

USER MANUAL

LIN – J1939 CAN Protocol Converter

P/N: AX140600 – 250kbit/s Baud Rate
P/N: AX140600-01 – 500kbit/s Baud Rate

In Europe:
Axiomatic Technologies Oy
Höytämöntie 6
33880 Lempäälä - Finland
Tel. +358 103 375 750
Fax. +358 3 3595 660
www.axiomatic.fi

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

ACRONYMS

CAN	Controller Area Network
DM	Diagnostic message. Defined in J1939/73 standard
EA	Electronic Assistant®. PC application software from Axiomatic, primarily designed to view and program Axiomatic control configuration parameters (setpoints) through CAN bus using J1939 Memory Access Protocol
ECU	Electronic control unit
EMI	Electromagnetic Interference
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
P/N	Part Number
RS-232	PC serial port interface
SAE J1939	CAN-based higher-level protocol designed and supported by Society of automobile Engineers (SAE)
USB	Universal Serial Bus
UTP	Unshielded twisted pair

TABLE OF CONTENTS

1	INTRODUCTION	5
2	CONVERTER DESCRIPTION.....	6
2.1	Hardware Block Diagram.....	6
2.2	Software Organization	6
2.3	LIN Interface	6
2.4	CAN Interface	7
2.4.1	J1939 Name and Address	8
2.4.2	Slew Rate Control	8
2.4.3	Network Bus Terminating Resistors	8
2.5	Default Settings	9
3	CONVERTER LOGICAL STRUCTURE.....	10
3.1	Function Block Signals	11
3.1.1	Undefined Signal	11
3.1.2	Discrete Signal	11
3.1.3	Continuous Signal	12
3.1.4	Signal Type Conversion	12
3.1.4.1	Discrete to Continuous Conversion	12
3.1.4.2	Continuous to Discrete Conversion	12
3.1.4.3	Undefined Signal Conversion	12
3.2	LIN Interface	12
3.2.1	LIN Common	12
3.2.2	LIN Signal.....	13
3.2.2.1	Receiving LIN Signals.....	14
3.2.2.2	Transmitting LIN Signals.....	15
3.2.3	LIN Slave Response.....	15
3.2.4	LIN Unconditional Frame.....	16
3.2.5	LIN Event Triggered Frame	17
3.2.6	LIN Sporadic Frame	18
3.2.7	Main Schedule Table.....	19
3.2.8	Collision Schedule Table	20
3.3	Binary Functions.....	21
3.4	Global Parameters.....	23
3.5	CAN Interface.....	24
3.5.1	J1939 Network	24
3.5.2	CAN Input Signal.....	25
3.5.3	CAN Output Message	27
4	CONFIGURATION PARAMETERS.....	30
4.1	Electronic Assistant Software	30
4.2	Function blocks in EA	31
4.3	Setpoint File.....	32
4.4	Configuration Example	33
4.4.1	User Requirements	33
4.4.1.1	LIN Bus	34
4.4.1.2	CAN bus	35
4.4.2	Configuration Steps.....	35
5	FLASHING NEW FIRMWARE.....	42

6 TECHNICAL SPECIFICATIONS.....45
6.1 Power45
6.2 Control Software.....45
6.3 General Specifications.....45
7 VERSION HISTORY.....47

1 INTRODUCTION

The following user manual describes: architecture, functionality, configuration parameters and flashing instructions for LIN – J1939 CAN Protocol Converter. 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 V2.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. It can run in LIN master or slave mode at different baud rate from 2.4...20 kbit/s. The J1939 CAN network operates at 250 kbit/s for P/N: AX140600 or 500 kbit/s for P/N AX140600-01.

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.

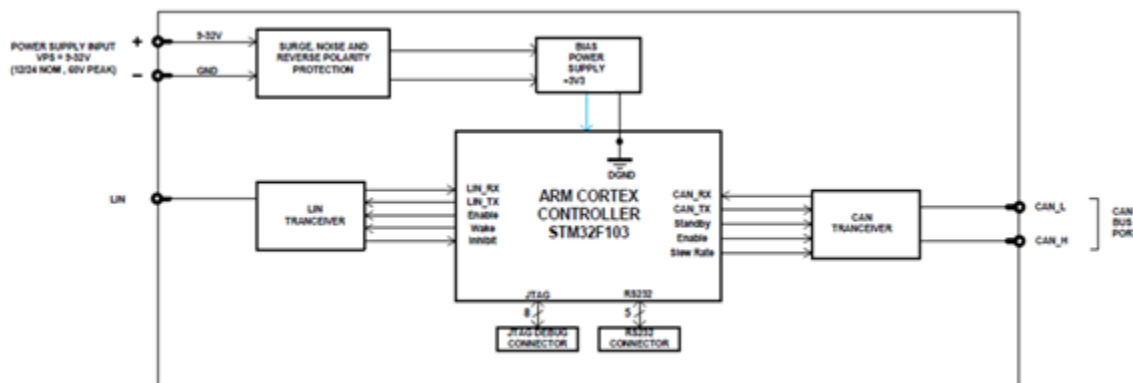


Figure 1. The Converter Hardware Block Diagram

The converter has a wide range of protection features including a transient and reverse polarity protection, see [Technical Specifications](#) section.

2.2 Software Organization

The LIN – J1939 CAN Protocol Converter 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 EA in the field, see [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 35 signals, 10 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. OCT2011.
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. 2003-05. 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).
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.

ISO/OSI Network Model Layer	J1939 Standard
	The converter can receive application specific PGNs with input signals and transmit application specific PGNs with up to five output signals. All application specific PGNs are user programmable.
	J1939/73 – Application Layer – Diagnostics. Rev. FEB 2010
	Memory access protocol (MAP) support: DM14, DM15, DM16 messages used by EA to program configuration parameters.

2.4.1 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	14 (AX140600, LIN – J1939 CAN, Axiomatic proprietary)	No
ECU Instance	3 bit	0 (First Instance)	Yes
Manufacturer Code	11 bit	162 (Axiomatic Technologies Corp.)	No
Identity Number	21 bit	Calculated on the base of the microcontroller unique ID	No

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

The controller 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 EA.

2.4.2 Slew Rate Control

The converter has an ability to adjust the CAN transceiver slew rate for better performance on the CAN physical network, see: [J1939 Network](#) function block for further details.

2.4.3 Network Bus Terminating Resistors

The controller 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.

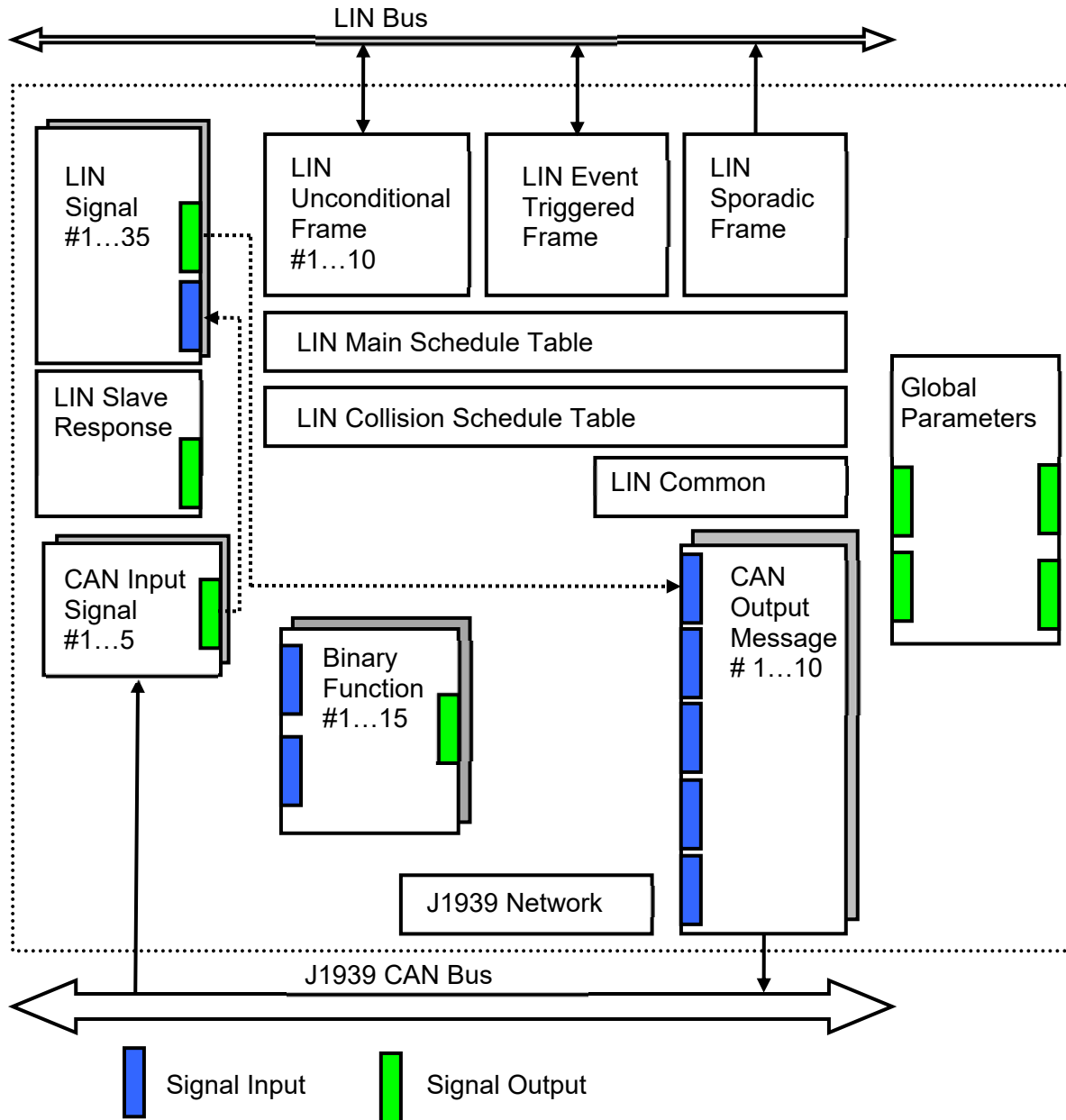
Even if the length of the CAN network is short and the signal reflection from both ends of the cable can be ignored, at least one 120 Ohm resistor is required for the majority of CAN transceivers to operate properly.

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 Input Signal*, *CAN Output Message* and *J1939 Network* function blocks. The *CAN Input Signal* 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 Output Message* 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.

For data processing, when required, the unit can use *Binary Function* blocks. They take two input signals and combine them together in one signal output using different functions.

The converter also has a *Global Parameters* function block containing two constant output signals and other auxiliary 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, see [Binary Function](#) blocks.

3.2 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 EA.

3.2.1 LIN Common

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

LIN Common

Figure 3. LIN Common Function Block

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

Table 4. 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”.

3.2.2 LIN Signal

LIN Signal function blocks are used to specify input and output signals on the LIN bus. There are 35 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.

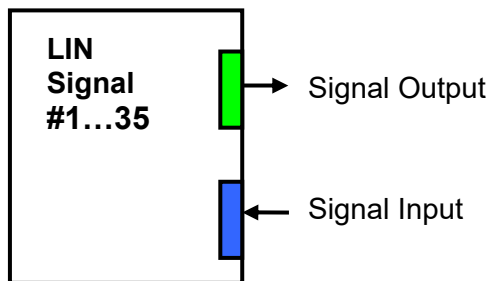


Figure 4. LIN Signal Function Block

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

Table 5. 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)

Name	Default Value	Range	Units	Description
				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.
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 6. LIN Signal Encoding Type

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

3.2.2.1 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:

$$LINOutputSignal = LINSignalCode * Scale + Offset, \quad \text{if} \quad (1)$$

$LINSignalCode \in [MinValue; MaxValue]$.

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.

3.2.2 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.

3.2.3 LIN Slave Response

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

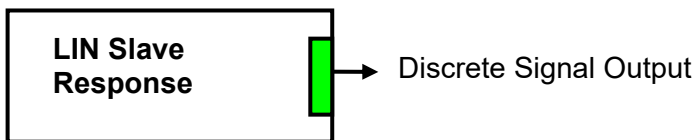


Figure 5. *LIN Slave Response* Function Block

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.

3.2.4 LIN Unconditional Frame

There are 10 *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.

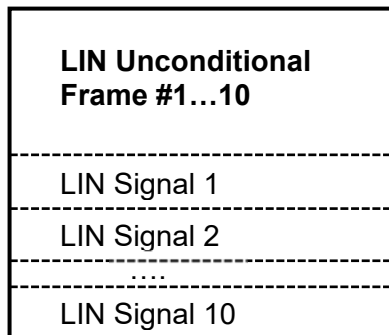


Figure 6. *LIN Unconditional Frame* Function Block

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 10 LIN signals associated with each LIN frame.

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

Table 7. *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.
Associated with Event Triggered Frame	No	{No, Yes}	–	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...35]	–	Number of the 1-st <i>LIN Signal</i> function block.
Signal #1 Offset	0	[0...63]	Bit	Offset of the LIN signal defined by the 1-st <i>LIN Signal</i> function block.
Signal #2 Number	0	[0...35]	–	Number of the 2-nd <i>LIN Signal</i> function block.
Signal #2 Offset	0	[0...63]	Bit	Offset of the LIN signal defined by the 2-nd <i>LIN Signal</i> function block.
...
Signal #10 Number	0	[0...35]	–	Number of the 10-th <i>LIN Signal</i> function block.

Name	Default Value	Range	Units	Description
Signal #10 Offset	0	[0...63]	Bit	Offset of the LIN signal defined by the 10-th <i>LIN Signal</i> function block.

The *LIN Signal* function blocks are numbered from 1 to 10. When the *Signal #1...10 Number* is equal to 0, the *LIN Signal* function block is not defined and the associated *Signal #1...10 Offset* configuration parameter is not used.

3.2.5 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.

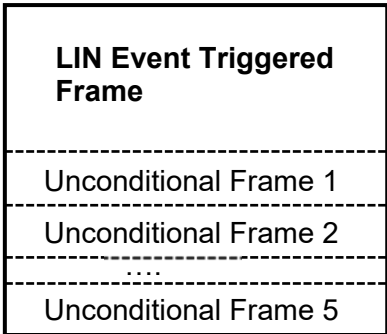


Figure 7. LIN Event Triggered Frame Function Block

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

Table 8. 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 <i>Collision Resolving Schedule Table</i> function block used in case of a collision.
Unconditional Frame #1 Number	0	[0...10]	–	Number of the 1-st <i>LIN Unconditional Frame</i> function block.
Unconditional Frame #2 Number	0	[0...10]	–	Number of the 2-nd <i>LIN Unconditional Frame</i> function block.
...

Name	Default Value	Range	Units	Description
Unconditional Frame #5 Number	0	[0...10]	–	Number of the 5-th <i>LIN Unconditional Frame</i> 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.

3.2.6 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.

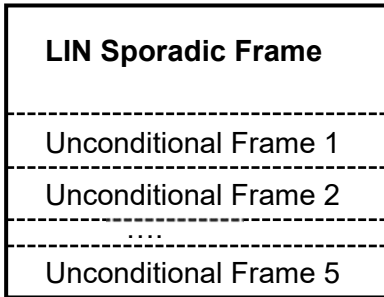


Figure 8. *LIN Sporadic Frame Function Block*

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 9. *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...10]	–	Number of the 1-st <i>LIN Unconditional Frame</i> function block.
Unconditional Frame #2 Number	0	[0...10]	–	Number of the 2-nd <i>LIN Unconditional Frame</i> function block.
...
Unconditional Frame #5 Number	0	[0...10]	–	Number of the 5-th <i>LIN Unconditional Frame</i> 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.

3.2.7 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 10 schedule entries. There are no signal inputs and outputs in this function block.

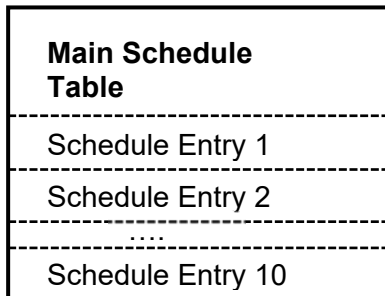


Figure 9. Main Schedule Table Function Block

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

Table 10. 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.
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 #10 Frame Type	Undefined	{Undefined, Unconditional, Event Triggered, Sporadic}	–	10-th schedule entry frame type.
Entry #10 Frame Number	0	Depends on the frame type	–	10-th schedule entry frame number.
Entry #10 Delay	0	[0...10000]	ms	10-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.

3.2.8 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.

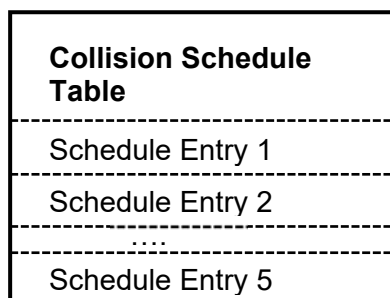


Figure 10. Collision Schedule Table Function Block

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 11. 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.
Entry #1 Frame Number	0	[0...10]	–	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...10]	–	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...10]	–	5-th schedule entry frame number.

Name	Default Value	Range	Units	Description
Entry #5 Delay	0	[0...10000]	ms	5-th schedule entry delay.

3.3 Binary Functions

There are fifteen *Binary Function* blocks available to the user for performing simple data conversions. Each *Binary Function* block has two continuous signal inputs and one continuous signal output.

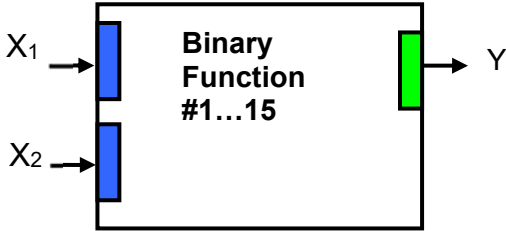


Figure 11. Binary Function Block

The *Binary Function* block performs the following data conversion:

$$Y = A \cdot F[a_1 \cdot f_1(X_1) + b_1; a_2 \cdot f_2(X_2) + b_2] + B, \quad n = 1,2; \quad (2)$$

- where:
- X_n – Input signal;
 - $f_n(X_n)$ – Unary function;
 - a_n – Scale;
 - b_n – Offset;
 - $F[x; y]$ – Binary Function;
 - A – Output Scale;
 - B – Output Offset.

The function block input signals can be undefined. The user can specify a default signal value that will be used when the signal is not defined. If the default signal value is not specified, the output signal of the function block will become undefined too.

The following unary functions can be used to process the input signals.

Table 12. Unary Functions

Function Name	Description	Comment
Undefined	$f(x) = x$	Signal is not processed
! Logical Not	$f(x) = !x$	x is converted into 4-byte unsigned integer before function is applied
~ Bitwise Not	$f(x) = \sim x$	x is converted into 4-byte unsigned integer before function is applied
abs(x) Absolute	$f(x) = x, \text{ if } x \geq 0$ $f(x) = -x, \text{ if } x < 0$	

The following binary functions are defined in the function block:

Table 13. Binary Functions

Function Name	Description	Comment
Undefined	$F[x;y] = \text{Undefined}$	Output signal is undefined
+ Addition	$F[x;y] = x + y$	
- Subtraction	$F[x;y] = x - y$	
* Multiplication	$F[x;y] = x * y$	
/ Division	$F[x;y] = x / y$	Division by 0 gives 0
% Modulus	$F[x;y] = x \% y$	x and y are converted into 4-byte unsigned integers before function is applied
max(x,y) Maximum	$F[x;y] = x, \text{ if } x \geq y$ $F[x;y] = y, \text{ if } x < y$	
min(x,y) Minimum	$F[x;y] = x, \text{ if } x \leq y$ $F[x;y] = y, \text{ if } x > y$	
== Equal	$F[x;y] = 1, \text{ if } x = y$ $F[x;y] = 0, \text{ if } x \neq y$	
!= Not Equal	$F[x;y] = 1, \text{ if } x \neq y$ $F[x;y] = 0, \text{ if } x = y$	
> Great	$F[x;y] = 1, \text{ if } x > y$ $F[x;y] = 0, \text{ if } x \leq y$	
>= Great Equal	$F[x;y] = 1, \text{ if } x \geq y$ $F[x;y] = 0, \text{ if } x < y$	
< Less	$F[x;y] = 1, \text{ if } x < y$ $F[x;y] = 0, \text{ if } x \geq y$	
<= Less Equal	$F[x;y] = 1, \text{ if } x \leq y$ $F[x;y] = 0, \text{ if } x > y$	
Logical OR	$F[x;y] = x \vee y$	x and y are converted into 4-byte unsigned integers before function is applied
&& Logical AND	$F[x;y] = x \wedge y$	x and y are converted into 4-byte unsigned integers before function is applied
Bitwise OR	$F[x;y] = x y$	x and y are converted into 4-byte unsigned integers before function is applied
& Bitwise AND	$F[x;y] = x \& y$	x and y are converted into 4-byte unsigned integers before function is applied
^ Bitwise XOR	$F[x;y] = x \wedge y$	x and y are converted into 4-byte unsigned integers before function is applied
<< Left Shift	$F[x;y] = x \ll y$	x and y are converted into 4-byte unsigned integers before function is applied
>> Right Shift	$F[x;y] = x \gg y$	x and y are converted into 4-byte unsigned integers before function is applied

The *Binary Function* has the following set of configuration parameters:

Table 14. Binary Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Binary Function	Undefined	See Binary Function table	–	$F[x;y]$ – Binary function
Output Scale	1	Any value	–	A – Output Scale
Output Offset	0	Any value	–	B – Output Offset

Name	Default Value	Range	Units	Description
Input #1 Signal Source	Not Connected	Any signal output of any function block or "Not Connected"	–	X_1 – Input Signal #1
Input #1 Signal Default	No	{No, Yes}	–	Defines whether the default signal value for X_1 is defined.
Input #1 Signal Default Value	0	Any value	–	X_1 default value, if <i>Input #1 Signal Default</i> is <i>Yes</i> .
Unary Function #1	Undefined	See Unary Function table	–	$f_1(x)$ – Unary function #1
Scale #1	1	Any value	–	a_1 – Scale #1
Offset #1	0	Any value	–	b_1 – Offset #1
Input #2 Signal Source	Not Connected	Any signal output of any function block or "Not Connected"	–	X_2 – Input Signal #2
Input #2 Signal Default	No	{No, Yes}	–	Defines whether the default signal value for X_2 is defined.
Input #2 Signal Default Value	0	Any value	–	X_2 default value, if <i>Input #2 Signal Default</i> is <i>Yes</i> .
Unary Function #2	Undefined	See Unary Function table	–	$f_2(x)$ – Unary function #2
Scale #2	1	Any value	–	a_2 – Scale #2
Offset #2	0	Any value	–	b_2 – Offset #2

3.4 Global Parameters

The *Global Parameters* function block gives the user access to the converter supply voltage and the microcontroller internal temperature as well as to a set of two constant output signals. The constant output signals – one discrete and one continuous – can be used by other function blocks for debugging and any other purposes.

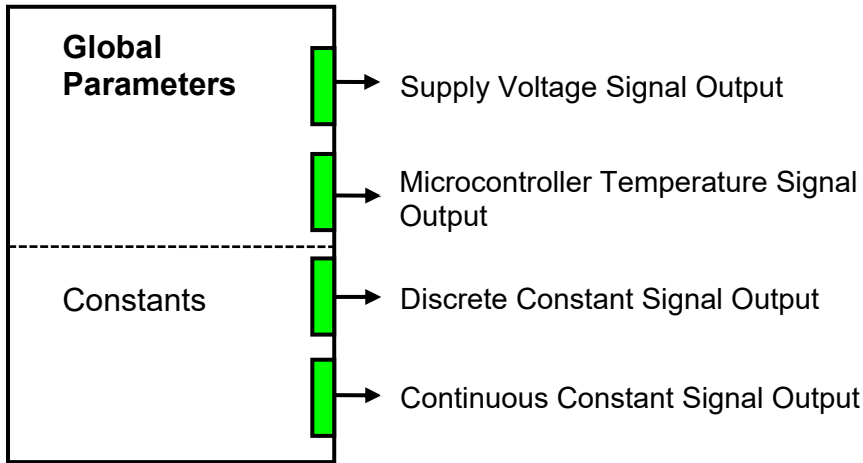


Figure 12. Global Parameters Function Block

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

Table 15. Global Parameters Function Block Configuration Parameters

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

3.5 CAN Interface

The converter CAN interface functionality is defined by *J1939 Network*, *CAN Input Signal*, *CAN Output Message* function blocks.

CAN signals are received from the CAN bus by the *CAN Input Signal* function blocks. CAN signals are transmitted on the CAN bus in the CAN single frame messages defined by the *CAN Output Message* function blocks. The *J1939 Network* function block contains the global CAN bus configuration settings.

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

3.5.1 J1939 Network

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



Figure 13. J1939 Network Function Block

Configuration parameters of the *J1939 Network* function block are presented below:

Table 16. 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.
ECU Address	128	[0...253]	–	J1939 ECU address.
Slew Rate	Low	{Low, High}	–	Slew rate control of the CAN transceiver.

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

The *ECU Address* can be changed by the user or as a result of an arbitration process on the J1939 CAN network. It can be also changed by a commanded address message.

The *Slew Rate* configuration parameter defines the slew rate of the CAN transceiver the following way:

Table 17. CAN Transceiver Slew Rate

Slew Rate Value	Transceiver Slew Rate	Note
Fast	19 V/μs	Available for both: 250kbit/s and 500kbit/s converters
Slow	4 V/ μs	Only available for 250kbit/s converts

The slow slew rate is preferable for the 250kbit/s converters due to the reduced EMI of the transceiver. The fast slew rate is necessary when the distance between nodes substantially exceeds the 40 m standard maximum value defined by the J1939/11(15) standard.

The 500kbit/s protocol converters always use the fast slew rate.

3.5.2 CAN Input Signal

There are 5 *CAN Input Signal* 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.

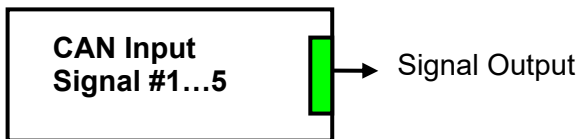


Figure 14. CAN Input Signal Function Block

The *CAN Input Signal* function block reads single-frame application specific CAN messages and extracts CAN signal data presented in user-defined data format. Different *CAN Input Signal* 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 Input Signal* 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 Input Signal* function block are presented below:

Table 18. CAN Input Signal Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
Signal Type	Undefined	{Undefined, Discrete, Continuous}	–	J1939 Signal type
PGN	0xFF00	Any J1939 PGN value ¹	–	Signal message PGN value.
PGN From Selected Address	No	{No, Yes}	–	Only CAN messages from the selected address will be accepted, if “Yes”.
Selected Address	0	[0; 253]	–	Address of the ECU transmitting CAN messages if PGN From Selected Address is set to “Yes”.
Data Position Byte	1	[1; 8]	–	Start byte of the CAN input signal in the CAN message data frame.
Data Position Bit	1	[1; 8]	–	Start bit of the CAN input signal in the Data Position Byte.
Size	1	[1...32]	bit	CAN input signal size.
Resolution	1	Any value	Signal Units / bit	CAN input signal resolution for continuous input signals.
Offset	0	Any value	Signal Units	CAN input signal offset for continuous input signals.
Autoreset Time	500	[0; 10000]	ms	Function block signal output auto-reset time. If Autoreset Time is 0, auto-reset is disabled.

¹Proprietary 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 *Data Position Byte* and *Data Position Bit* 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:

$$\text{StartBit} = (\text{DataPositionByte} - 1) \cdot 8 + (\text{DataPositionBit} - 1), \quad (3)$$

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

Resolution and *Offset* configuration parameters are set for continuous CAN input signals. They are not used with discrete CAN signals.

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 *Autoreset Time* is not equal to 0, the function block will auto-reset the output signal to the undefined state if the output signal is not being updated within the auto-reset time frame by the new CAN message data.

3.5.3 CAN Output Message

There are 10 *CAN Output Message* 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 can contain up to 5 CAN output signals. Each CAN output signal is presented by its function block signal input.

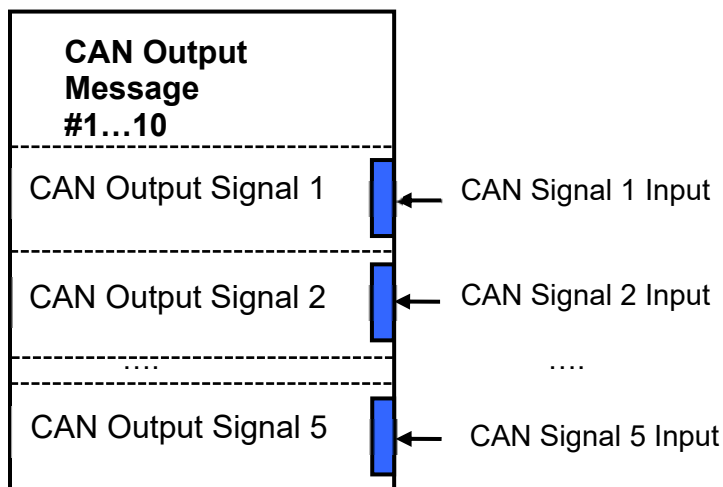


Figure 15. CAN Output Message Function Block

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

Table 19. CAN Output Message Function Block Configuration Parameters

Name	Default Value	Range	Units	Description
PGN	0xFF00	Any J1939 PGN value ¹	–	CAN message PGN.
Transmission Enable	No	{Yes, No}	–	Enables the CAN output message transmission.
Transmission Rate	0	[0;10000]	ms	CAN output message transmission rate. If 0 – transmission is upon request.

Name	Default Value	Range	Units	Description
Destination Address	0xFF	[0; 255]	–	Destination address of the PDU1 PGN messages.
Length	8	[0...8]	–	CAN message data frame length.
Priority	6	[0...7]	–	CAN message priority.
Signal #1 Type	Undefined	{Undefined, Discrete, Continuous}	–	Type of the 1-st CAN output signal.
Signal #1 Source	Not Connected	Any signal output of any function block or “Not Connected”	–	Input signal source of the 1-st CAN output signal.
Signal #1 Byte Position	1	[1...8]	–	Byte position of the 1-st CAN output signal.
Signal #1 Bit Position	1	[1...8]	–	Bit position of the 1-st CAN output signal.
Signal #1 Size	1	[1...32]	bit	Size of the 1-st CAN output signal.
Signal #1 Resolution	1	Any value	Signal Units / bit	Resolution of the 1-st CAN continuous output signal.
Signal #1 Offset	0	Any value	Signal Units	Offset of the 1-st CAN continuous output signal.
Signal #2 Type	Undefined	{Undefined, Discrete, Continuous}	–	Type of the 2-nd CAN output signal.
Signal #2 Source	Not Connected	Any signal output of any function block or “Not Connected”	–	Input signal source of the 2-nd CAN output signal.
Signal #2 Byte Position	1	[1...8]	–	Byte position of the 2-nd CAN output signal.
Signal #2 Bit Position	1	[1...8]	–	Bit position of the 2-nd CAN output signal.
Signal #2 Size	1	[1...32]	bit	Size of the 2-nd CAN output signal.
Signal #2 Resolution	1	Any value	Signal Units / bit	Resolution of the 2-nd CAN continuous output signal.
Signal #2 Offset	0	Any value	Signal Units	Offset of the 2-nd CAN continuous output signal.
...
Signal #5 Type	Undefined	{Undefined, Discrete, Continuous}	–	Type of the 5-th CAN output signal.
Signal #5 Source	Not Connected	Any signal output of any function block or “Not Connected”	–	Input signal source of the 5-th CAN output signal.
Signal #5 Byte Position	1	[1...8]	–	Byte position of the 5-th CAN output signal.
Signal #5 Bit Position	1	[1...8]	–	Bit position of the 5-th CAN output signal.

Name	Default Value	Range	Units	Description
Signal #5 Size	1	[1...32]	bit	Size of the 5-th CAN output signal.
Signal #5 Resolution	1	Any value	Signal Units / bit	Resolution of the 5-th CAN continuous output signal.
Signal #5 Offset	0	Any value	Signal Units	Offset of the 5-th CAN continuous output signal.

¹Proprietary A PGN (61184) is excluded. It is taken by Axiomatic Simple Proprietary Protocol and therefore cannot be used in function blocks.

Configuration parameters: *Signal #1...5 Byte Position* and *Signal #1...5 Bit Position*, together with the *Signal #1...5 Size* have the same meaning as in the *CAN Input Signal* 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 Resolution* and *Signal #1...5 Offset* configuration parameters and then saturated to the continuous signal code range defined in the J1939 standard.

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 Electronic Assistant Software

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

The user should use EA software version 4.9.76.0 or higher, which supports this converter firmware. The most recent EA software version can be downloaded from www.axiomatic.com website.

Before connecting to the converter, the user should first check whether the baud rate of EA matches the baud rate of the converter. It should be either J1939 standard 250kbit/s or 500kbit/s depending on the converter modification. The baud rate is displayed in the bottom-right corner of the EA screen.

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

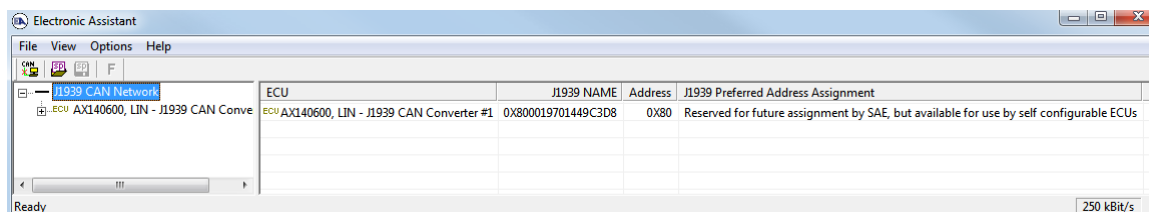


Figure 16. LIN - J1939 CAN Protocol Converter in EA

The user can then browse through the ECU parameters, read *General ECU Information* and *Bootloader Information* groups, view and modify configuration parameters, see: Figure 17.

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.

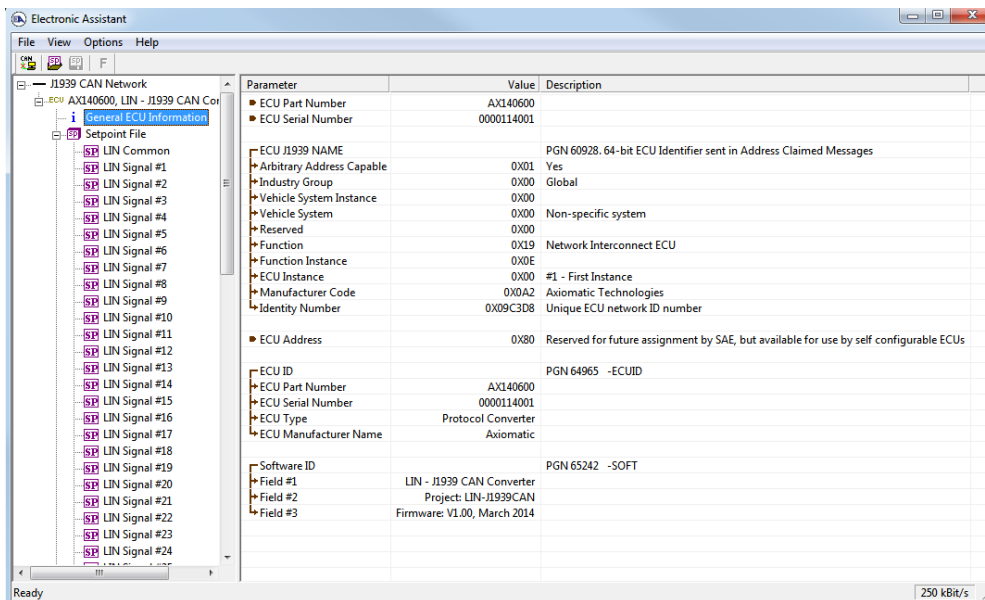


Figure 17. General ECU Information Screen

4.2 Function blocks in EA

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

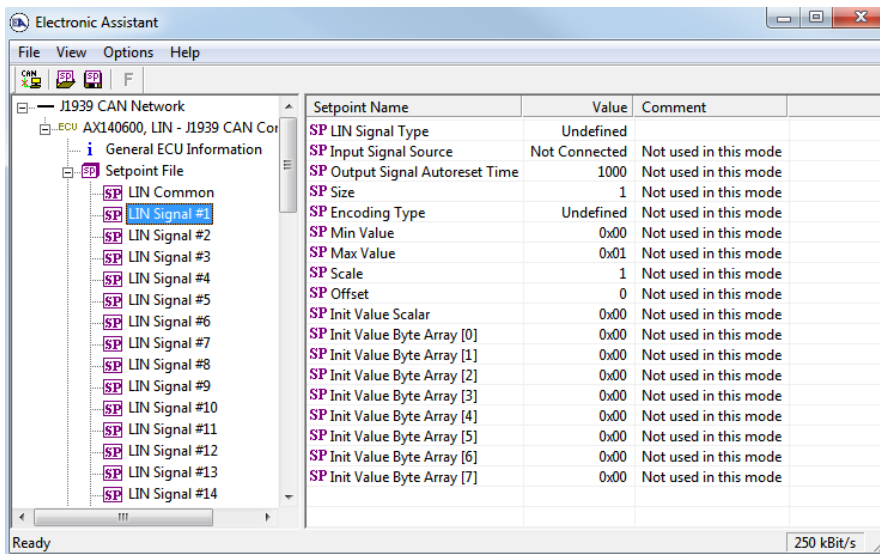


Figure 18. LIN Signal #1 Function Block in EA

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:

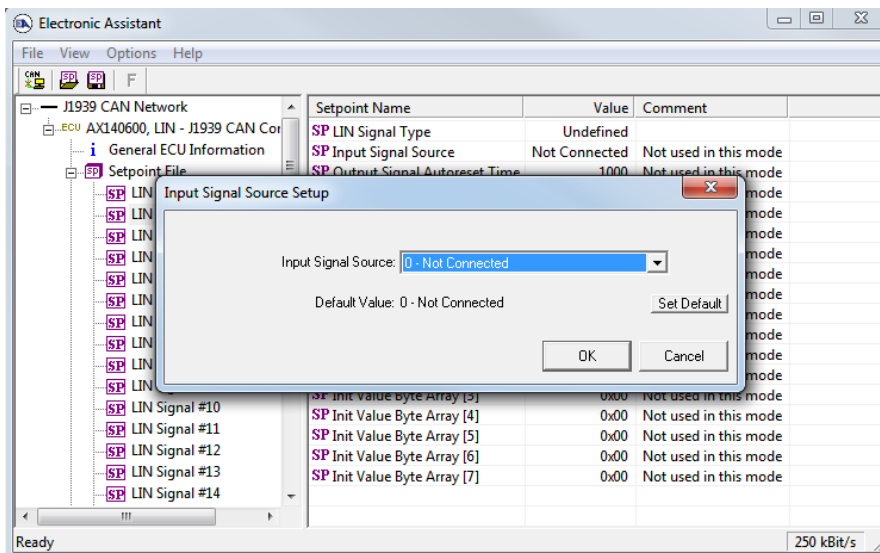


Figure 19. Changing a Configuration Parameter in EA

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 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.

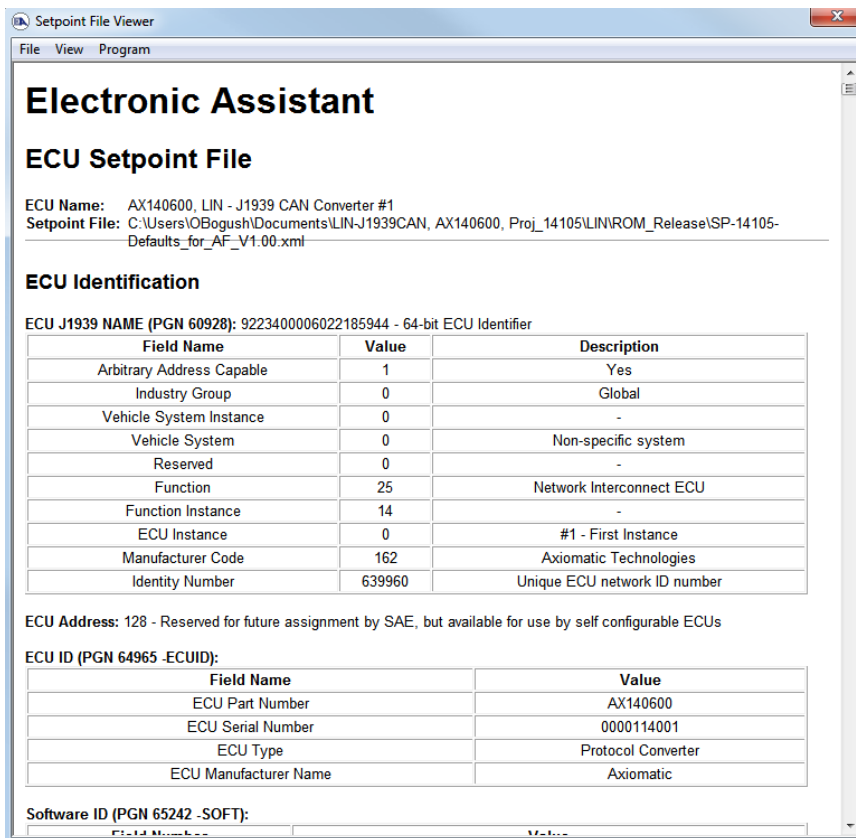


Figure 20. EA Setpoint File

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

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

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

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 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 21. Then, configure each individual function block, see Figure 21.

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 22.

Then configure *LIN Signal #2* as RedSaturation signal, see: Figure 23. This signal will have “CAN Input Signal #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 Input Signal #2” and “CAN Input Signal #3”, respectively, see: Figure 24 and Figure 25.

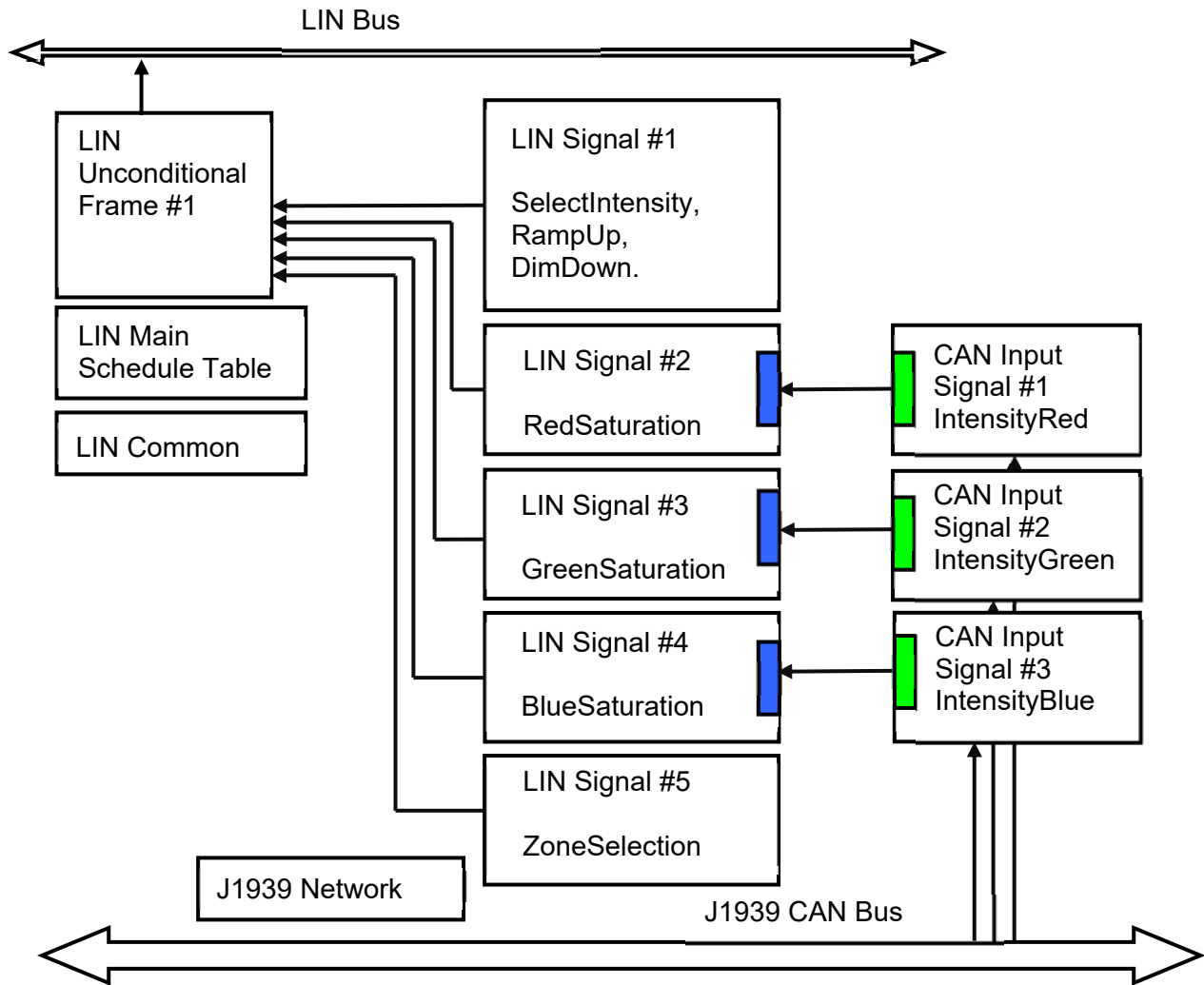


Figure 21. Block Diagram of the Example Converter Configuration

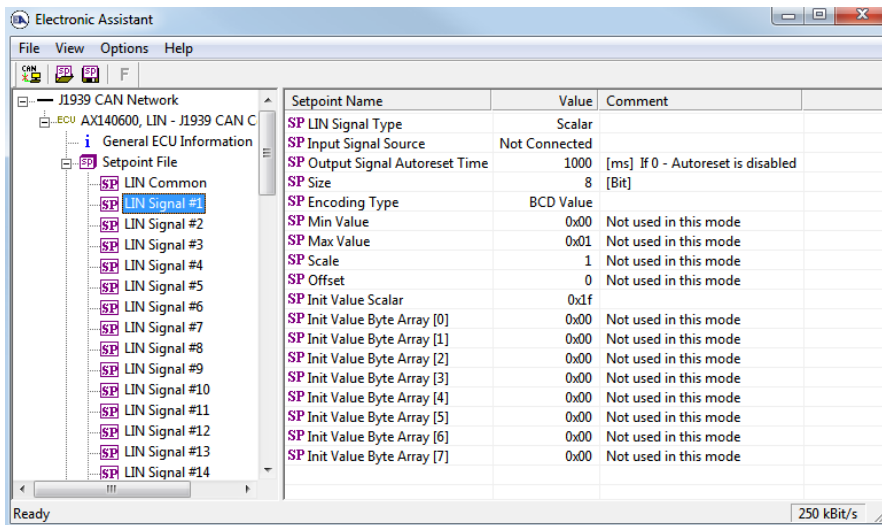


Figure 22. Example Configuration. LIN Signal #1

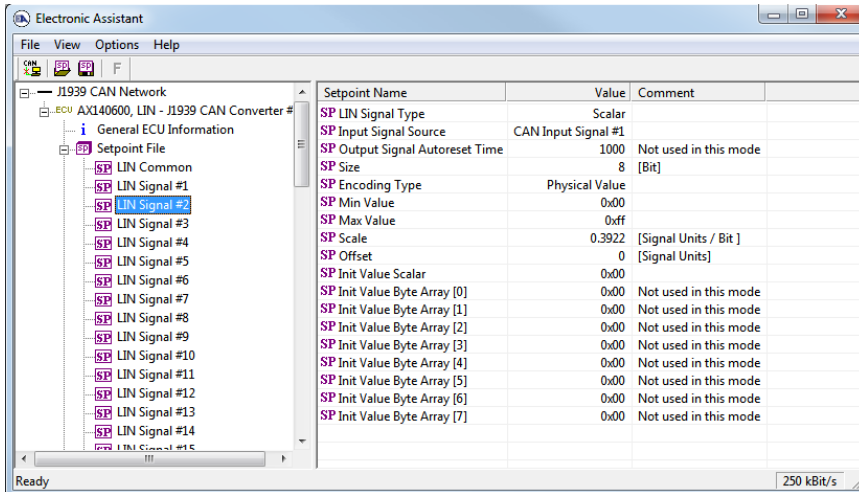


Figure 23. Example Configuration. LIN Signal #2

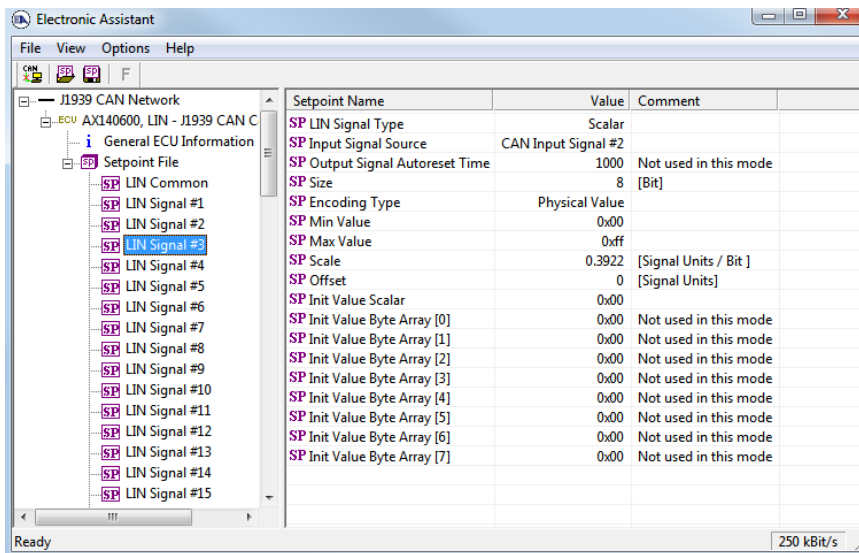


Figure 24. Example Configuration. LIN Signal #3

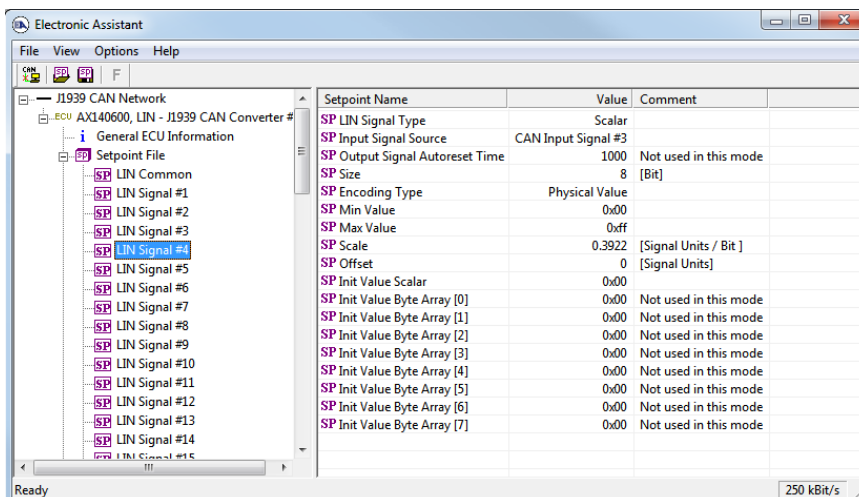


Figure 25. Example Configuration. LIN Signal #4

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.

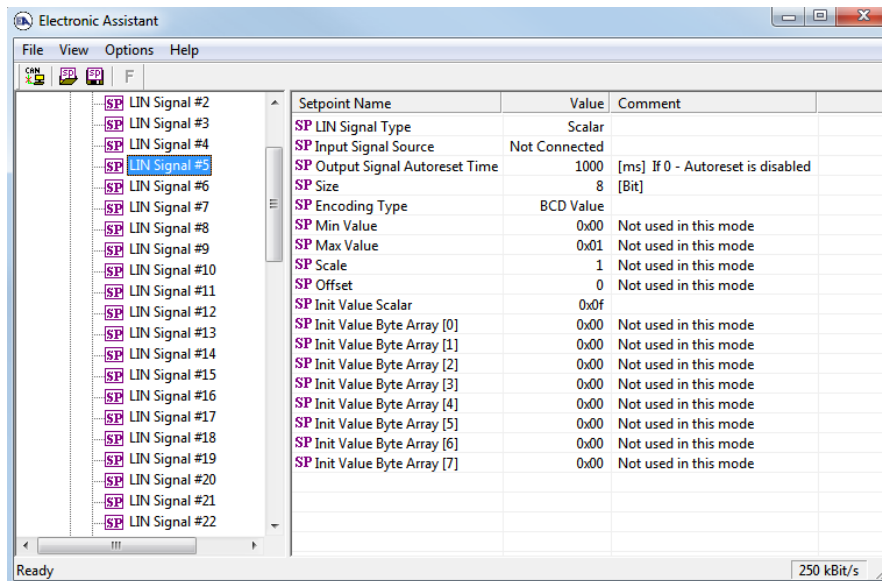


Figure 26. Example Configuration. LIN Signal #5

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.

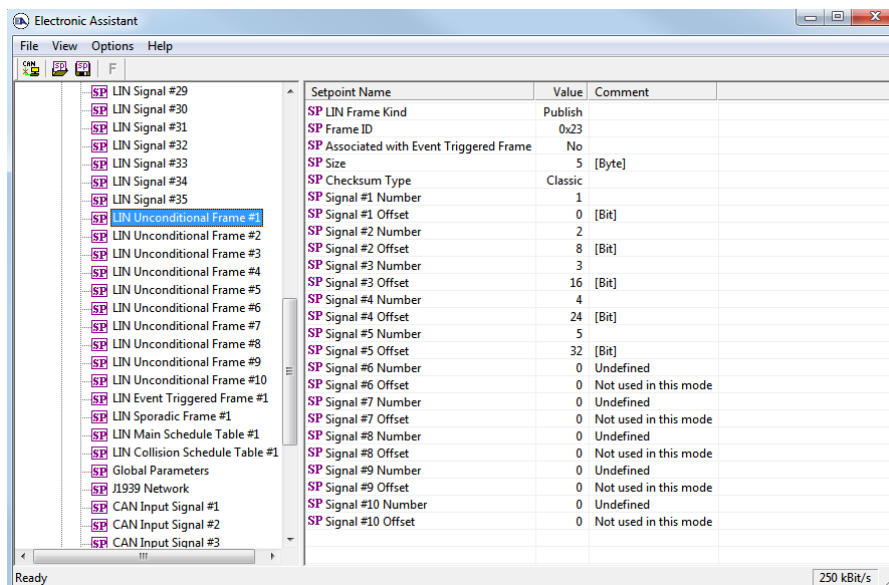


Figure 27. Example Configuration. LIN Unconditional Frame #1

Set *LIN Schedule Table #1* with only one entry: *LIN Unconditional Frame #1*. Set *Delay* to 50 ms, see Figure 28.

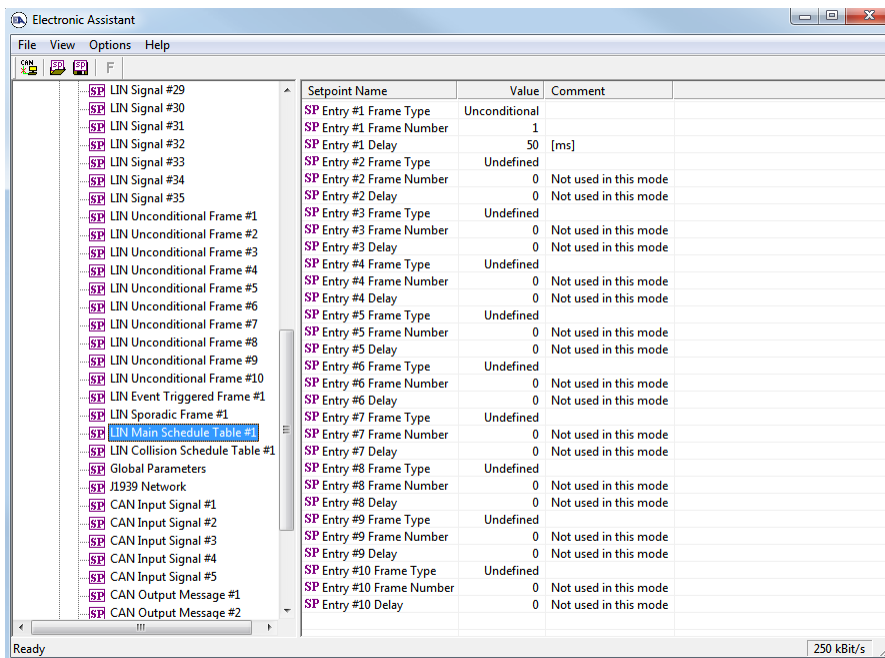


Figure 28. Example Configuration. LIN Main Schedule Table #1

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.

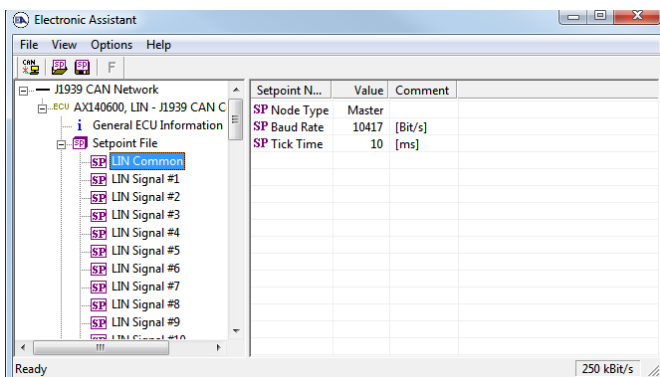


Figure 29. Example Configuration. LIN Common.

The LIN bus is now running. Configure CAN bus input signals. All three input signals will parse one CAN message. Start from *CAN Input Signal #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: Figure 30.

Configure *CAN Input Signal #2* and *CAN Input Signal #3* in a similar way as IntensityGreen and IntensityBlue signals, respectively, see: Figure 31, Figure 32.

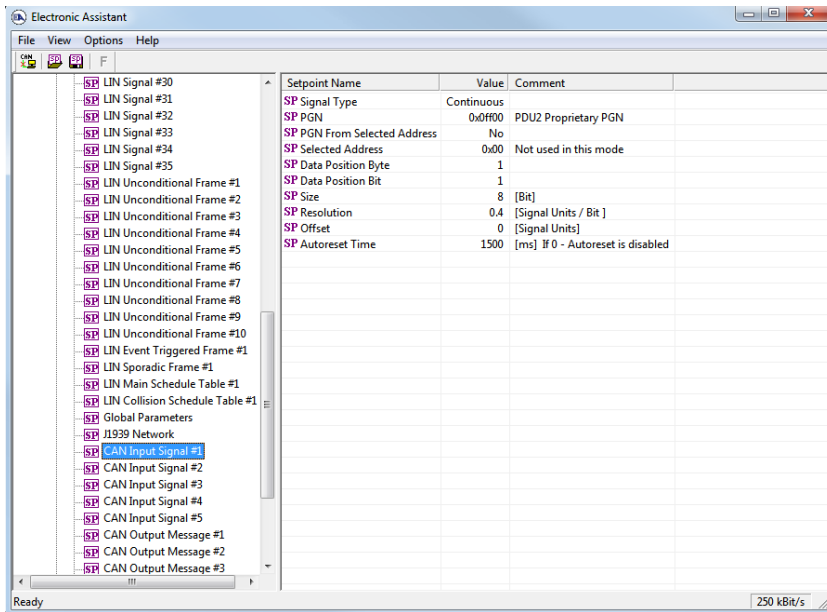


Figure 30. Example Configuration. CAN Input Signal #1

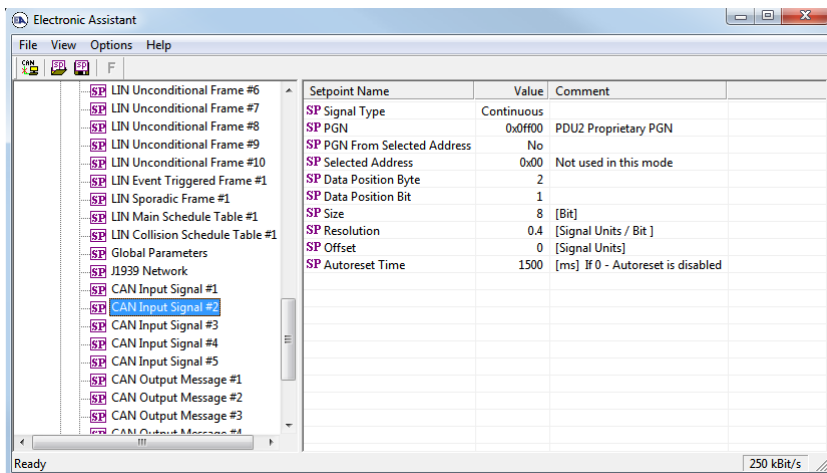


Figure 31. Example Configuration. CAN Input Signal #2

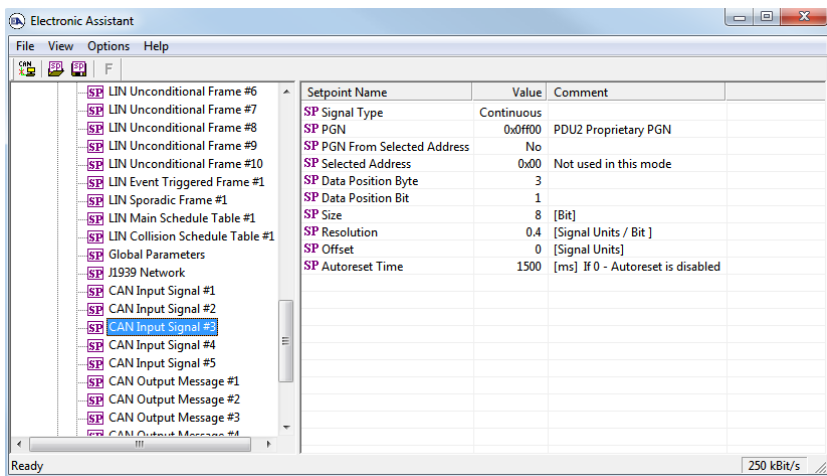


Figure 32. Example Configuration. CAN Input Signal #3

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.

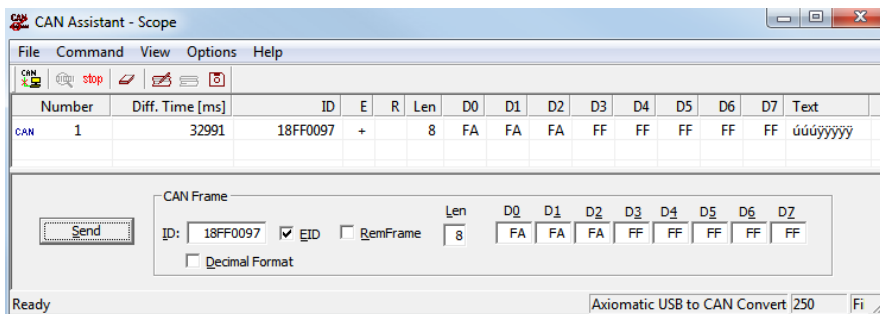


Figure 33. Example Configuration. Generating CAN Test Message

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

The user can flash the new converter firmware in the field using the unit embedded bootloader. In some seldom cases, incompatibility between an application firmware and the bootloader can exist¹.

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

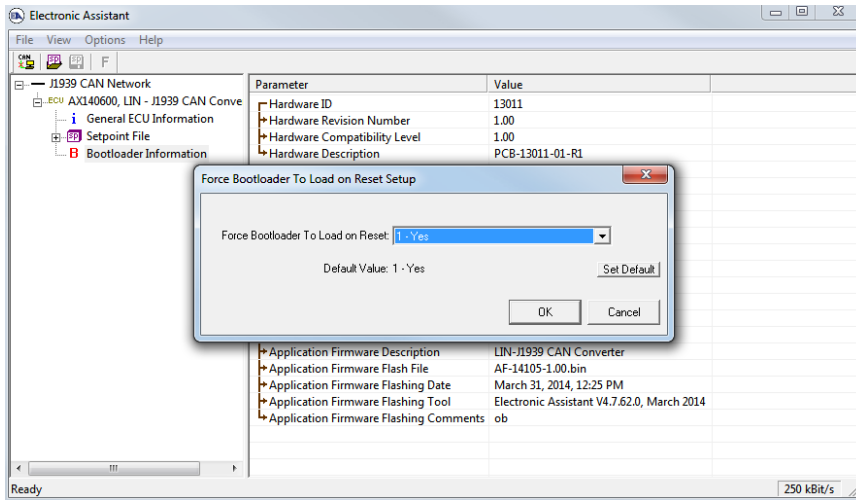


Figure 34. Bootloader Activation. First Step

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. Select “Yes”.

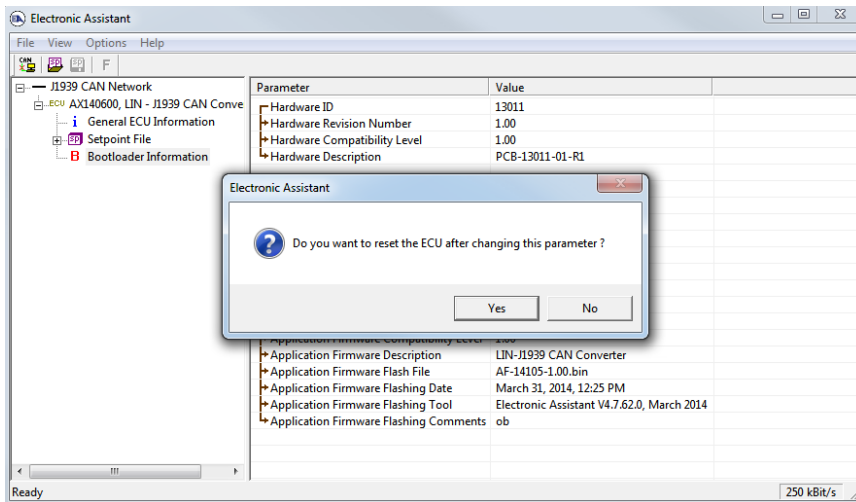


Figure 35. Bootloader Activation. Final Reset

¹ Converters with application firmware version 1.xx have a bootloader incompatible with other versions of the application firmware.

After automatic reset, instead of AX140600, LIN - J1939 CAN Converter, the user will see J1939 Bootloader ECU in the J1939 CAN Network top-level group in the EA. This means that the bootloader is activated and ready to accept the new firmware.

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

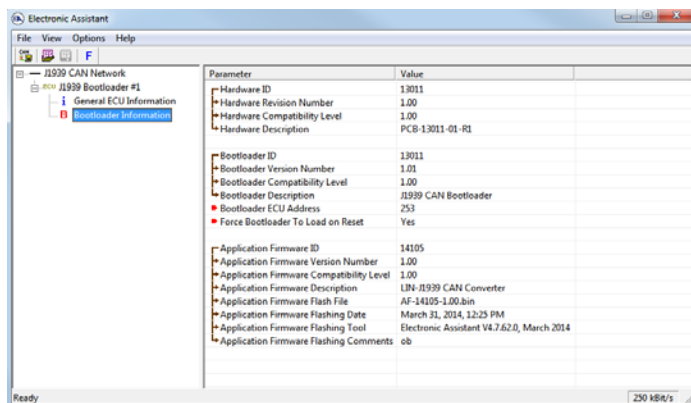


Figure 36. Bootloader Information Screen

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

To flash the new firmware, the user should click on **F** toolbar icon or from the *File* menu select the *Open Flash File* command. The *Open Application Firmware Flash File* dialog will appear. Pick up the flash file with the new converter firmware and confirm the selection by pressing the *Open* button. The *Flash Application Firmware* dialog window will appear².

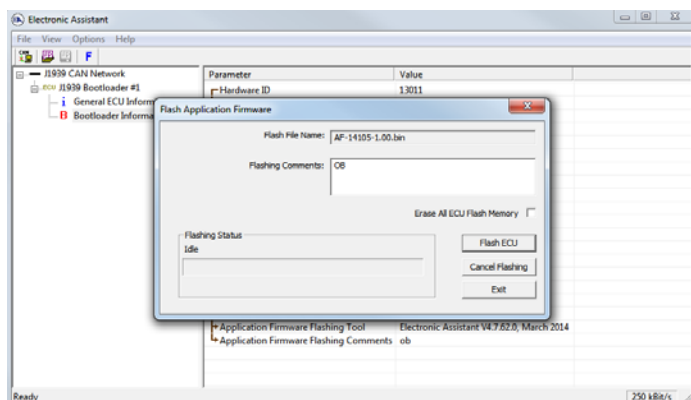


Figure 37. Flashing New Firmware. Preparation

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.

² In this example, instead of the new firmware the old firmware is being simply re-flashed.

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

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

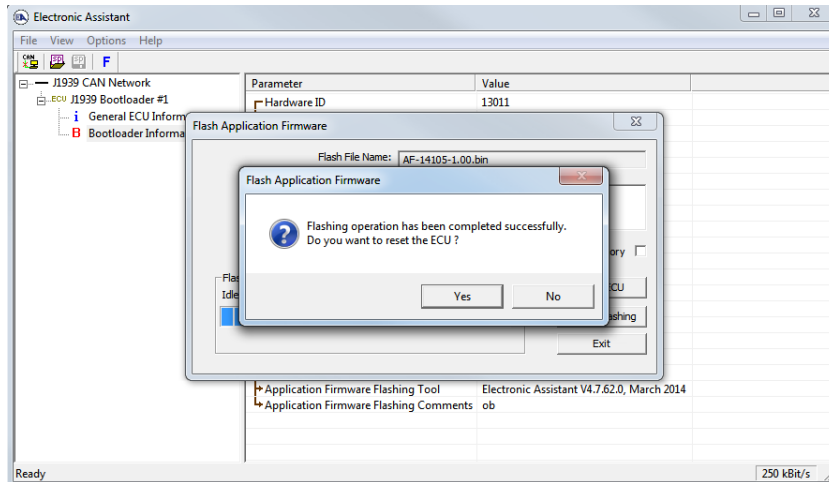


Figure 38. *Flashing New Firmware. Final Reset*

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

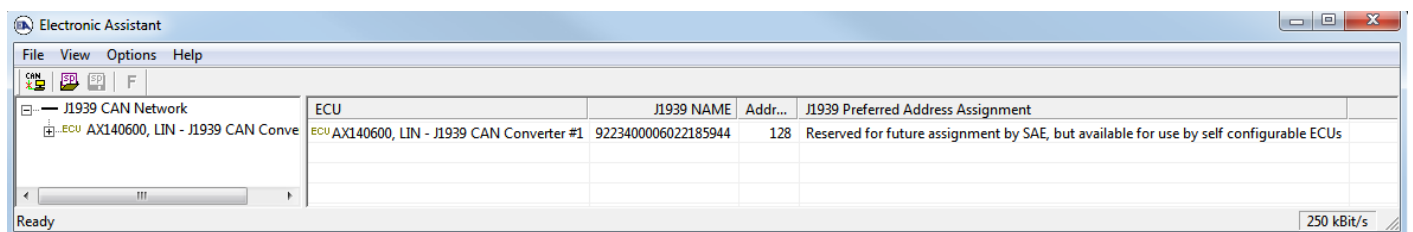


Figure 39. *Firmware has been Updated. New Firmware Screen*

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

6 TECHNICAL SPECIFICATIONS

Typical at nominal input voltage and 25 degrees C unless otherwise specified.

6.1 Power

Table 20. Power

Power Supply Input	12 V or 24 Vdc nominal; 9...32 Vdc
Surge Protection	Meets the surge requirements of SAE J1445
Reverse Polarity Protection	Provided

6.2 Control Software

Table 21. Control Software

Software Platform	The Protocol Converter comes pre-programmed with customer specific protocol conversion logic for data exchange between 1 LIN network and 1 CAN network (SAE J1939).
-------------------	---

6.3 General Specifications

Table 22. General Specifications

Microprocessor	32-bit, 128 KByte flash program memory																
LIN Port	1 LIN (LIN 2.2)																
CAN Port	1 CAN (SAE J1939, 250kbit/s AX140600 or 500kbit/s AX140600-01)																
Power Supply Current	40 mA @ 12V																
Operating Conditions	-40 to 85 °C (-40 to 185 °F)																
Weight	0.15 lbs. (0.068 kg)																
Protection Rating	IP67																
Vibration	Random Vibration: 7.68 Grms peak Sinusoidal Component: 10 g peak Based on MIL-STD-202G, Methods 204G and 214A																
Shock	50 g half sine pulse, 9ms per axis Based on MIL-STD-202G, Method 213B, Test Condition A																
Packaging and Dimensions	Plastic Enclosure, Nylon 6-6 with 30% glass fill Integral Deutsch IPD connector Refer to dimensional drawing (below).																
Electrical Connections	6 pin Deutsch IPD connector P/N: DT04-6P A mating plug kit is available as Axiomatic P/N: AX070119 <table border="1" data-bbox="483 1585 928 1822"> <thead> <tr> <th colspan="2">CAN and I/O Connector</th> </tr> <tr> <th>PIN #</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>CAN_SH (shield)</td> </tr> <tr> <td>2</td> <td>CAN_L</td> </tr> <tr> <td>3</td> <td>CAN_H</td> </tr> <tr> <td>4</td> <td>LIN</td> </tr> <tr> <td>5</td> <td>BATT+</td> </tr> <tr> <td>6</td> <td>BATT-</td> </tr> </tbody> </table>	CAN and I/O Connector		PIN #	Description	1	CAN_SH (shield)	2	CAN_L	3	CAN_H	4	LIN	5	BATT+	6	BATT-
CAN and I/O Connector																	
PIN #	Description																
1	CAN_SH (shield)																
2	CAN_L																
3	CAN_H																
4	LIN																
5	BATT+																
6	BATT-																
Software Reflashing	Electronic Assistant® AX070502																

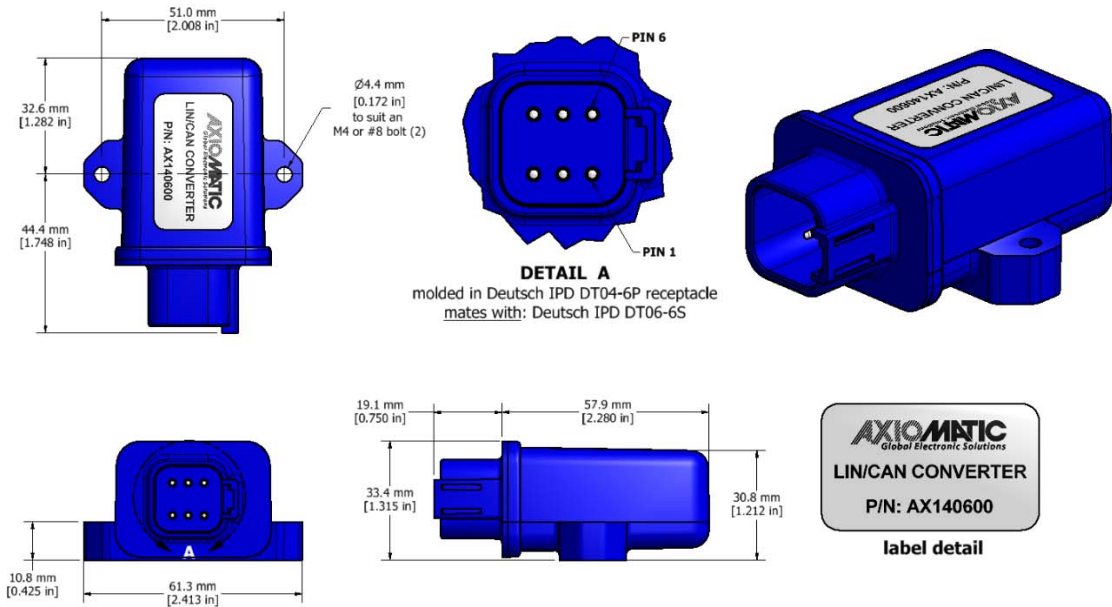


Figure 40. Unit Dimensions. AX140600

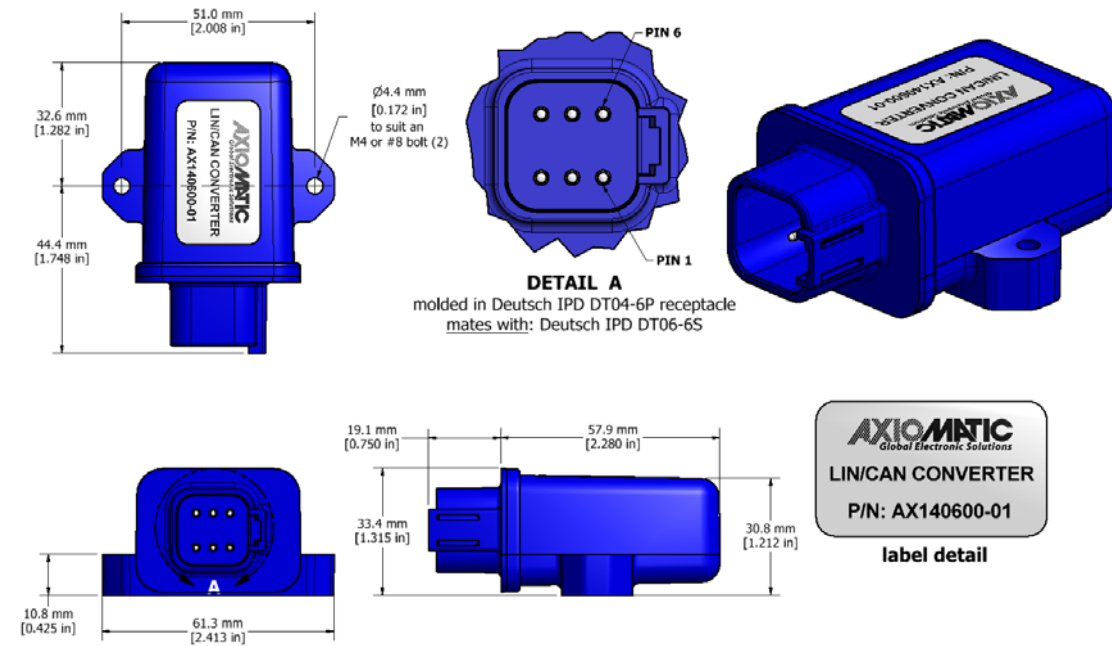


Figure 41. Unit Dimensions. AX140600-01

7 VERSION HISTORY

User Manual Version	Firmware version	Electronic Assistant® (EA) version	Date	Author	Modifications
2D	2.xx	4.9.76.0 or higher	Sept 14, 2020	Olek Bogush	<ul style="list-style-type: none"> Added notes excluding Proprietary A PGN (61184) from customer's use in function blocks. Updated <i>CAN Standard Implementation</i> table, <i>CAN Input Signal</i> and <i>CAN Output Message</i> function blocks.
2C	2.xx	4.9.76.0 or higher	June 3, 2020	Olek Bogush	<ul style="list-style-type: none"> Updated Finnish office phone number on the front page.
2B	2.xx	4.9.76.0 or higher	Feb 27, 2017	Olek Bogush	<ul style="list-style-type: none"> Added P/N: AX140600-01, LIN - J1939 CAN Protocol Converter – 500kbit/s Baud Rate. Corrected number of associated LIN frames in event-triggered LIN frame.
2A	2.xx	4.9.76.0 or higher	Feb 8, 2017	Olek Bogush	<ul style="list-style-type: none"> Updated <i>Continuous to Discrete Conversion</i> subsection. Renamed <i>LIN Input</i> subsection to <i>Receiving LIN Signals</i> subsection and updated its contents. Renamed <i>LIN Output</i> subsection to <i>Transmitting LIN Signals</i> subsection and updated its contents. Changed kBit/s to kbit/s, Bit/sec to bit/s. Bit to bit, Byte to byte in the unit representations.
			March 27, 2015	Amanda Wilkins	<ul style="list-style-type: none"> Added vibration and shock compliance Added updated dimensional drawing
2	2.xx	4.9.76.0 or higher	Jan 14, 2015	Olek Bogush	<ul style="list-style-type: none"> Added <i>Binary Function</i> blocks. Changed function signal conversion rules to allow conversion between <i>Discrete</i> and <i>Continuous</i> signals. Changed description and processing of <i>the LIN Signals</i>, encoding type: <i>Logical Value</i>. Changed: <i>Min Value</i> and <i>Max Value</i> LIN Signal configuration parameters. Added a note about the bootloader incompatibility with certain versions of application firmware. Corrected <i>Figure 24. Example Configuration. LIN Signal #3</i>. Named all tables, figures, and equations.

1D	1.xx	4.9.74.0 or higher	Nov 25, 2014	Olek Bogush	<ul style="list-style-type: none"> Corrected the <i>Destination Address</i> configuration parameter range in the <i>CAN Output Message</i> function block.
1C	1.xx	4.7.63.0 or higher	Sept 30, 2014	Olek Bogush	<ul style="list-style-type: none"> Corrected <i>Function Block Signals</i> subsection in <i>Converter Logical Structure</i>.
1B	1.xx	4.7.63.0 or higher	August 5, 2014	Olek Bogush	<ul style="list-style-type: none"> Clarified <i>CAN Interface</i> subsection in <i>Converter Description</i>. Corrected <i>Fig. 1</i>.
1A	1.xx	4.7.63.0 or higher	April 11, 2014	Olek Bogush	<ul style="list-style-type: none"> Corrected document section numbering. Clarified the EA version number in <i>Electronic Assistant Software</i> subsection.
1	1.xx	4.7.63.0 or higher	April 2, 2014	Olek Bogush	Initial release – draft version.



OUR PRODUCTS

Actuator Controls
Automotive Ethernet Converters
Battery Chargers
CAN bus Controls
CAN/Wifi, CAN/Bluetooth
Current/Voltage Converters
DC/DC Power Converters
Engine Temperature Scanners
Ethernet/CAN Converters, Switches
Fan Drive Controllers
Gateways, CAN/Modbus Protocols
Gyroscope Inclinometers
Hydraulic Valve Controllers
Inclinometers, Triaxial
I/O Controls
LVDT Simulators
Machine Controls
Modbus Controls
Motor Controls
Power Supplies
PWM Signal Converters/Isolators
Resolver Signal Conditioners
Service Tools
Signal Conditioners, Converters
Strain Gauge CAN Controls
Surge Suppressors

OUR COMPANY

Axiomatic provides electronic machine controls, components, and systems 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. We emphasize service and partnership with our customers, suppliers, and employees to build long term relationships and mutual trust.

QUALITY DESIGN AND MANUFACTURING

Axiomatic in Canada operates an ISO 9001:2015 registered design and manufacturing facility.

SERVICE

All products to be returned to Axiomatic require a Return Materials Authorization Number (RMA#). Please request an RMA# from sales@axiomatic.com.

Please provide the following information when requesting an RMA number:

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

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

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.

WARRANTY, APPLICATION APPROVALS/LIMITATIONS

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

CONTACTS

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

Axiomatic Technologies Oy
Höytämöntie 6
33880 Lempäälä
FINLAND
TEL: +358 103 375 750
FAX: +358 3 3595 660
www.axiomatic.fi