conditions as determined by the logic in Section 3.5. There are several logical combinations that can be selected, as listed in the table below.

Table 23. Condition Logical Operator Options

Value	Meaning
0	*Default Table*
1	*Cnd1 And Cnd2 And Cnd3*
2	*Cnd1 Or Cnd2 Or Cnd3*
3	*(Cnd1 And Cnd2) Or Cnd3*
4	*(Cnd1 Or Cnd2) And Cnd3*

Not every evaluation is going to need all three conditions. The case given in the earlier section, for example, only has one condition listed, i.e. that the Engine RPM be below a certain value. Therefore, it is important to understand how the logical operators would evaluate an Error or N/A result for a condition.

Table 24. Conditions Evaluation Based on Selected Logical Operator

Logical Operator	Select Conditions Criteria
Default Table	Associated table is automatically selected as soon as it is evaluated.
Cnd1 And Cnd2 And Cnd3	**Should be used when two or three conditions are relevant, and all must be true to select the table.** If any condition equals False or Error, the table is not selected. An N/A is treated like a True. If all three conditions are True (or N/A), the table is selected. If((Cnd1==True) &&(Cnd2==True)&&(Cnd3==True)) Then Use Table
Cnd1 Or Cnd2 Or Cnd3	**Should be used when only one condition is relevant. Can also be used with two or three relevant conditions.** If any condition is evaluated as True, the table is selected. Error or N/A results are treated as False If((Cnd1==True) || (Cnd2==True) || (Cnd3==True)) Then Use Table
(Cnd1 And Cnd2) Or Cnd3	**To be used only when all three conditions are relevant.** If both Condition 1 and Condition 2 are True, OR Condition 3 is True, the table is selected. Error or N/A results are treated as False If(((Cnd1==True)&&(Cnd2==True)) || (Cnd3==True)) Then Use Table
(Cnd1 Or Cnd2) And Cnd3	**To be used only when all three conditions are relevant.** If Condition 1 And Condition 3 are True, OR Condition 2 And Condition 3 are True, the table is selected. Error or N/A results are treated as False If(((Cnd1==True)||(Cnd2==True)) && (Cnd3==True)) Then Use Table

The default **"Table X, Conditions Logical Operator"** for Table 1 and Table 2 is *'Cnd1 And Cnd2 And Cnd3,'* while Table 3 is set to be the *'Default Table.'*

3.5.3 Logic Block Output

Recall that Table X, where X = 1 to 3 in the Programmable Logic function block does NOT mean Lookup Table 1 to 3. Each table has a setpoint **"Table X – Lookup Table Block Number"** which allows the user to select which Lookup Tables they want associated with a particular Programmable Logic Block. The default tables associated with each logic block are listed in the table below.

Table 25. Programmable Logic Block Default Lookup Tables

Programmable Logic Block Number	Table 1 – Lookup Table Block Number	Table 2 – Lookup Table Block Number	Table 3 – Lookup Table Block Number
1	1	2	3
2	4	5	6
3	7	8	9

If the associated Lookup Table does not have an **"X-Axis Source"** selected, then the output of the Programmable Logic block will always be "Not Available" so long as that table is selected. However, should the Lookup Table be configured for a valid response to an input, be it Data or Time, the output of the Lookup Table function block (i.e. the Y-Axis data that has been selected based on the X-Axis value) will become the output of the Programmable Logic function block so long as that table is selected.

Unlike all other function blocks, the Programmable Logic does NOT perform any linearization calculations between the input and the output data. Instead, it mirrors exactly the input (Lookup Table) data. Therefore, when using the Programmable Logic as a control source for another function block, it is HIGHLY recommended that all the associated Lookup Table Y-Axes either be (a) Set between the 0 to 100% output range or (b) all set to the same scale.

3.6 Math Function Block

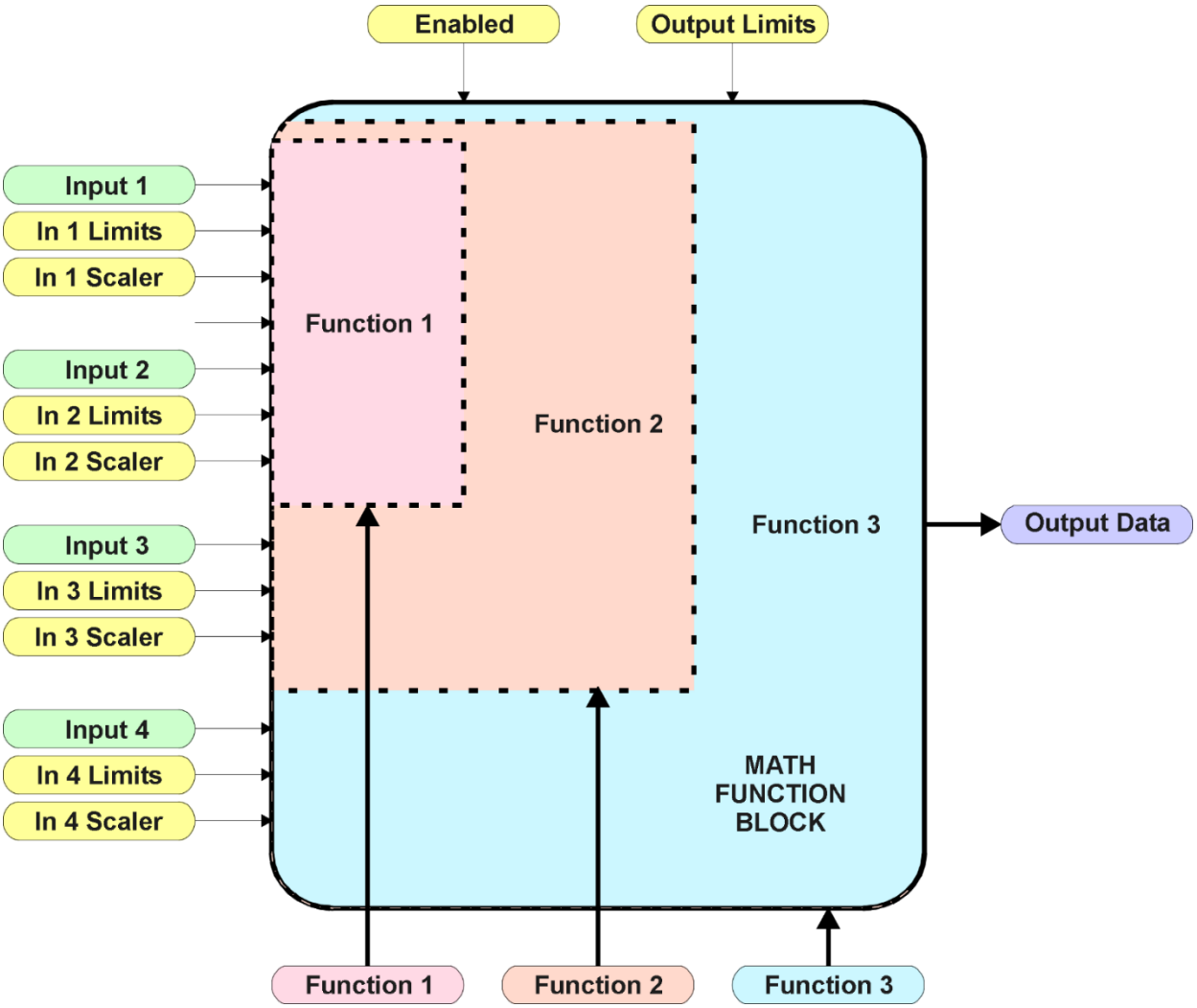


Figure 22. Programmable Logic Flowchart

There are four mathematic function blocks that allow the user to define basic algorithms. A math function block can take up to four input signals, as listed in Table 8 in Section 3.2.4. Each input is then scaled according the associated limit and scaling setpoints.

Inputs are converted into a percentage value based on the **"Math Input X Minimum"** and **"Math Input X Maximum"** values selected, where X = 1 to 4. For additional control, the user can also adjust the **"Math Input X Scaler."** By default, each input has a scaling 'weight' of 1.0. However, each input can be scaled from -1.0 to 1.0 as necessary before it is applied in the function.

For example, in the case where the user may want to combine two inputs such that a joystick (Input 1) is the primary control of an output, but the speed can be incremented or decremented based on a potentiometer (Input 2), it may be desired that 75% of the scale is controlled by the joystick position, while the potentiometer can increase or decrease the min/max output by up to 25%. In this case, Input 1 would be scaled with 0.75, while Input 2 uses 0.25. The resulting addition will give a command from 0 to 100% based on the combined positions of both inputs.

The appropriate arithmetic or logical operation is performed on the two inputs, InA and InB, according the associated function. The list of selectable function operations is defined in the table below.

Table 26. Math Function Operators

Value	Meaning	Notes
0	=	True when InA Equals InB
1	!=	True when InA Not Equal InB
2	>	True when InA Greater Than InB
3	>=	True when InA Greater Than or Equal InB
4	<	True when InA Less Than InB
5	<=	True when InA Less Than or Equal InB
6	OR	True when InA or InB is True
7	AND	True when InA and InB are True
8	XOR	True when InA/InB is True, but not both
9	+	Result = InA plus InB
10	-	Result = InA minus InB
11	x	Result = InA times InB
12	/	Result = InA divided by InB
13	MIN	Result = Smallest of InA and InB
14	MAX	Result = Largest of InA and InB

For Function 1, InA and InB are Inputs 1 and 2 respectively.
For Function 2, InA is the result of Function 1, and InB is Input 3.
For Function 3, InA is the result of Function 2, and InB is Input 4.

Exclusively **within** a Math Block, there is a third control parameter: "**Math Input X Function Number**". This parameter allows for the result of any Function (1, 2 or 3) to be the input to any Math Input Y within the same Math Block. Therefore, "**Math Input X Source**" must be a Math Block and "**Math Y Input Number**" must be the same number as being configured. When these three parameters match, if "**Math Input X Function Number**" is set to 1, 2, or 3, the respective input will be the result of the Function selected. By default, it is set to 0 – in which case this parameter is ignored and uses the Math Block output result. These functions can only be used **within** the Math Block. They cannot be used for other Math Blocks or logic Blocks.

For a valid result, the control source for an input must be a non-zero value, i.e. something other than *'Control Source Not Used.'* Otherwise, the corresponding function is ignored, and the "Output Data" for the math function block is the result of the earlier function scaled according to the output limit setpoints. For example, if Input 4 is not used, the math output would be the result of the Function 2 operation.

For logical operators (6, 7 or 8), any SCALED input greater than or equal to 0.5 is treated as a TRUE input. For logic output operators (0 to 8), the result of the calculation for the function will always be 0 (FALSE) or 1 (TRUE).

Error data (i.e. input measured out of range) is always treated as a 0.0 input into the function.

For the arithmetic functions (9 to 14), it is recommended to scale the data such that the resulting operation will not exceed full scale (0 to 100%) and saturate the output result.

When dividing, a zero InB value will always result is a zero output value for the associated function. When subtracting, a negative result will always be treated as a zero, unless the function is multiplied by a negative one, or the inputs are scaled with a negative coefficient first.

The resulting mathematical calculation, represented as a percentage value, can be scaled into the appropriate physical units using the **"Math Output Minimum Range"** and **"Math Output Maximum Range"** setpoints. These values are also used as the limits when the Math Function is selected as the input source for another function block.

Table 27. Global Parameters Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Math Function Enabled	0, False	Drop List	–	True or False
Math Output Minimum Range	0.0	[-10^4 …10^4]	–	Converts calculation from a percentage value to the desired physical unit.
Math Output Maximum Range	100.0	[-10^4 …10^4]	–	
Math Decimal Digits	0	[0…4]	–	Change the resolution of range values
Function X Input Y Source (X = 1 to 3, Y = A or B)	0, Control Source Not Used	Drop List	–	See Table 8.
Function X Input Y Number	0	Per Source	–	See Table 8.
Function X Input Y Decimal Digits	0	[0…4]	–	Change the resolution of range values
Function X Input Y Function Number	0	[0…3]	–	See previous section
Function X Input Y Minimum	0	[-10^4 …10^4]	–	Converts input to a percentage before use in the calculation.
Function X Input Y Maximum	0	[-10^4 …10^4]	–	
Function X Input Y Scaler	1.00	[-10^4 …10^4]		See previous section

3.7 Constant Data

The *Constant Data* functional block gives the user access to a set of global constants. The function block has five configurable *Constant Discrete Data* outputs, five configurable *Constant Continuous Data* outputs.

Configuration parameters of the *Constant Data* function block are presented below:

Table 28. Global Parameters Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Global Discrete Constant Signal	0	[0…0xFFFFFFFF]	–	Discrete constant signal.
Global Discrete Constant Signal	0	[0…0xFFFFFFFF]	–	Discrete constant signal.
Global Discrete Constant Signal	0	[0…0xFFFFFFFF]	–	Discrete constant signal.
Global Discrete Constant Signal	0	[0…0xFFFFFFFF]	–	Discrete constant signal.
Global Discrete Constant Signal	0	[0…0xFFFFFFFF]	–	Discrete constant signal.
Global Continuous Constant Signal	0	Any value	–	Continuous constant signal.
Global Continuous Constant Signal	0	Any value	–	Continuous constant signal.
Global Continuous Constant Signal	0	Any value	–	Continuous constant signal.
Global Continuous Constant Signal	0	Any value	–	Continuous constant signal.
Global Continuous Constant Signal	0	Any value	–	Continuous constant signal.

3.8 CAN Interface

The converter CAN interface functionality is defined by *Miscellaneous*, *CAN Receive, CAN Transmit* function blocks.

CAN signals are received from the CAN bus by the *CAN Receive* function blocks. CAN signals are transmitted on the CAN bus in the CAN single frame messages defined by the *CAN Transmit* function blocks. The *Miscellaneous* function block contains the global CAN bus configuration settings.

The function blocks will be presented in the order they appear in the Axiomatic EA.

3.8.1 Miscellaneous

The *Miscellaneous* function block defines the global J1939 CAN bus settings. It does not have signal inputs and outputs.

Figure 23. J1939 Network Function Block

Configuration parameters of the *Miscellaneous* function block are presented below. They contain *ECU Network* and *CAN Network Parameters*.

Table 29. J1939 Network Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
ECU Instance Number	0	[0…7]	–	ECU Instance field of the J1939 ECU Name.

Name	Default Value	Range	Units	Description
ECU Address	128	[0…253]	–	J1939 ECU address.
Automatic Baud Rate Detection	True	{False, True}	–	Set to *False* once ECU is permanently installed on the CAN network.
Baud Rate	–	{250, 500, 667, 1000}	kbit/s	Read only parameter. Current baud rate on the CAN network.

3.8.1.1 ECU Network Parameters

The user can change the *ECU Instance Number* and *ECU Address* to adjust the unit on the CAN network.

Changing the *ECU Instance Number* is necessary to accommodate multiple converters on the same CAN network. The list of available ECU instances is shown in the *ECU Instance Number Setup* dialog window in the Axiomatic EA. The user should select the required ECU instance number and then press OK or, starting from EA 5.14.103.0, double-click the selected instance number.

The *ECU Address* is automatically adjusted as the result of an address arbitration process on the J1939 CAN network. It can also be changed by a commanded address message. The user can also manually change the ECU address using the *ECU Address* configuration parameter.

The user selects the new ECU address from the list of available ECU addresses in the *ECU Address Setup* dialog window similar to the ECU instance number setup dialog. After the required ECU address is selected, the user should press OK button or, starting from EA 5.14.103.0, double-click the selected address.

3.8.1.2 CAN Network Parameters

The *Baud Rate* configuration parameter shows the current baud rate on the CAN network.

The *Automatic Baud Rate Detection* parameter defines whether the ECU will try to detect the CAN baud rate in case of communication errors. The baud rate is detected from the list of supported CAN baud rates.

To avoid an arbitrary selection of the CAN baud rate by ECUs involved in the automatic baud rate detection process, it is necessary to disable the automatic baud rate detection in ECUs that are already permanently installed on the CAN network.

3.8.2 CAN Receive

There are 25 *CAN Receive* function blocks available to the user. Each function block represents one CAN input signal that can be received from the CAN bus. The function block has one signal output.

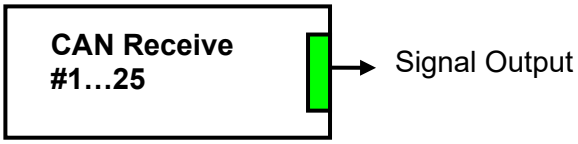


Figure 24. CAN Receive *Function Block*

The *CAN Receive* function block reads single-frame application specific CAN messages and extracts CAN signal data presented in user-defined data format. Different *CAN Receive* function blocks can read and process the same CAN message to extract different CAN signal data.

The CAN messages transmitted by the converter itself are also processed by *CAN Receive* function blocks. 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 the state of the bus.

Configuration parameters of the *CAN Receive* function block are presented below:

Table 30. CAN Receive Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Received Signal Type	CAN Receive 1- Continuous Others - Undefined	{Undefined, Discrete, Continuous}	–	J1939 Signal type
Receive PGN	0xFF80+X[1]	Any J1939 PGN value[2]	–	Signal message PGN value.
Enable Specific Address	False	{False, True}	–	Only CAN messages from the selected address will be accepted, if “Yes”.
Specific Address that sends PGN	0	[0; 253]	–	Address of the ECU transmitting CAN messages if PGN From Selected Address is set to “Yes”.
Received message Timeout	500	[0; 60000]	ms	Function block signal output timeout value. If Autoreset Time is 0, auto-reset is disabled.
Received Data Size	0	[1…32]	bit	CAN input signal size.
Received Data Index in Array (LSB)	0	[0; 7]	–	Start byte of the CAN input signal in the CAN message data frame.
Received Bit Index in Byte (LSB)	0	[0; 7]	–	Start bit of the CAN input signal in the Data Position Byte.
Received Data Resolution	1	Any value	Signal Units / bit	CAN input signal resolution for continuous input signals.
Received Data Offset	0	Any value	Signal Units	CAN input signal offset for continuous input signals.
Received Data Min (Off Threshold)	0	[-100000; 100000]	Signal Units	The minimum value of the X-axis used in the linear calculations. As the name imply, it is also used as the OFF thresholds for digital input types.

Name	Default Value	Range	Units	Description
Received Data Max (On Threshold)	100.0	[-100000; 100000]	Signal Units	The maximum value of the X-axis used in the linear calculations. As the name imply, it is also used as the OFF thresholds for digital input types.

[1]X is defined by the Block Number, ex. X=1 for CAN Receive 1
[2]Proprietary A PGN (61184) is excluded. It is taken by Axiomatic Simple Proprietary Protocol and therefore cannot be used in function blocks.

The CAN input signal position is defined within the CAN message data frame by the *Received Data Index in Array (LSB)* and *Received Bit Index in Byte (LSB)* configuration parameters the same way as in the J1939 standard. The start and stop bits of the CAN signal in the 64-bit CAN message data frame are calculated using the formulae:

$$\text{StartBit} = (\text{DataPositionByte} - 1) \cdot 8 + (\text{DataPositionBit} - 1), \quad (2)$$

$$StopBit = StartBit + Size - 1, where: StartBit, StopBit \in [0 \dots 63].$$

Resolution and *Offset* configuration parameters are set for continuous CAN input signals. They are not used with discrete CAN signals. *Received Data min* and *Received Data max* values are in whatever units the output data is AFTER the resolution and offset is applied to the CAN data.

The following rules apply when converting the CAN signal data to the function block output signal:

- It is assumed that CAN signal code with all bits set to 1 represents an undefined signal;
- Discrete signals can take any value except the one reserved for the undefined signal;
- Continuous signals can take only values from the range reserved for continuous signals in the J1939 standard. If the CAN signal code is outside of the range reserved for the continuous signal, the signal is ignored.

When the *Received Message Timeout* is not equal to 0, if the message is not received off the bud within the time set, the message data is zeroed and it will flag on signal in question. If enabled, Lost Communication fault will be flagged as well, which could trigger a Lost Communication event as described in section 3.9. In order to avoid timeouts on a heavily saturated network, it is recommended to set the period at least three times longer than the expected update rate. To disable the timeout feature, simply set this value to zero, in which case the received message will never trigger a Lost Communication fault.

3.8.3 CAN Transmit

There are *25 CAN Transmit* function blocks available to the user. Each function block represents one single frame CAN output message that can be sent on the CAN bus.

The message contains up to 5 CAN output signals. In case more signals are required, the user can merge two or more CAN output messages with the same PGN in one CAN output message.

Each CAN output signal is presented by its signal input in the function block.

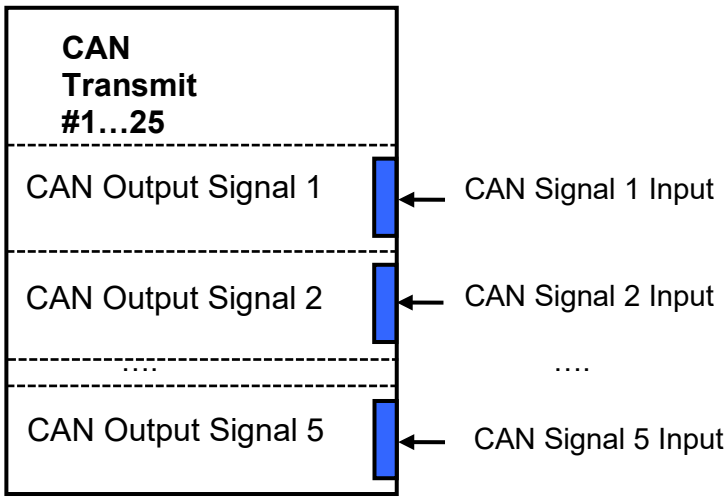


Figure 25. CAN Output Message Function Block

Configuration parameters of the *CAN Output Message* function block are presented below:

Table 31. CAN Transmit Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Transmit PGN	0xFF00+X[1]	Any J1939 PGN value[2]	–	CAN message PGN.
Transmission Enabled	False	{False, True}	–	Enables the CAN output message transmission.
Transmission Repetition Rate	1000	[0;65535]	ms	CAN output message transmission rate. If 0 – transmission is upon request.
Destination Address	0xFF	[0; 255]	–	Destination address of the PDU1 PGN messages.
Length	0	[0…8]	–	CAN message data frame length.
Priority	6	[0…7]	–	CAN message priority.
Transmit on LIN Unconditional Frame Number	0	[0…25]	–	Transmit CAN message on successful transmission or reception of the LIN unconditional frame number.
Signal 1 Data Type	Undefined	{Undefined, Discrete, Continuous}	–	Type of the 1-st CAN output signal.
Signal 1 Data Source	Control Not Used	Any signal output of any function block or “Control Not Used”	–	Input signal source of the 1-st CAN output signal.
Signal 1 Data Number	0	Depends on the data source chosen	–	Input signal number of the 1-st CAN output signal
Signal 1 Size	1	[1…32]	bit	Size of the 1-st CAN output signal.

Name	Default Value	Range	Units	Description
Signal 1 Data Index in Array (LSB)	1	[1…8]	–	Byte position of the 1-st CAN output signal.
Signal 1 Bit Index in Byte (LSB)	1	[1…8]	–	Bit position of the 1-st CAN output signal.
Signal 1 Data Resolution	1	Any value	Signal Units / bit	Resolution of the 1-st CAN continuous output signal.
Signal 1 Data Offset	0	Any value	Signal Units	Offset of the 1-st CAN continuous output signal.
Signal 2 Data Type	Undefined	{Undefined, Discrete, Continuous}	–	Type of the 2-nd CAN output signal.
Signal 2 Data Source	Control Not Used	Any signal output of any function block or “Control Not Used”	–	Input signal source of the 2-nd CAN output signal.
Signal 2 Data Number	0	Depends on the data source chosen	–	Input signal number of the 2-nd CAN output signal
Signal 2 Data Size	1	[1…32]	bit	Size of the 2-nd CAN output signal.
Signal 2 Data Index in Array (LSB)	1	[1…8]	–	Byte position of the 2-nd CAN output signal.
Signal 2 Bit Index in Byte (LSB)	1	[1…8]	–	Bit position of the 2-nd CAN output signal.
Signal 2 Data Resolution	1	Any value	Signal Units / bit	Resolution of the 2-nd CAN continuous output signal.
Signal 2 Data Offset	0	Any value	Signal Units	Offset of the 2-nd CAN continuous output signal.
…	…	…	…	…
Signal 5 Data Type	Undefined	{Undefined, Discrete, Continuous}	–	Type of the 5-th CAN output signal.
Signal 5 Data Source	Control Not Used	Any signal output of any function block or “Control Not Used”	–	Input signal source of the 5-th CAN output signal.
Signal 5 Data Number	0	Depends on the data source chosen	–	Input signal number of the 5-th CAN output signal
Signal 5 Data Size	1	[1…32]	bit	Size of the 5-th CAN output signal.
Signal 5 Data Index in Array (LSB)	1	[1…8]	–	Byte position of the 5-th CAN output signal.
Signal 5 Bit Index in Byte (LSB)	1	[1…8]	–	Bit position of the 5-th CAN output signal.
Signal 5 Data Resolution	1	Any value	Signal Units / bit	Resolution of the 5-th CAN continuous output signal.
Signal 5 Data Offset	0	Any value	Signal Units	Offset of the 5-th CAN continuous output signal.

[1]X is defined by the Block Number, ex. X=1 for CAN Transmit 1
[2]Proprietary A PGN (61184) is excluded. It is taken by Axiomatic Simple Proprietary Protocol and therefore cannot be used in function blocks.

The *Transmit on LIN Unconditional Frame Number* is used to force transmission of the CAN output message on successful reception or transmission of the selected LIN unconditional frame. When *Transmit on LIN Unconditional Frame Number* is equal to 0, the frame is undefined, and this function is not used.

Configuration parameters: *Signal 1…5 Data Index in Array (LSB)* and *Signal 1…5 Bit Index in Byte (LSB),* together with the *Signal 1…5 Data Size* have the same meaning as in the *CAN Receive* function block. The user should be careful not to overlap the output signals.

The following rules apply when converting the function block signal output data to the CAN output signal code:

- Undefined signals are presented in the signal code with all bits set to 1.
- Discrete signals are directly assigned to the signal code without any conversion.
- Continuous signals are converted to the signal code based on the *Signal 1…5 Data Resolution* and *Signal 1…5 Data Offset* configuration parameters and then saturated to the continuous signal code range defined in the J1939 standard.

3.9 Diagnostic Function Block

There are several types of diagnostics supported by the converter. As described in Section 3.2.4, fault detection and reaction are associated with the output drive. In addition, it can also detect/react to power supply over/under voltage measurements, a processor over-temperature, or lost communication events.

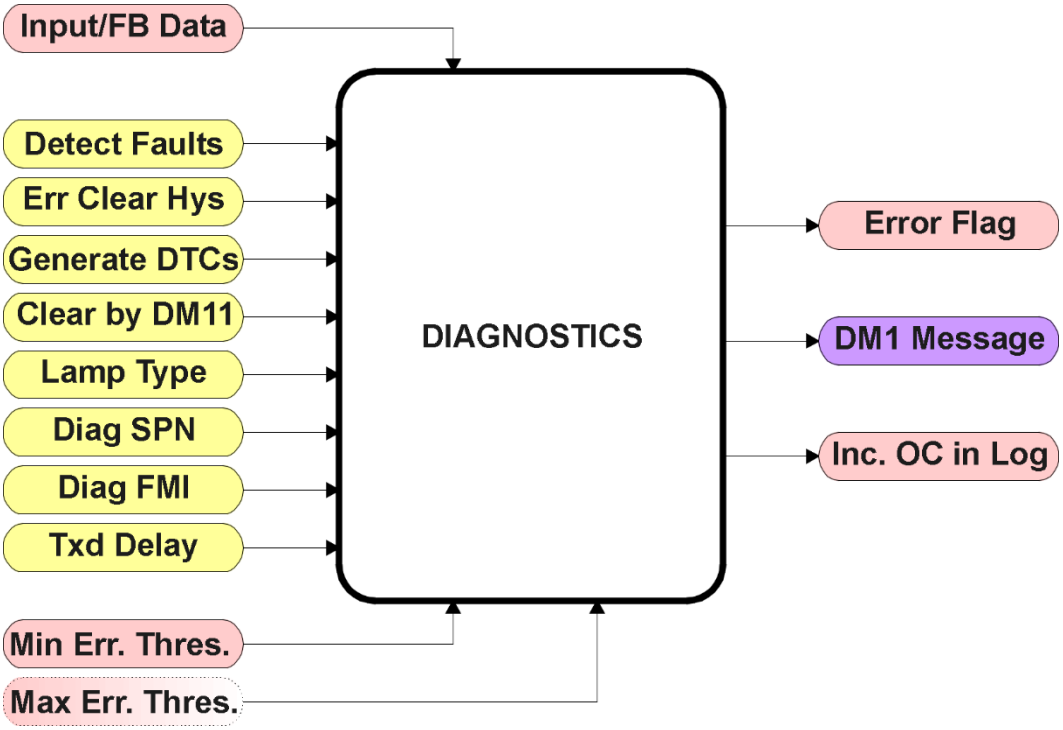


Figure 26. Diagnostics Function Block

The **"Fault Detection is Enabled"** is the most important setpoint associated with this function block, and it should be selected first. Changing it will result in other setpoints being enabled/disabled as appropriate. When disabled, all diagnostic behaviour associated with the output or event in question is ignored (i.e. this type of fault will not disable the output.)

In the case of a power supply error or temperature error, "**Fault Disables Outputs**" setpoint can be selected to disable all the outputs. By default, a power supply under/over voltage or over temperature condition will not shutoff the output.

In most cases, faults can be flagged as either a low or high occurrence. The min/max thresholds for all diagnostics supported by the converter are listed in the table below. Bolded values are user configurable setpoints. Some diagnostics react only to a single condition, in which case a N/A is listed in one of the columns.

Table 32. Fault Detect Thresholds

See Section	See Section	See Section
Output Drive	N/A	N/A
Power Supply	**Power Undervoltage Threshold**	**Power Overvoltage Threshold**

Over Temperature	N/A	**Over Temperature Shutdown**
Lost Communication	N/A	**Received Message Timeout** (any)

When applicable, a hysteresis setpoint is provided to prevent the rapid setting and clearing of the error flag when the measured value is right near the fault detection threshold. For the low end, once a fault has been flagged, it will not be cleared until the measured value is greater than or equal to the Minimum Threshold **+ "Hysteresis to Clear Fault."** For the high end, it will not be cleared until the measured value is less than or equal to the Maximum Threshold **– "Hysteresis to Clear Fault."** The minimum, maximum and hysteresis values are always measured in the units of the fault in question.

Note: It is recommended to take in consider of another block when changing the setpoints in Under Voltage Diagnostics block or Over Voltage. Changing for the Hysteresis in either block will result in the change for the other block. However, disables outputs setpoint is used for their own fault flags.

The next setpoint in these function blocks is the **"Event Generates a DTC in DM1."** If and only if this is set to true will the other setpoints in the function block be enabled. They are all related to the data that is sent to the J1939 network as part of the DM1 message, Active Diagnostic Trouble Codes.

A Diagnostic Trouble Code (DTC) is defined by the J1939 standard as a four byte value which is a combination of:

SPN	Suspect Parameter Number	(first 19 bits of the DTC, LSB first)
FMI	Failure Mode Identifier	(next 5 bits of the DTC)
CM	Conversion Method	(1 bit, always set to 0)
OC	Occurrence Count	(7 bits, number of times the fault has happened)

In addition to supporting the DM1 message, the CAN-4AOUT Controller also supports

DM2	Previously Active Diagnostic Trouble Codes	**Sent only on request**
DM3	Diagnostic Data Clear/Reset of Previously Active DTCs	**Done only on request**
DM11	Diagnostic Data Clear/Reset for Active DTCs	**Done only on request**

So long as even one Diagnostic function block has **"Event Generates a DTC in DM1"** set to True, the converter will send the DM1 message every one second, regardless of whether or not there are any active faults, as recommended by the standard. While there are no active DTCs, the converter will send the "No Active Faults" message. If a previously inactive DTC becomes active, a DM1 will be sent immediately to reflect this. As soon as the last active DTC goes inactive, it will send a DM1 indicating that there are no more active DTCs.

If there is more than one active DTC at any given time, the regular DM1 message will be sent using a multipacket Broadcast Announce Message (BAM). If the controller receives a request for a DM1 while this is true, it will send the multipacket message to the Requester Address using the Transport Protocol (TP).

At power up, the DM1 message will not be broadcasted until after a 5 second delay. This is done to prevent any power up or initialization conditions from being flagged as an active error on the network.

When the fault is linked to a DTC, a non-volatile log of the occurrence count (OC) is kept. As soon as the controller detects a new (previously inactive) fault, it will start decrementing the "**Delay Before Sending DM1**" timer for that Diagnostic function block. If the fault has remained present during the delay time, then the controller will set the DTC to active, and will increment the OC in the log. A DM1 will immediately be generated that includes the new DTC. The timer is provided so that intermittent faults do not overwhelm the network as the fault comes and goes, since a DM1 message would be sent every time the fault shows up or goes away.

The Diagnostic function block has a setpoint **"Event Cleared only by DM11."** By default, this is always set to False, which means that as soon as the condition that caused an error flag to be set goes away, the DTC is automatically made Previously Active, and is no longer included in the DM1 message. However, when this setpoint is set to True, even if the flag is cleared, the DTC will not be made inactive, so it will continue to be sent on the DM1 message. Only when a DM11 has been requested will the DTC go inactive. This feature may be useful in a system where a critical fault needs to be clearly identified as having happened, even if the conditions that caused it went away.

In addition to all the active DTCs, another part of the DM1 message is the first byte which reflects the Lamp Status. Each Diagnostic function block has the setpoint **"Lamp Set by Event in DM1"** which determines which lamp will be set in this byte while the DTC is active. The J1939 standard defines the lamps as *'Malfunction', 'Red, Stop', 'Amber, Warning'* or *'Protect'.* By default, the *'Amber, Warning'* lamp is typically the one set by any active fault.

By default, every Diagnostic function block has associated with it a proprietary SPN. However, this setpoint **"SPN for Event used in DTC"** is fully configurable by the user should they wish it to reflect a standard SPN define in J1939-71 instead. If the SPN is changed, the OC of the associate error log is automatically reset to zero.

Every Diagnostic function block also has associated with it a default FMI. The only setpoint for the user to change the FMI is **"FMI for Event used in DTC,"** even though some Diagnostic function blocks can have both high and low errors as shown in Table 30. In those cases, the FMI in the setpoint reflect that of the low end condition, and the FMI used by the high fault will be determined in the table below. If the FMI is changed, the OC of the associate error log is automatically reset to zero.

Table 33. Low Fault FMI versus High Fault FMI

FMI for Event used in DTC – Low Fault	Corresponding FMI used in DTC – High Fault
FMI=1, Data Valid But Below Normal Operational Range – Most Severe Level	FMI=0, Data Valid But Above Normal Operational Range – Most Severe Level
FMI=4, Voltage Below Normal, Or Shorted To Low Source	FMI=3, Voltage Above Normal, Or Shorted To High Source

FMI=5, Current Below Normal Or Open Circuit	FMI=6, Current Above Normal Or Grounded Circuit
FMI=17, Data Valid But Below Normal Operating Range – Least Severe Level	FMI=15, Data Valid But Above Normal Operating Range – Least Severe Level
FMI=18, Data Valid But Below Normal Operating Range – Moderately Severe Level	FMI=16, Data Valid But Above Normal Operating Range – Moderately Severe Level
FMI=21, Data Drifted Low	FMI=20, Data Drifted High

If the FMI used is anything other than one of those in the table above, then both the low and high faults will be assigned the same FMI. This condition should be avoided, as the log will still used different OC for the two types of faults, even though they will be reported the same in the DTC. It is the user's responsibility to make sure this does not happen.

4 CONFIGURATION PARAMETERS

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

4.1 Axiomatic Electronic Assistant Software

Axiomatic provides PC-based Electronic Assistant (EA) software to communicate with a wide range of Axiomatic products, including this converter. The software can be downloaded from Axiomatic website www.axiomatic.com.

The Axiomatic EA uses the Axiomatic USB-CAN converter P/N AX070501 to connect to the CAN network. The converter with cables can be ordered as an Axiomatic EA KIT, P/Ns: AX070502 or AX070506K.

Please, refer to the user manual UMAX07050X for description of the EA and associated products, and for the CAN network connection troubleshooting.

The most recent EA software version can be downloaded from Axiomatic website.

Before connecting to the CAN network, the user should ensure that the EA baud rate is the same as the baud rate used by ECUs on the network. The EA baud rate is displayed in the bottom-right corner of the EA screen and can be changed in the *Options* menu.

If the converter is the only one ECU on a temporary network set for configuring the unit, the EA baud rate should be set to the baud rate of the CAN network where the converter is planned to be deployed. This baud rate will be stored in the ECU non-volatile memory and used by the unit on the next power-up.

Upon connection, the EA will show the converter on the list of ECUs that are present on the J1939 CAN network. If the converter is the only one ECU on the network, the following screen will appear, see the figure below.

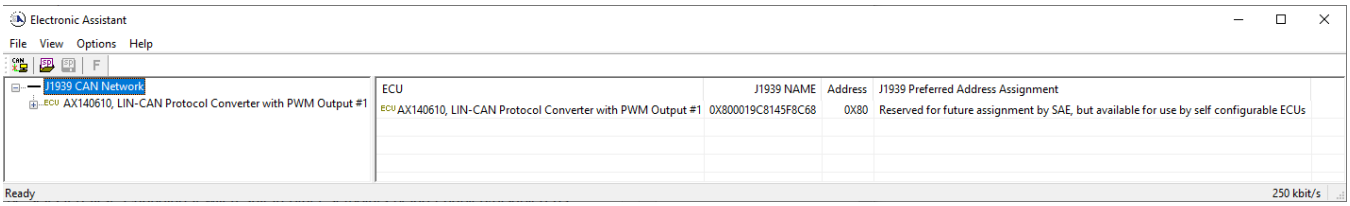


Figure 27. LIN - J1939 CAN Protocol Converter with PWM Output in the Axiomatic EA

The user can then browse through the ECU parameters, read *General ECU Information* and *Bootloader Information* groups, view and modify configuration parameters, see Figure 28 figure below.

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

In the *General ECU Information* group, the user will see the version number of the application firmware. Please, make sure that the user manual version number matches with the most

significant part of the application firmware version number. Otherwise, a different user manual is required to work with this converter.

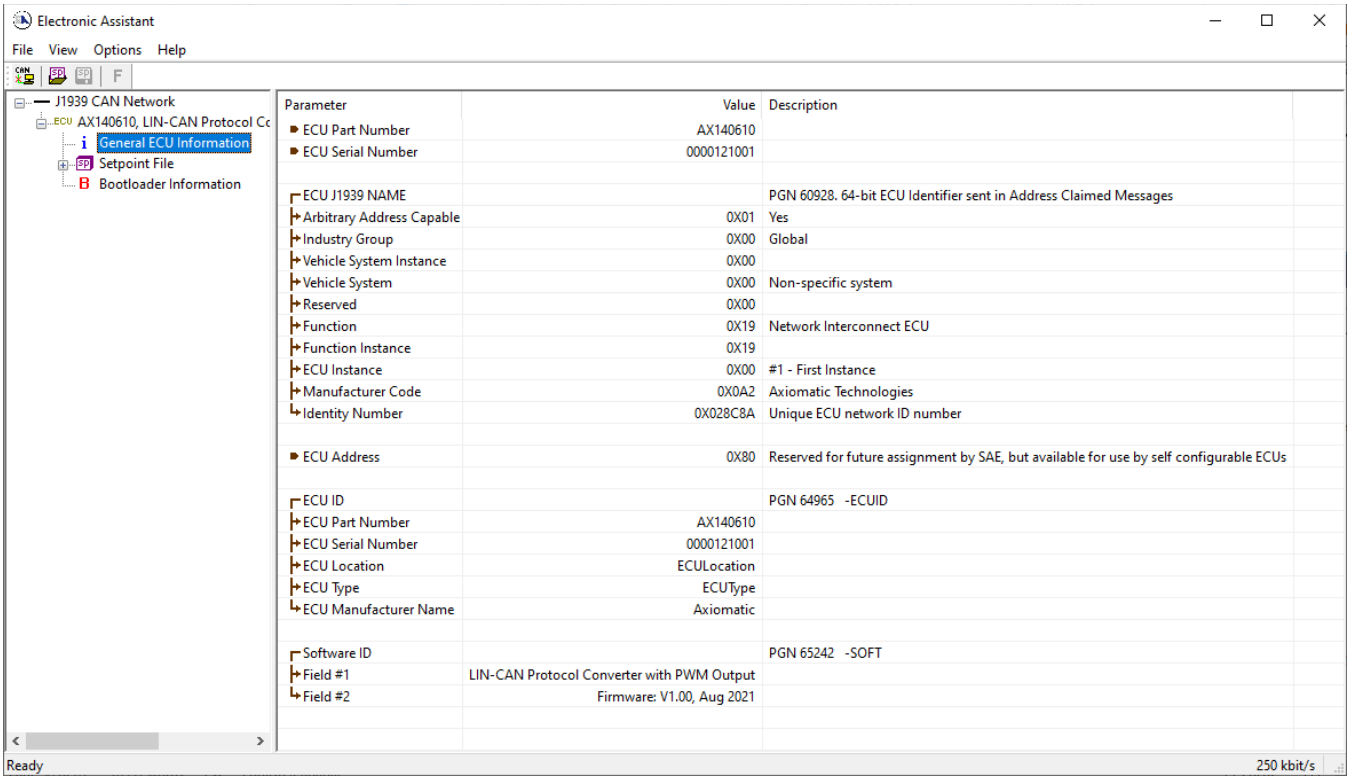


Figure 28. General ECU Information Screen

4.2 Function blocks in the Axiomatic EA

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

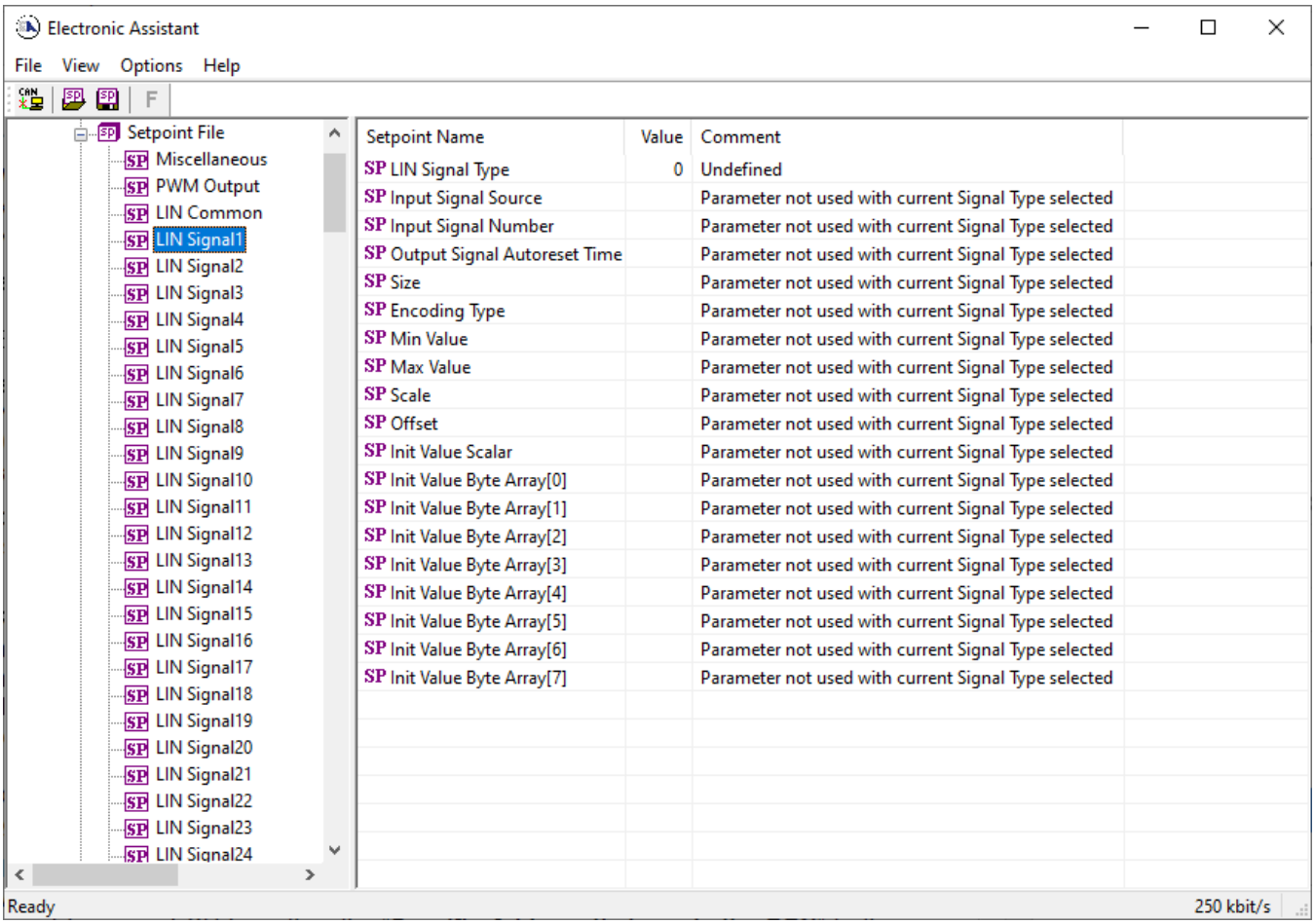


Figure 29. LIN Signal #1 Function Block in the Axiomatic EA

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

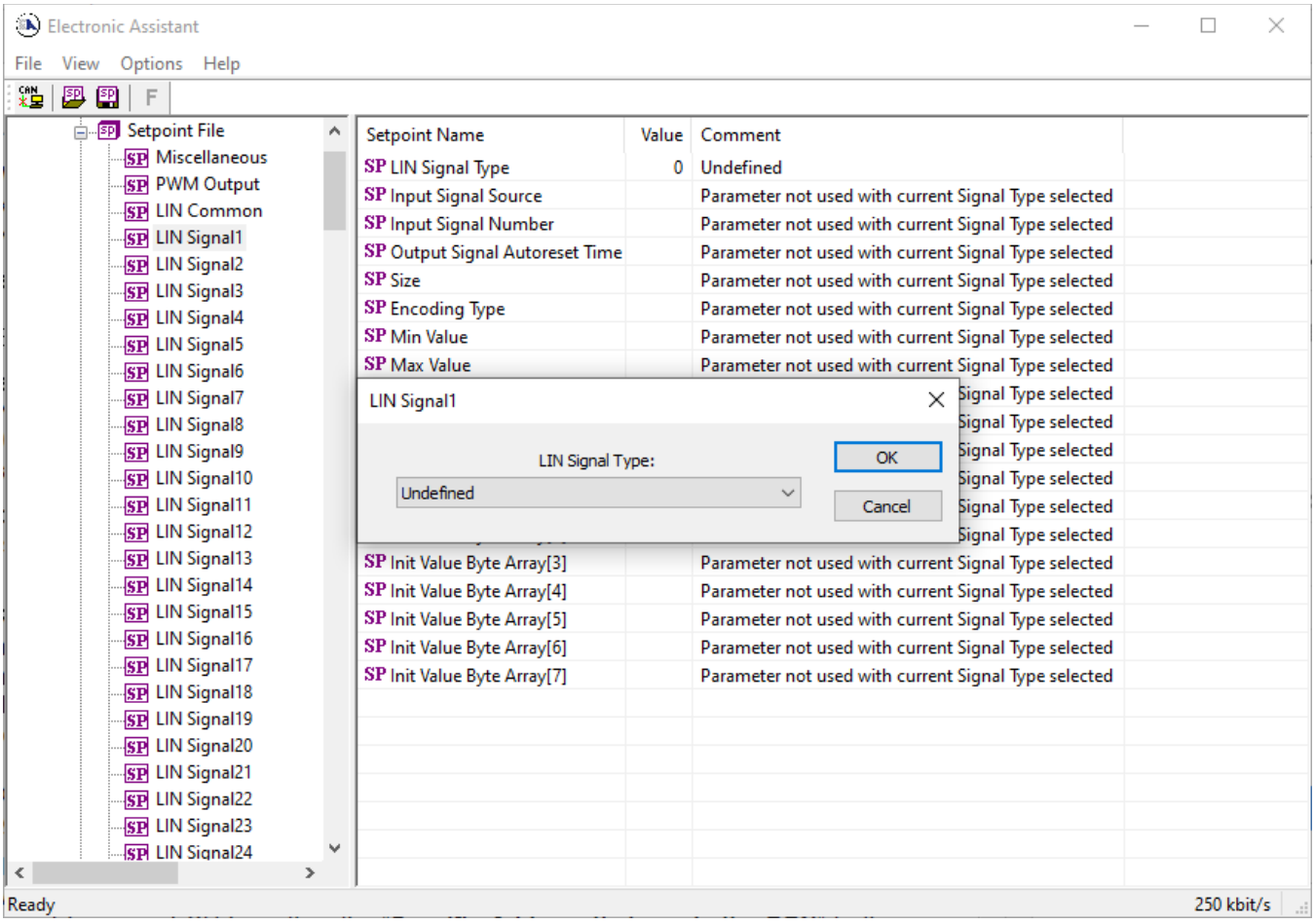


Figure 30. Changing a Configuration Parameter in the Axiomatic EA

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

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

4.3 Setpoint File

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

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

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

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

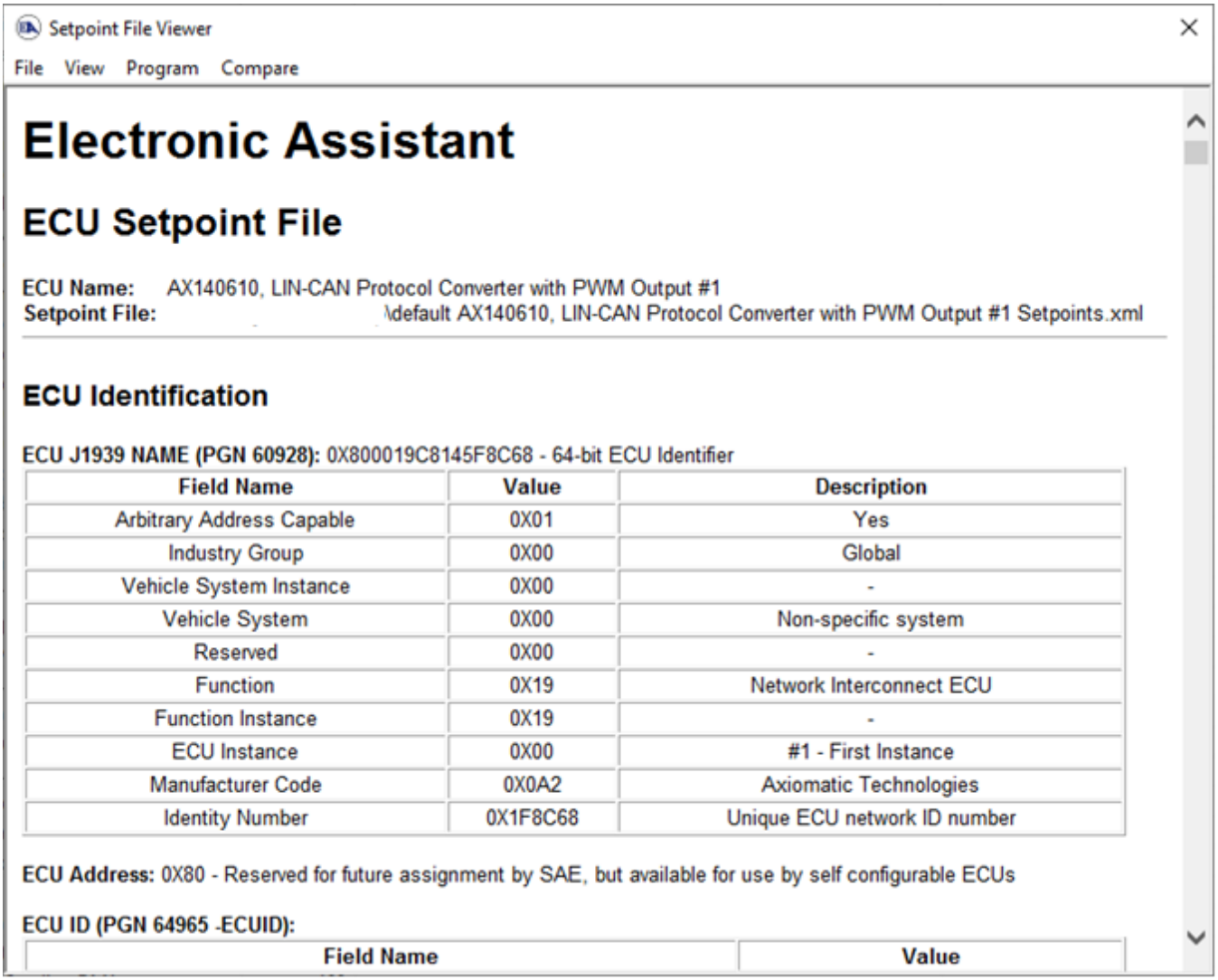


Figure 31. An Axiomatic EA Setpoint File

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

4.4 Configuration Example

The converter should be configured to perform the required system functionality before being used in the system. A detailed description of the converter configuration process for communication between CAN and LIN is presented below, as an example.

4.4.1 User Requirements

Let the converter be used to control the Microchip Technology's Interior Ambient Lighting Module with LIN interface, part number: APGRD004.

The CAN bus will carry a message controlling light intensity of the red, green and blue components of the module LED. Ramp up and dim out features will not be used. The LED control message will control all ambient lighting modules on the LIN bus in all lighting zones.

The module LED should be switched off when the LED control CAN message is not available due to loss of CAN communication, etc.

4.4.1.1 LIN Bus

The module is designed to work on a LIN bus at a standard baud rate of 10417 bit/sec, defined in SAE J2602.

The light intensity command frame has the following format:

Frame ID:	0x23
Data Length:	5 byte
Checksum:	Standard

Start Position	Length	Signal Name
0	5 bit	SelectIntensity
5	1 bit	Reserved (should be 0)
6	1 bit	RampUp
7	1 bit	DimDown
8	8 bit	RedSaturation
16	8 bit	GreenSaturation
24	8 bit	BlueSaturation
32	4 bit	ZoneSelection
36	4 bit	Reserved (should be 0)

Signal Name:	SelectIntensity
Init Value:	0x1F
Range:	0…0x1F (0 – off, 0x1F – maximum intensity)

Signal Name:	RampUp
Init Value:	0
Range:	0…1 (0 – no ramp, 1 – ramp up)

Signal Name:	DimDown
Init Value:	0
Range:	0…1 (0 – no dim, 1 – dim out)

Signal Name:	RedSaturation, GreenSaturation, BlueSaturation
Init Value:	0
Range:	0…0xFF (0 – off, 0xFF – maximum intensity)

Signal Name:	ZoneSelection
Init Value:	0x0F
Range:	0…0x0F (0 – no zones, …, 0x0F – all zones)

For more information on the LIN interface, see: “Interior Ambient Lighting Module with LIN Interface User’s Guide. Microchip Technology Inc., 2008.”

4.4.1.2 CAN bus

A dedicated J1939 proprietary message with the following parameters will be used to control the light intensity of the red, green and blue components of the module LED:

Transmission Repetition Rate:	0.5 sec
Data Length:	8
Default Priority:	6
Parameter Group Number:	65280 (Proprietary B)

Start Position	Length	Parameter Name	SPN
1	1 byte	IntensityRed	N/A
2	1 byte	IntensityGreen	N/A
3	1 byte	IntensityBlue	N/A

Parameter Name:	IntensityRed, IntensityGreen, IntensityBlue
Data Length:	1 byte
Resolution:	0.4 %/bit
Offset:	0
Type:	Measured

4.4.2 Configuration Steps

As a first step, create a block diagram of the required converter configuration using the converter function blocks, Figure 32. Then, configure each individual function block, see Figure 32.

Start from LIN signals. Configure constant LIN signal containing: SelectIntensity, RampUp, and DimDown as *LIN Signal #1*. Set *Signal Type* to *Scalar*, *Size* to *8 bit*, *Encoding Type* to *BCD Value* and *Init Value Scalar* to *0x1F*, see Figure 33.

Then configure *LIN Signal #2* as RedSaturation signal, see: Figure 34. This signal will have *CAN Receive #1* as *Input Signal Source* and *Physical Value* as *Encoding Type*. *Min Value*, *Max Value*, *Scale* and *Offset* will be set to convert 0…100% physical value to 0x00…0xFF LIN signal range:

```
MinValue   = 0;
MaxValue   = 0xFF;
Offset     = 0 [%];
Scale      = 100%/0xFF=0.3922 [%/Bit]
```

Configure *LIN Signal #3* and *LIN Signal #4* in a similar way as GreenSaturation and BlueSaturation signals, connecting them to: *CAN Receive #2* and *CAN Receive #3*, respectively, see Figure 35 and Figure 36.

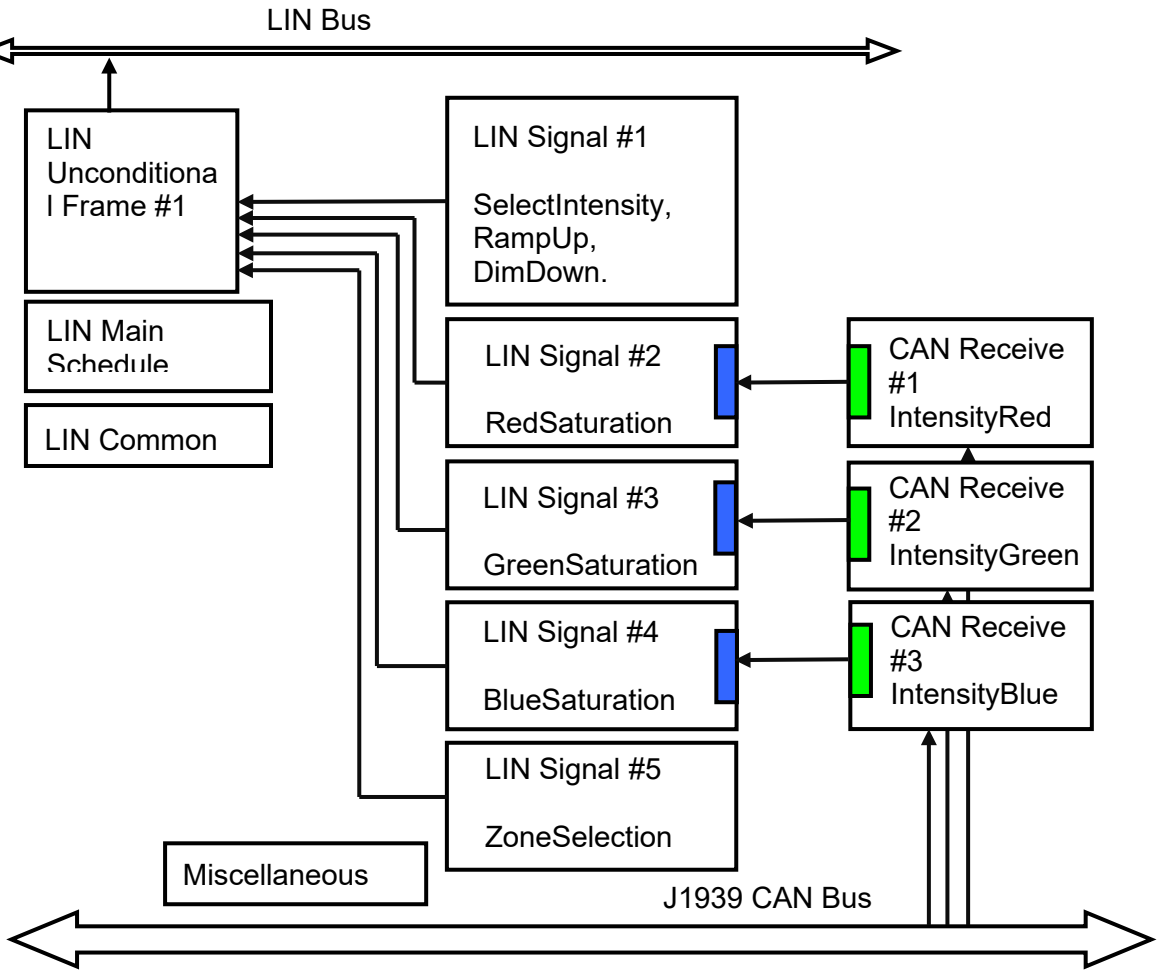


Figure 32. Block Diagram of the Example Converter Configuration

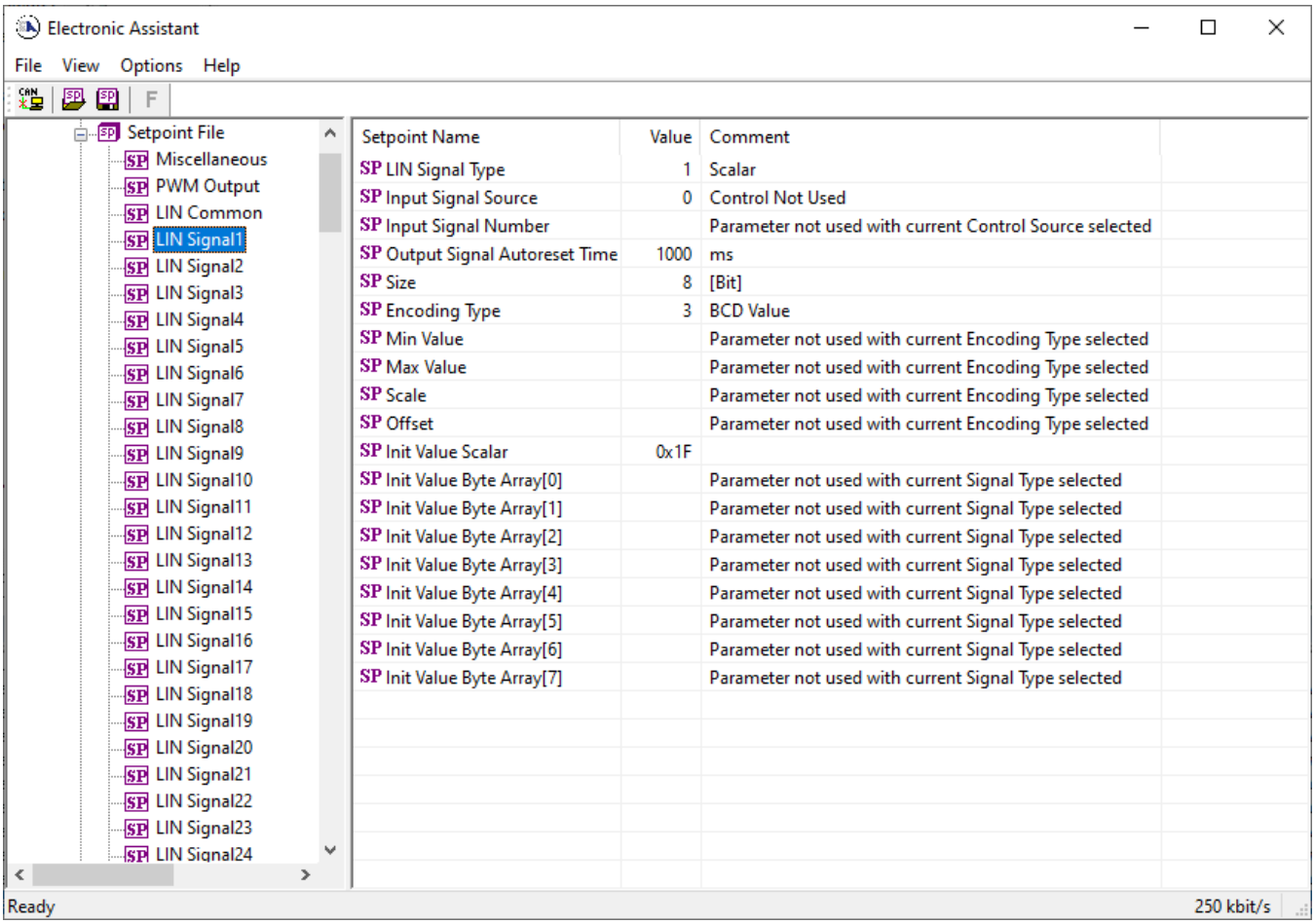


Figure 33. Example Configuration. LIN Signal #1

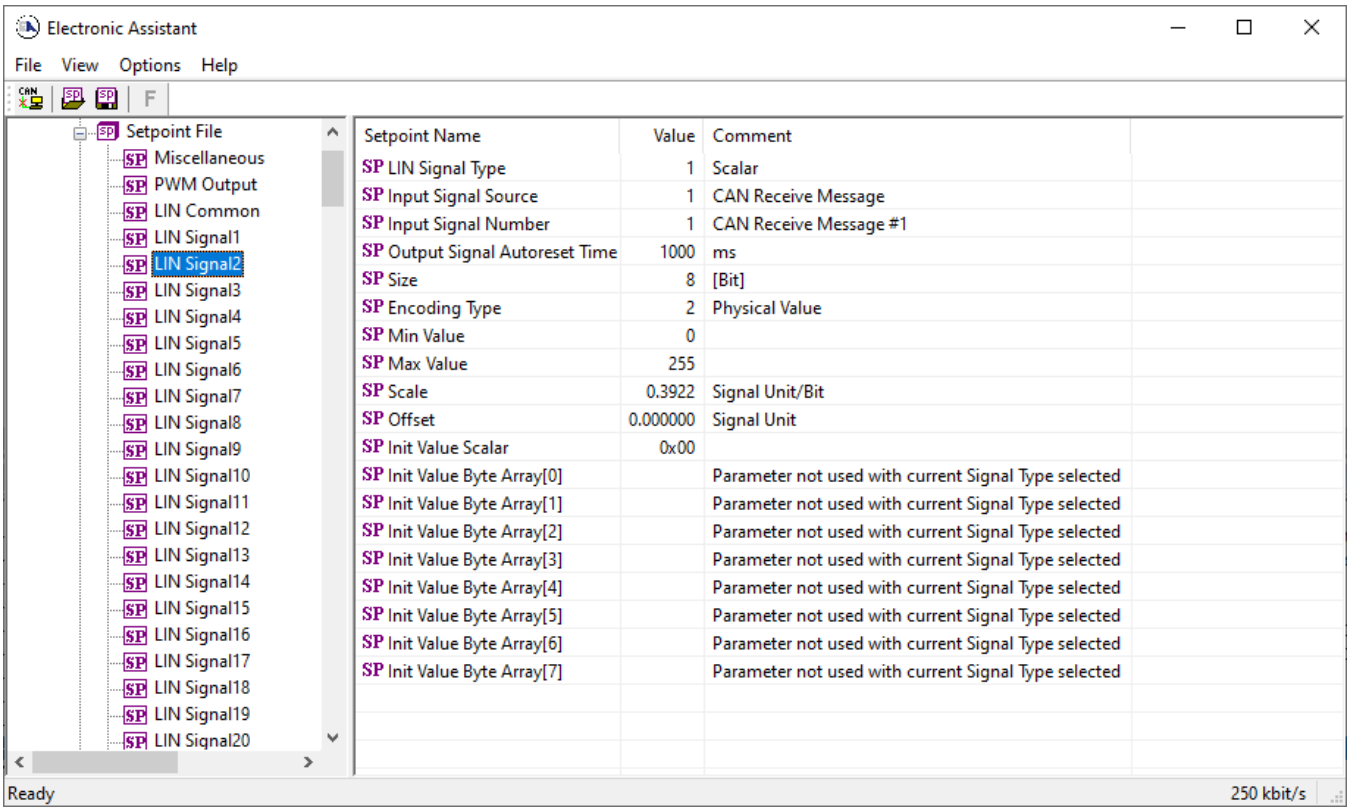


Figure 34. Example Configuration. LIN Signal #2

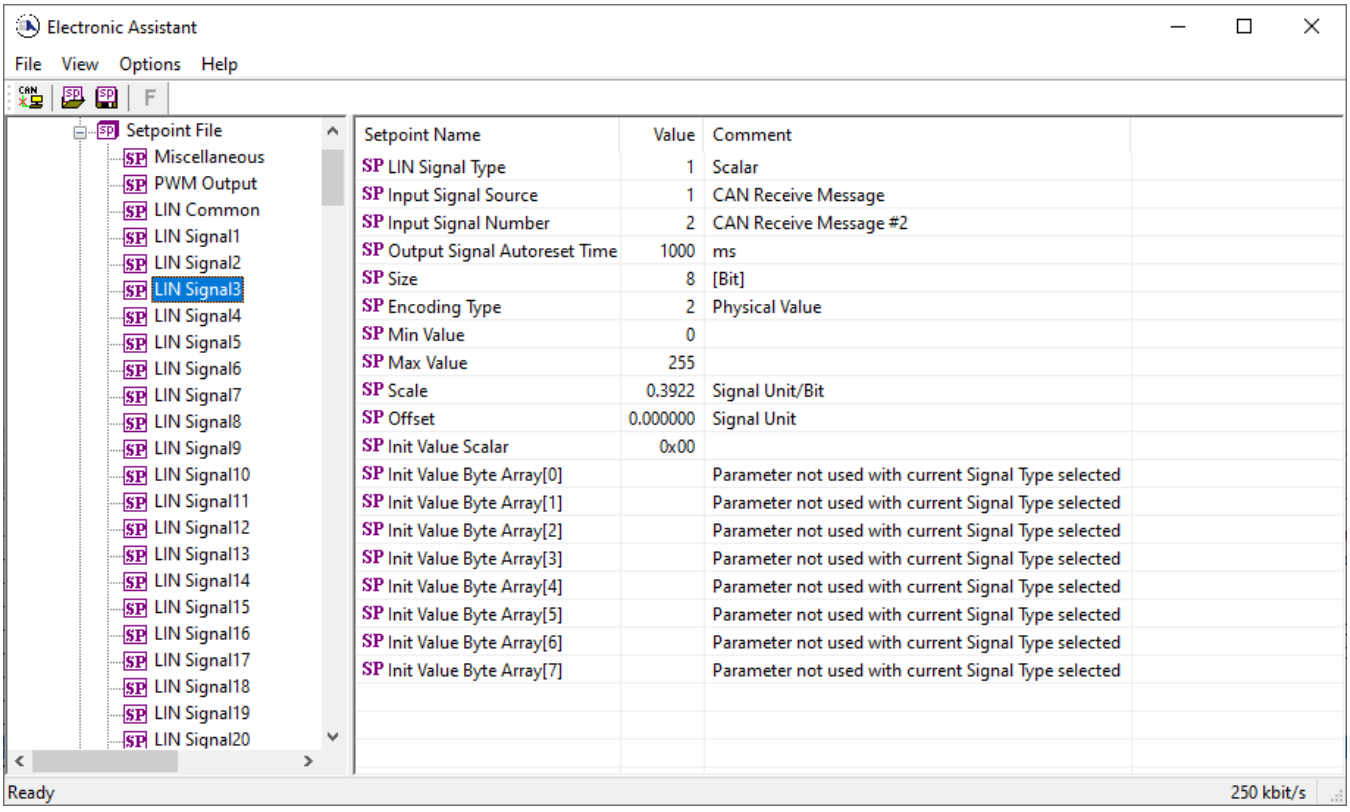


Figure 35. Example Configuration. LIN Signal #3

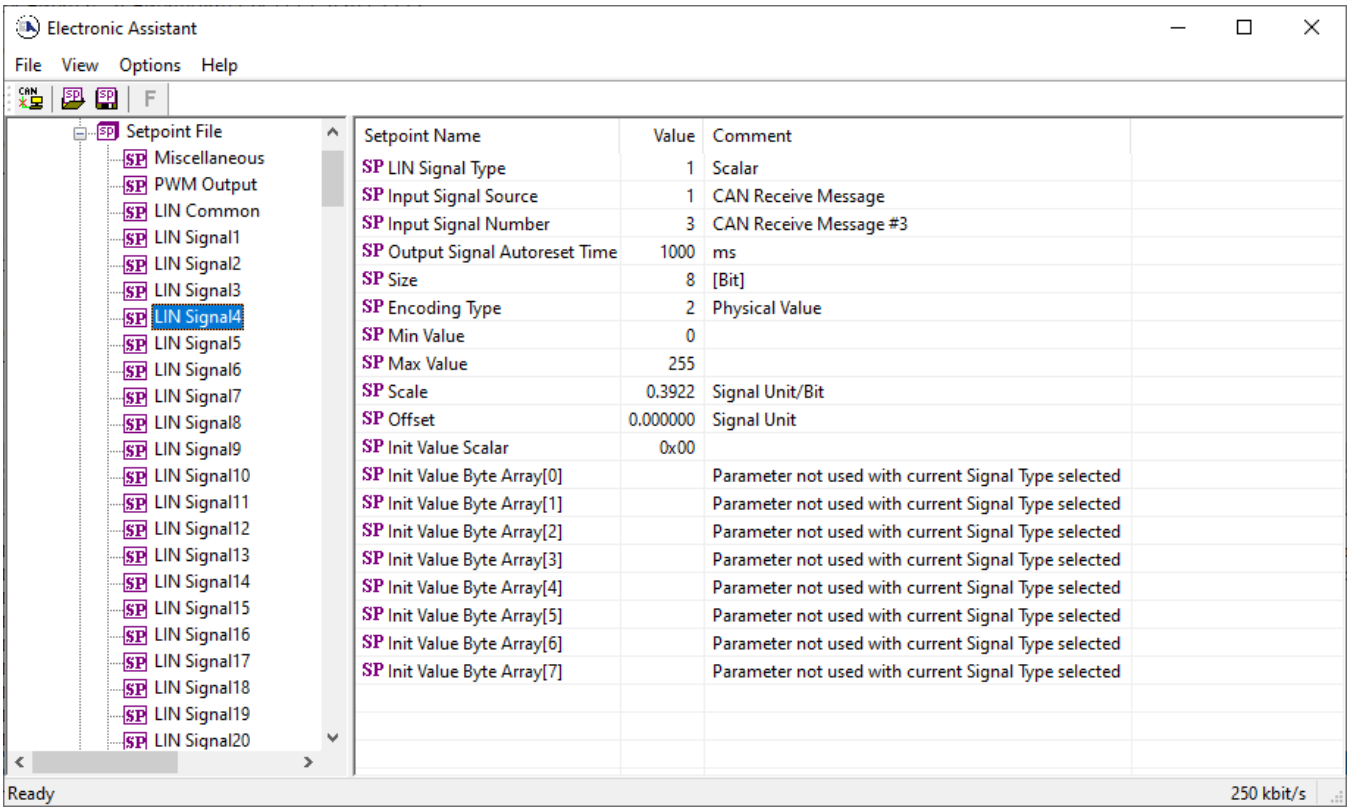


Figure 36. Example Configuration. LIN Signal #4

Configure *LIN Signal #5* as ZoneSelection constant signal. Set *Signal Type* to *Scalar*, *Size* to *8 bit*, *Encoding Type* to *BCD Value* and *Init Value Scalar* to *0x0F*.

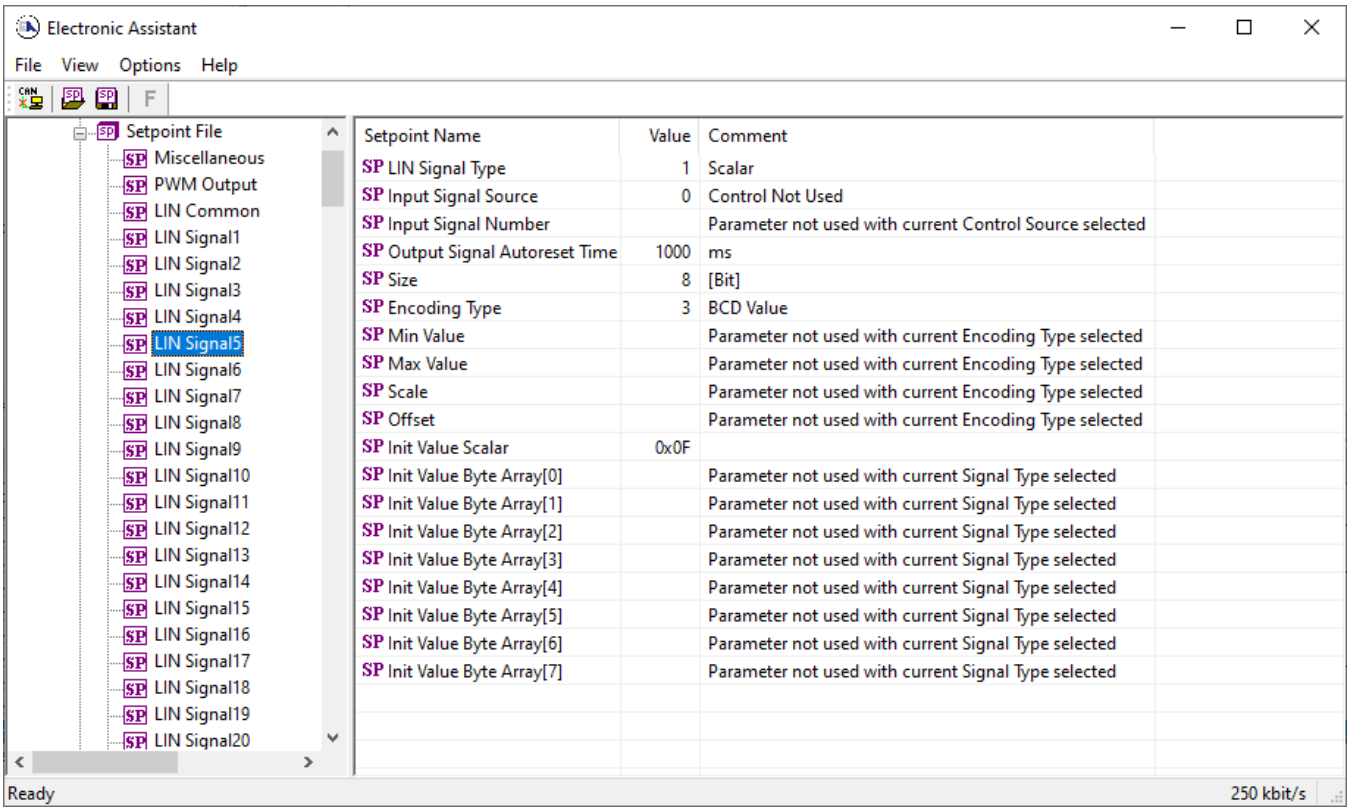


Figure 37. Example Configuration. LIN Signal #5

Now, configure *LIN Unconditional Frame #1*. Set *LIN Frame Kind* to *Publish*, *Frame ID* to *0x23*, *Size* to *5 bytes*. Add all previously configured LIN signals to the frame using *(Signal #1…10 Number, Signal #1…10 Offset)* configuration parameter pairs.

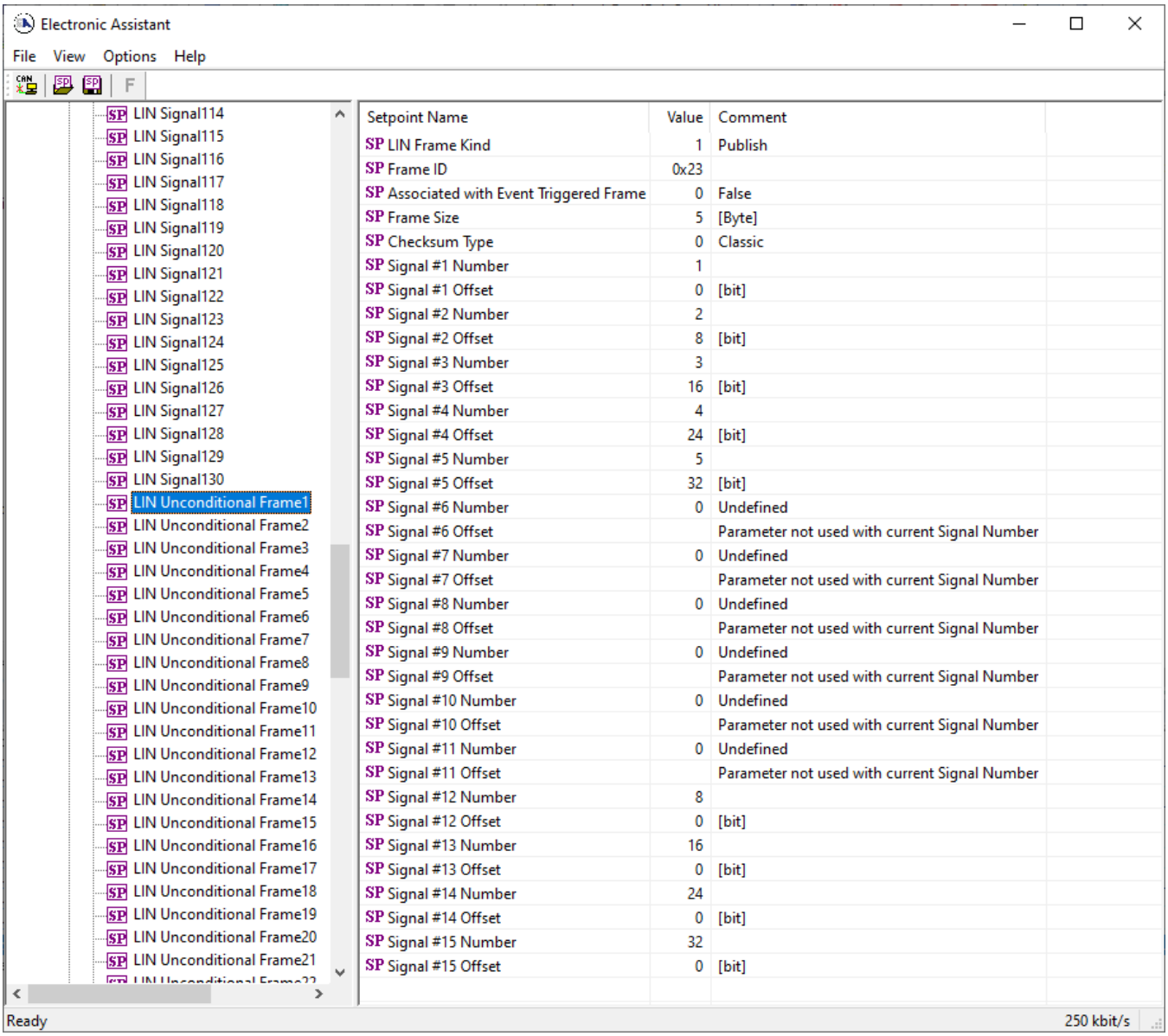


Figure 38. Example Configuration. LIN Unconditional Frame #1

Set *LIN Schedule Table #1* with only one entry: *LIN Unconditional Frame #1*. Set *Delay* to *50 ms*, see the figure below.

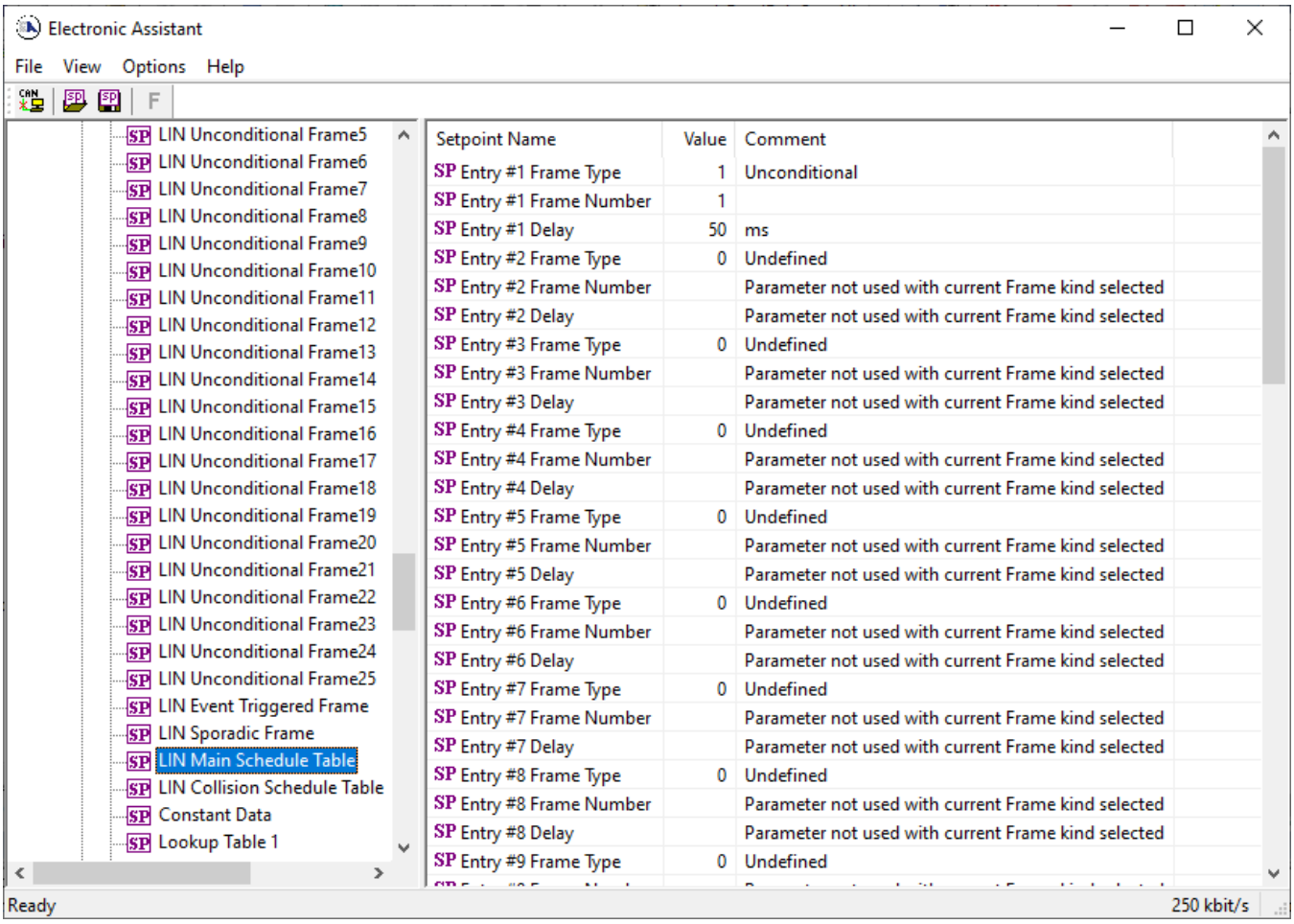


Figure 39. Example Configuration. LIN Main Schedule Table #1

Finally, start the LIN bus communication by configuring the *LIN Common* function block. Set: *Node Type* to *Master* and *Baud Rate* to *10417 bit/sec*. *Tick Time* should be left at the default value of *10 ms*.

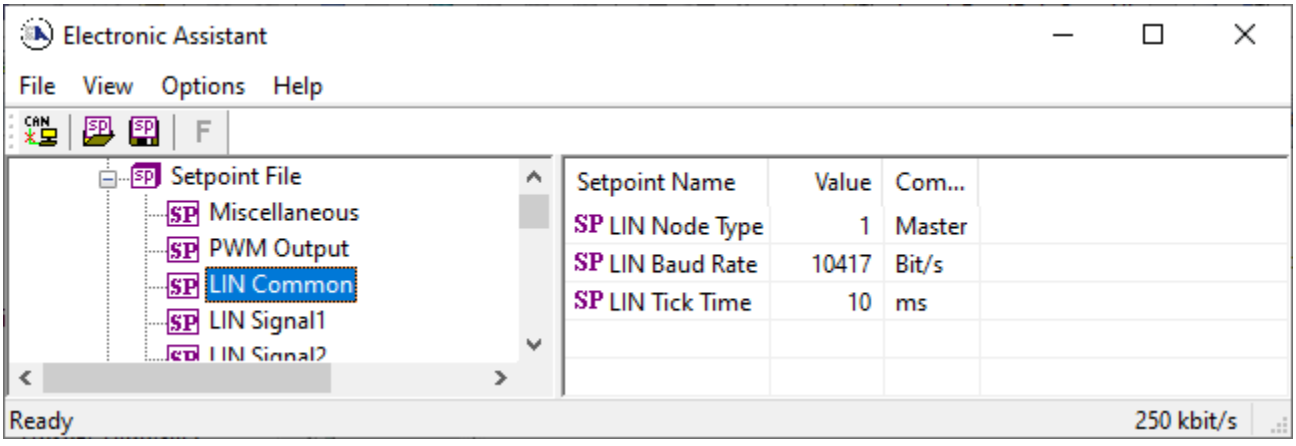


Figure 40. Example Configuration. LIN Common

The LIN bus is now running. Configure CAN bus input signals. All three input signals will parse one CAN message. Start from *CAN Receive #1* and configure it as IntensityRed signal. The

Autoreset Time set to *1500 ms* (1.5 seconds) to ensure that the LED will not switch off in case one or two CAN messages coming every 0.5 seconds are accidentally lost, see the figure below.

Configure *CAN Receive #2* and *CAN Receive #3* in a similar way as IntensityGreen and IntensityBlue signals, respectively, see Figure 42the figure below.

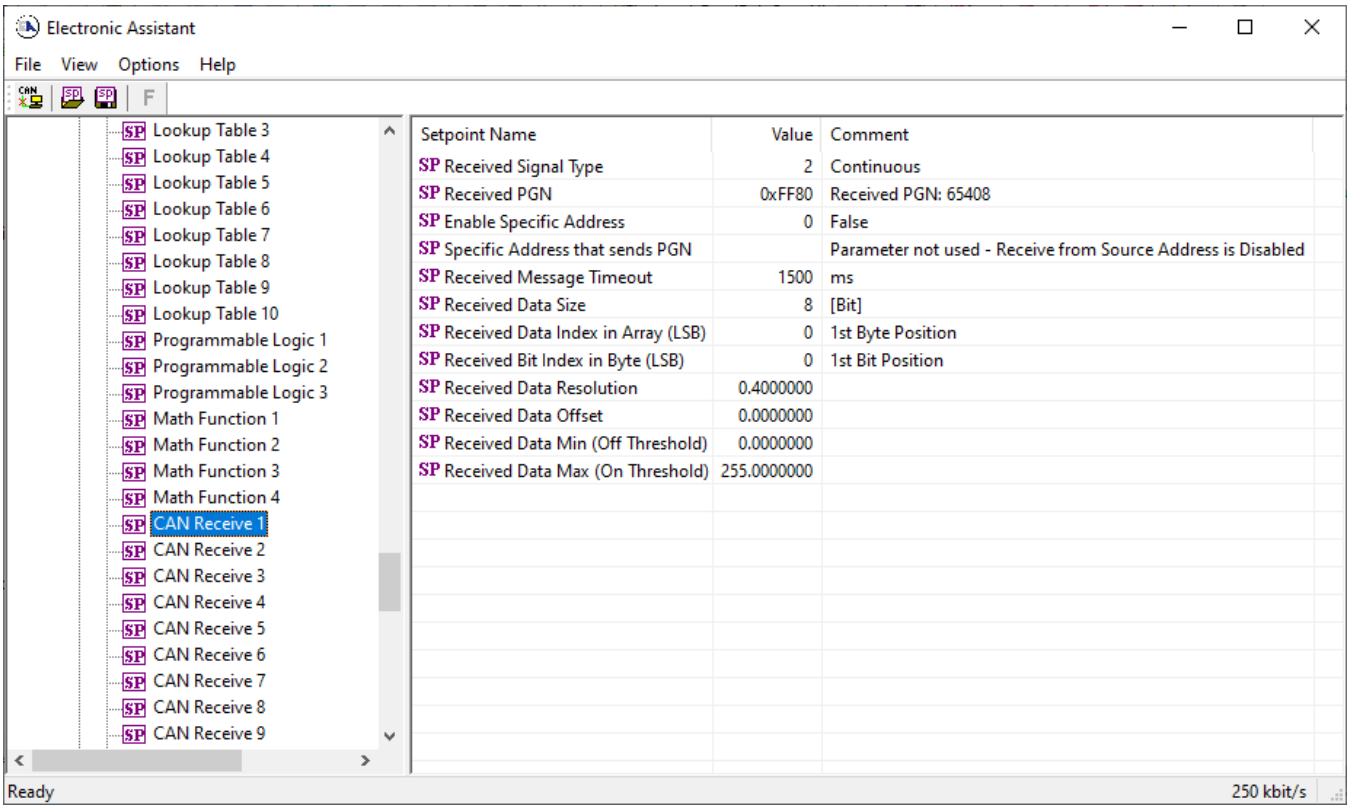


Figure 41. Example Configuration. CAN Receive #1

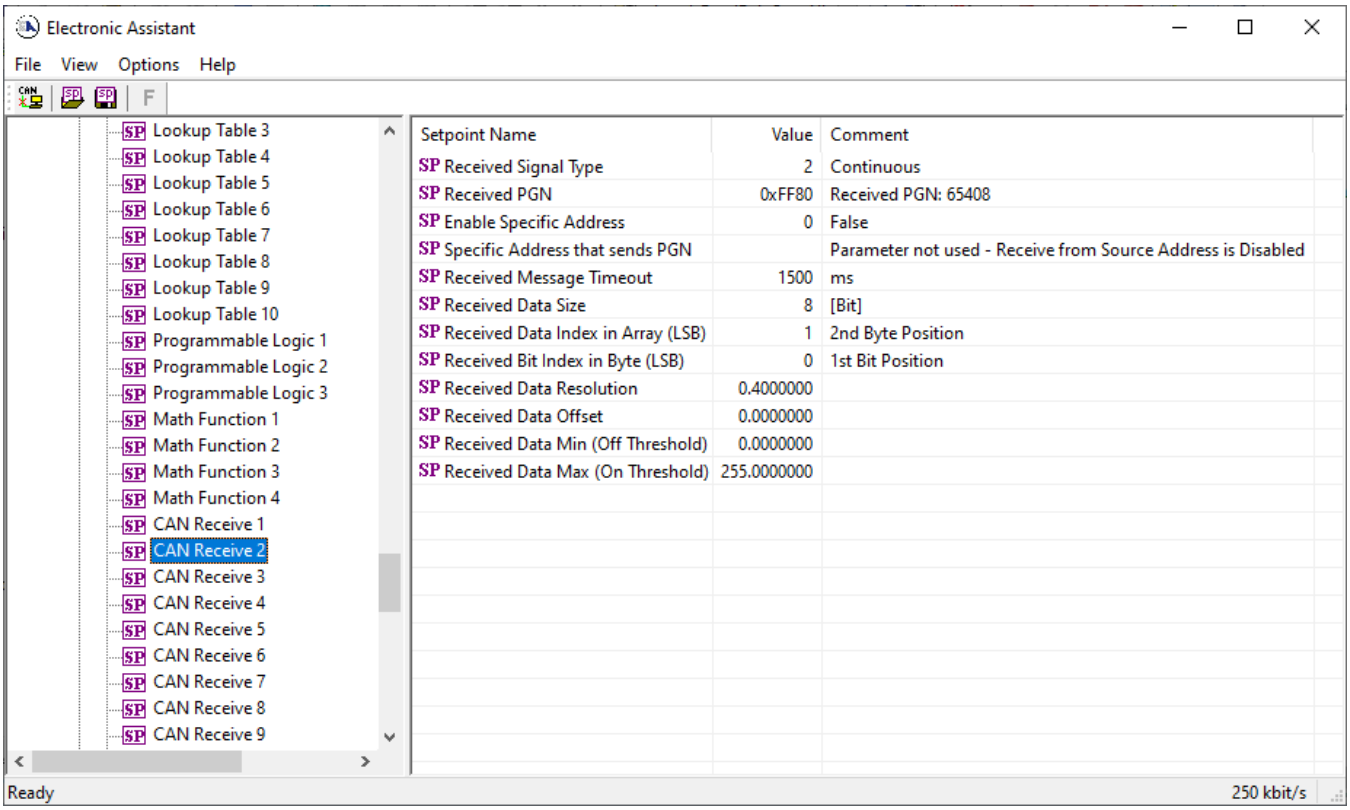


Figure 42. Example Configuration. CAN Receive #2

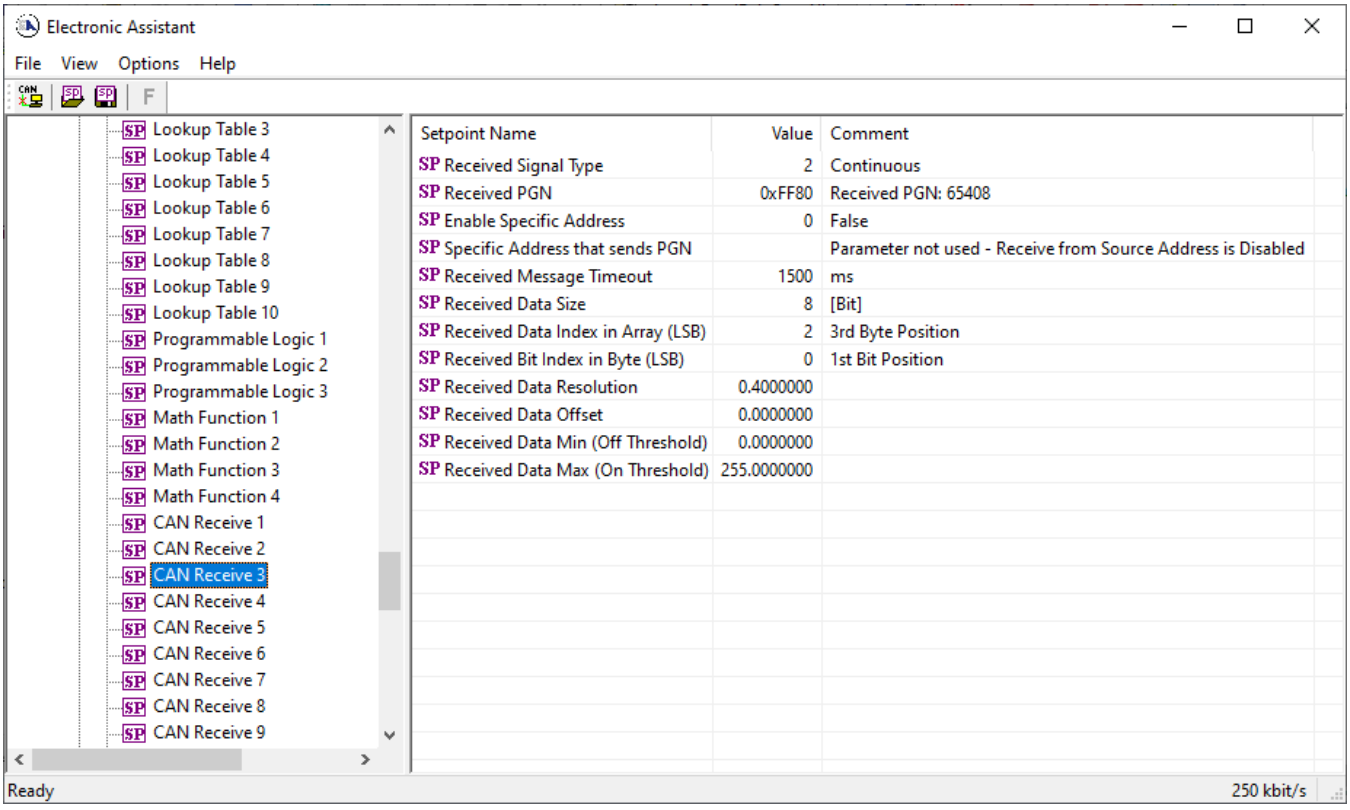


Figure 43. Example Configuration. CAN Receive #3

Now, the converter configuration is finished. All new settings are in the non-volatile memory. You can test the new converter functionality by sending a LED control message on the CAN bus. For example, a message with all intensity data fields set to 0xFA (maximum value) will turn the module LED to the intense white color.

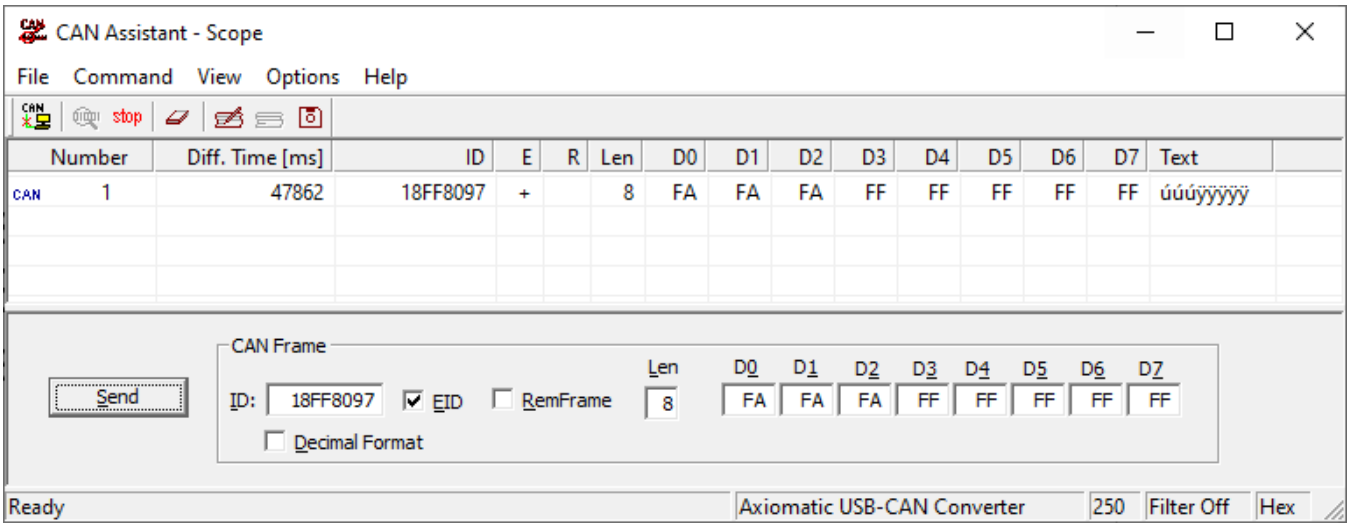


Figure 44. Example Configuration. Generating CAN Test Message

The LED will stay on for 1.5 seconds and then switch off, if the message is not being retransmitted, as per the user requirements.

A setpoint file containing configuration parameters for this example is available upon request.

5 FLASHING NEW FIRMWARE

When the new firmware becomes available, the user can replace the converter firmware in the field using the unit embedded bootloader. The firmware file can be received from Axiomatic on request.

To flash the new firmware, the user should activate the embedded bootloader. To do so, start the Axiomatic EA and in the *Bootloader Information* group screen click on the *Force Bootloader to Load on Reset* parameter. The following dialog will appear, see Figure 45.

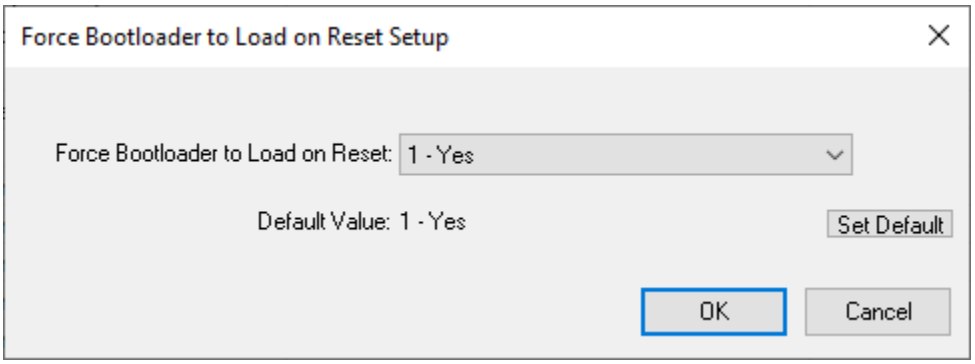


Figure 45. Bootloader Activation. First Step

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, see Figure 46. Select *Yes*.

After automatic reset, instead of *AX140610, LIN - CAN Protocol Converter with PWM Output*, 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.

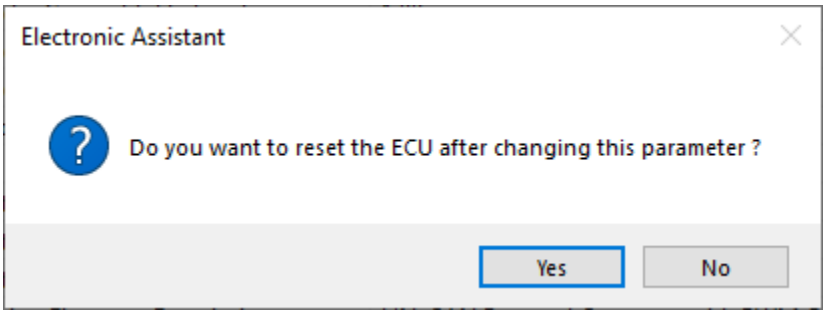


Figure 46. Bootloader Activation. Final Reset

All the bootloader specific information: converter hardware, bootloader details, and the currently installed application firmware remains the same in the bootloader mode and the user can read it in the *Bootloader Information* group screen, see Figure 47. The information can be slightly different for different versions of the bootloader.

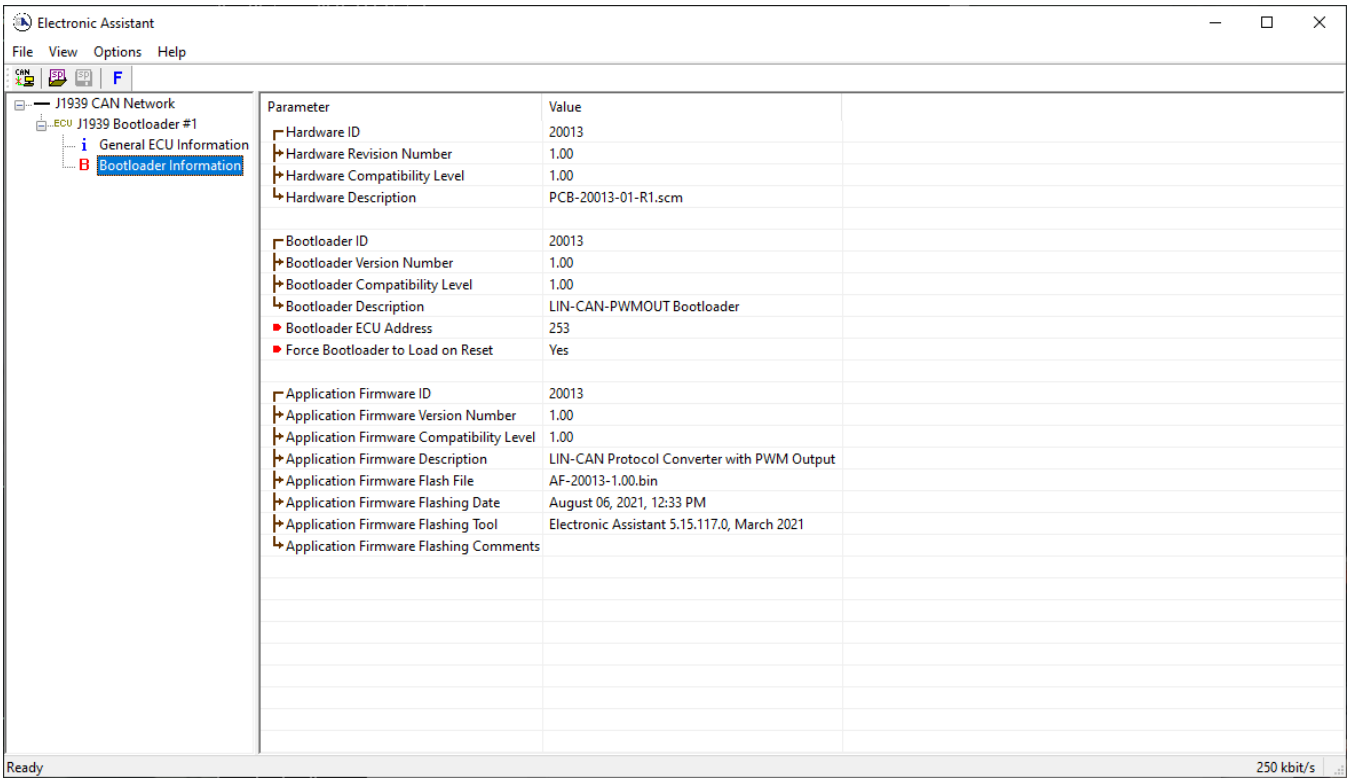


Figure 47. Bootloader Information Screen

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

To flash the new firmware, the user should click on F toolbar icon or from the *File* menu select the *Open Flash File* command. The *Open Application Firmware Flash File* dialog will appear. Pick up the flash file with the new converter firmware and confirm the selection by pressing the *Open* button. The *Flash Application Firmware* dialog window will appear[1], see Figure 48.

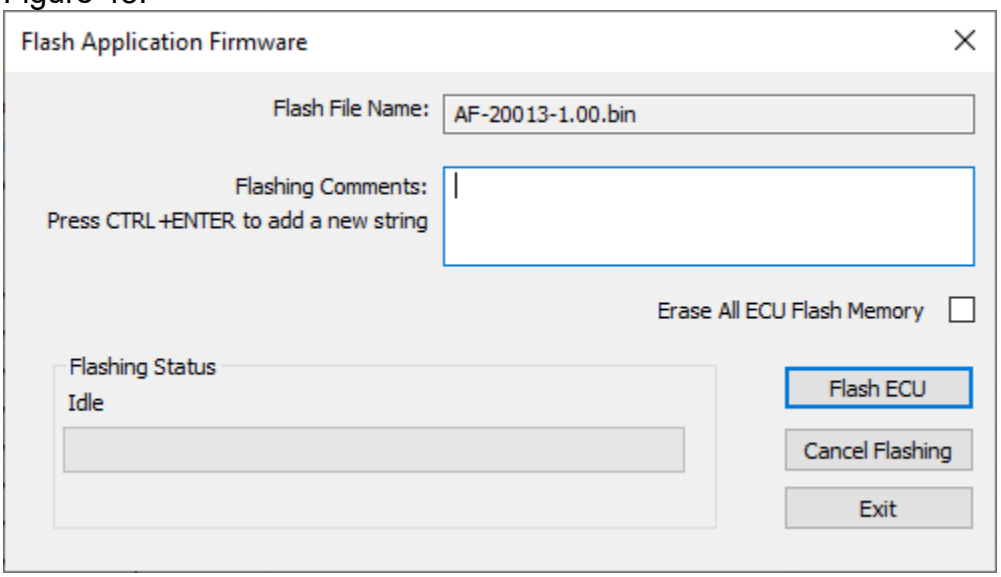


Figure 48. Flashing New Firmware. Preparation

[1] *In this example, instead of the new firmware, the old firmware V1.00 is being simply re-flashed.*

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

The user can also check the *Erase All ECU Flash Memory* flag to erase all configuration parameters related to LIN set by the old firmware and force the converter to load the LIN default values after flashing the new firmware. This operation, used in other products to reset all configuration parameters kept in the flash memory to their default values, has no effect in this product. This is because all the configuration parameters of the converter other than LIN are stored in a separate EEPROM memory.

Otherwise, the default LIN values will be set only to the new configuration parameters introduced in the new firmware. The old configuration parameters will keep their original values unless otherwise is stated in the user manual.

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, see Figure 49.

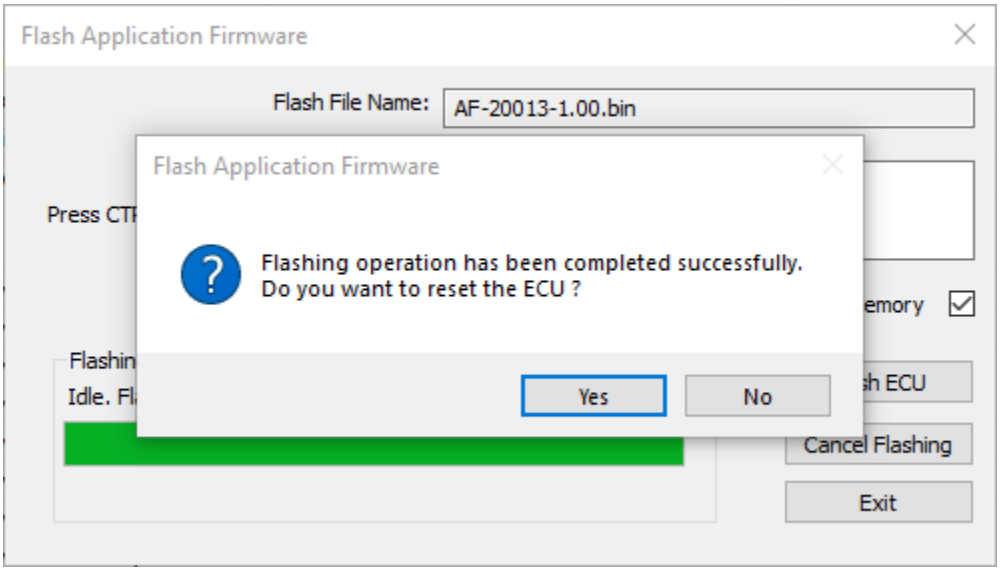


Figure 49. Flashing New Firmware. Final Reset.

Select *Yes* and see the ECU running the new firmware, see Figure 50. This will indicate that the flashing operation has been performed successfully.

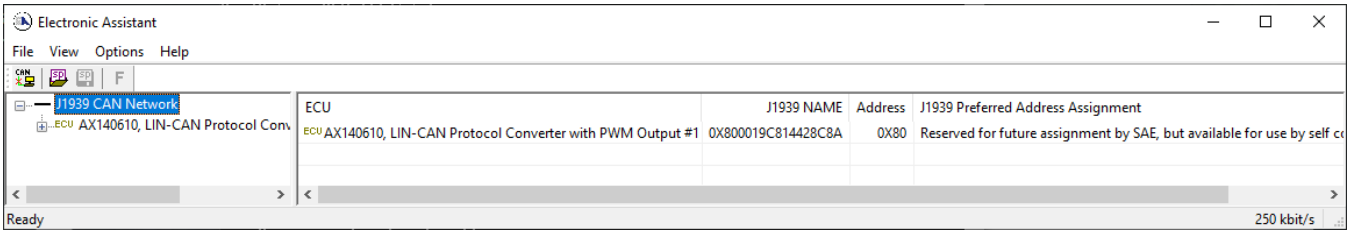


Figure 50. Firmware has been Updated. New Firmware Screen

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

6 TECHNICAL SPECIFICATIONS

Specifications are typical at nominal input voltage and 25 degrees C unless otherwise specified.

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/.

6.1 Power

Table 34. Power

Power Supply Input - Nominal	12 V or 24 Vdc nominal; 9…32 Vdc
Surge Protection	Provided up to -100V
Reverse Polarity Protection	Provided up to -80 V
Undervoltage Protection	Undervoltage hardware shutdown at 6 V
Overvoltage Protection	Overvoltage hardware shutdown at 45 V

6.2 Output

Table 35. Power

PWM Output	One Output configurable as PWM/Frequency or Digital PWM Signal, Frequency Signal, or Mixed Output: - 1 Hz to 20 kHz - 0-100% Duty Cycle (User configurable) - 6V or 13V amplitude - Push pull output - Maximum load is 50 mA (at 5V) or 30 mA (at 12V). - Over-current protection (50 mA) Digital Level: - Digital On/Off - 6V or 13V Amplitude - Maximum load is 50 mA (at 5V) or 30 mA (at 12V)
Output Accuracy	PWM Signal: +/- 0.5% Frequency Signal: +/- 0.1%
Output Feedback Accuracy	PWM Signal: +/- 0.5% Frequency Signal: +/- 0.5%

6.3 Control Software

Table 36. Control Software

Control Logic	Configurable with the Axiomatic Electronic Assistant, P/Ns: **AX070502** or **AX070506K**

6.4 General Specifications

Table 37. General Specifications

Microprocessor	STM32F413CGU6

<table>
<tr><td></td><td>32-bit, 1 Mbyte Flash Program Memory</td></tr>
<tr><td>LIN Port</td><td>1 LIN (LIN 2.2, 2.4…20 kbit/s)</td></tr>
<tr><td>CAN Port</td><td>1 CAN (SAE J1939, 250kbit/s, 500kbit/s, 667kbit/s, 1Mbit/s. Automatic Baud Rate Detection)</td></tr>
<tr><td>Power Supply Current</td><td>19mA @ 24Vdc, 35mA @ 12Vdc</td></tr>
<tr><td>Operating Conditions</td><td>-40 to 85 °C (-40 to 185 °F)</td></tr>
<tr><td>Weight</td><td>0.143 lbs. (0.065 kg)</td></tr>
<tr><td>Protection Rating</td><td>IP67</td></tr>
<tr><td>Vibration</td><td>Random Vibration: 7.68 Grms peak
Sinusoidal Component: 10 g peak
Based on MIL-STD-202G, Methods 204G and 214A</td></tr>
<tr><td>Shock</td><td>50 g half sine pulse, 9ms per axis
Based on MIL-STD-202G, Method 213B, Test Condition A</td></tr>
<tr><td>Packaging and Dimensions</td><td>Plastic Enclosure, Nylon 6-6 with 30% glass fill
Integral, TE Deutsch type connector
Refer to dimensional drawing (Below).</td></tr>
<tr><td>Electrical Connections</td><td>Integral 8-pin connector (equivalent TE Deutsch P/N: DT04-08PA)
A mating plug kit is available as Axiomatic P/N: AX070112.

<table><tr><th colspan="2">CAN and I/O Connector</th></tr><tr><th>Pin #</th><th>Description</th></tr><tr><td>1</td><td>PWM Output -</td></tr><tr><td>2</td><td>CAN_L</td></tr><tr><td>3</td><td>Not Used</td></tr><tr><td>4</td><td>BATT-</td></tr><tr><td>5</td><td>BATT+</td></tr><tr><td>6</td><td>LIN</td></tr><tr><td>7</td><td>CAN_H</td></tr><tr><td>8</td><td>PWM_Output+</td></tr></table></td></tr>
<tr><td>Software Reflashing</td><td>Electronic Assistant AX070502</td></tr>
<tr><td>User Interface</td><td>For SAE J1939 models, parameters are configurable using the Axiomatic Electronic Assistant.
Axiomatic Electronic Assistant KIT P/Ns: AX070502 or AX070506K
The 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.</td></tr>
</table>

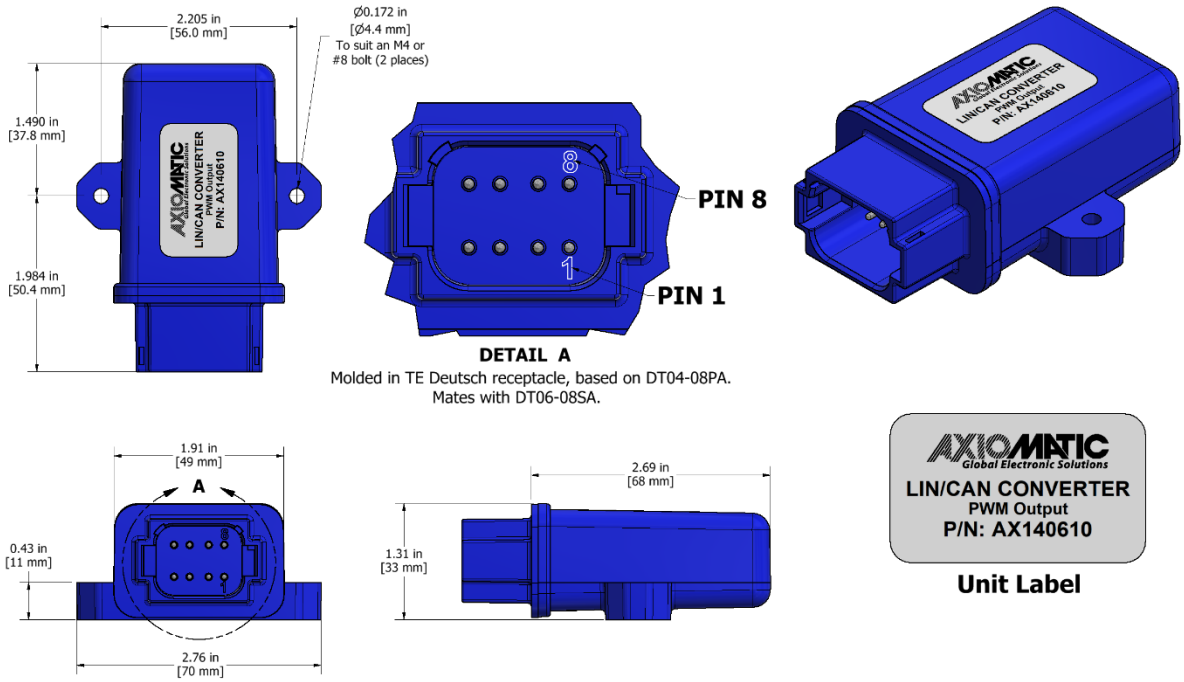


Figure 51. Unit Dimensions

7 VERSION HISTORY

User Manual Version	Date	Author	Modifications
1	August 6th, 2021	Jessica Chen	• Initial release.
-	January 24, 2022	Amanda Wilkins	• Updated dimensional drawing
1.1	September 12, 2023	Kiril Mojsov	• Performed Legacy Updates

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